Plan-based Axiom Absorption for Tableau-based Description Logics Reasoning

Jiewen Wu

A THESIS

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE AND SOFTWARE ENGINEERING

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE AT

CONCORDIA UNIVERSITY

MONTRÉAL, QUÉBEC, CANADA

JULY 2008

Your file Votre référence
ISBN: 978-0-494-42537-4
Our file Notre référence
ISBN: 978-0-494-42537-4

Abstract

Plan-based Axiom Absorption for Tableau-based Description Logics Reasoning

Jiewen Wu

Description logic knowledge bases traditionally contain a set of axioms (Tbox) describing background knowledge. DL reasoners generally handle axioms by using the lazy unfolding technique, which reduces the nondeterminism introduced by axioms. Axiom absorption is an optimization technique that rewrites axioms into the unfoldable part of the Tbox suitable for lazy unfolding.

Absorptions are generally employed in DL reasoners in a mostly uniform way regardless of the characteristics of an input knowledge base. Though there exist a number of absorptions, their overall effectiveness remains to be improved, especially when a large quantity of complex axioms are present in the knowledge bases, which is well beyond the capability of any single absorption technique.

To ameliorate absorption techniques, this thesis presents a framework applying AI planning to axiom absorption. In this framework, a state space planner is used to encode state-of-the-art absorption techniques. Some designed heuristics concerning the characteristics of an input KB are utilized for the cost estimation during planning. The planner first applies appropriate absorptions to axioms, then it produces a solution with a minimized cost. Such a solution automatically organizes absorptions in a certain sequence to maximize the number of axioms for absorptions. Compared to a predetermined or fixed order of applying absorption techniques, the proposed framework benefits from the advantages to consider more absorption alternatives, which tends to be more flexible and effective.

Acknowledgments

I owe my deep gratitude and respect to my supervisor Prof. Volker Haarslev, Dr. rer.-nat. habil., whose wisdom, knowledge and experiences shape numerous insightful conversations, without which many ideas in our research would have not been well developed. His understanding, support and guidance helped me to make this thesis possible.

My father Fengchun Wu, my mother Yingliu Tian and my brother Jixiong Wu deserve my special thanks and affections. Their boundless love and dedication are always the inspiration throughout my life. To them I dedicate this thesis. I am particularly indebted to my entire extended family, who supports and encourages me all the way.

My warm thanks are due to Dr. Yu Ding, Alexander K. Hudek, Chih-Wei Hsu, Prof. Grant Weddell, and Ming Zuo for their discussions about some topics in this thesis. I am also grateful to Xi Deng and Dr. Hsueh-Ieng Pai for their comments and suggestions regarding publications, and to Francis Gasse for his contribution to the prototype of the reasoner.

I would like to express my gratitude to the faculty members and staff in our department for their teaching, research seminars and services. I also wish to extend my sincere thanks to my colleagues Ahmed Alasoud, Jocelyne Faddoul, Nasim Farsinia and Jinzan Lai for providing a stimulating and fun working environment.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Semantic Web [Berners-Lee et al., 2001], envisioned as an evolution of the existing Web and an infrastructure for knowledge exchange, gives rise to the popularity of ontologies that can be used to formalize digital contents over the Web. An ontology, formally defined by Gruber [1993] as an "explicit specification of a conceptualization", is designed for sharing resources and reusing components. Nowadays, ontologies can be seen in a variety of domains, such as bioinformatics, medicine, user interfaces, linguistics and so on. At present, ontologies are designed using languages such as the Web Ontology Language (OWL) [Dean and Schreiber, 2004] to represent terminologies in a machine-understandable manner.

As a consequence, reasoning services to obtain (implicit) knowledge from ontologies are indispensable for knowledge understanding. Typically, Description Logics (DLs) play a key role in such reasoning services because of their relationship with OWL. OWL, on the one hand, provides a powerful mechanism to describe terminologies; one the other hand, it restricts its constructors to allow for decidable reasoning. Particularly, OWL-DL, one sub-language of OWL, closely corresponds to the description logic $\mathcal{SHOIN}(\mathcal{D})$. Description logics are considered to well accommodate reasoning about ontologies in the context of the Semantic Web.

1.1 Motivation

1.1.1 DL Reasoning and Optimizations

It is hard for description logic reasoning to remain tractable without compromising the expressiveness of the logic. An intuition is that intractability may lead to limited practicality of realistic reasoning systems. For example, Schmidt-Schauss and Smolka [1991] proved that checking coherence and subsumption in the fundamental DL $\mathcal{ALC}$ are already PSPACE-complete problems. Nebel [1990] argued that a number of strategies, some of which restrict the expressiveness of DLs, can be used to deal with intractability and practicality. Nevertheless, restrictions imposed on the expressiveness of DLs might result in unpractical representation languages. Thus, Nebel [1990] further showed that although reasoning in theory may be intractable for the worst case, in practice DL reasoning can be handled in polynomial time with optimization techniques due to rare occurrences of worst cases. Actually, existing highly optimized reasoners demonstrated their acceptable reasoning performance, even for very expressive DLs [Horrocks, 2003]. The reason is that optimization techniques are essential for DL systems to be practical. Under the development of DLs, well-known DL reasoners such as Racer [Haarslev and Möller, 2001b], FaCT++ [Tsarkov and Horrocks, 2006], and Pellet [Sirin and Parsia, 2006] adapt and implement a wide range of optimization techniques.

Simultaneously, ontologies emerging from application domains, however, have the tendency to stress or even exceed the capabilities of highly optimized DL reasoners because they are too complicated (such as GALEN [Rector and Horrocks, 1997] or FMA [Rosse and Mejino, 2003]) so that dedicated optimization techniques are required. It is reasonable to believe that ontologies developed in relevant domains in the near future could even be harder.

Absorptions as Optimizations The difficulty for DL reasoners to reason about ontologies has various sources, among which the presence of a large number of axioms is considered to be one of the major factors [Tsarkov et al., 2007]. To better deal with axioms, DL reasoners employ a typical optimization technique, called absorption, to rewrite general axioms into special forms for easy manipulation while preserving their semantics. For instance, Horrocks and Tobies [2000] showed enhanced reasoning performance of FaCT by applying this rewriting technique.

It remains to be discussed why axioms significantly increase reasoning difficulty in the absence of certain optimizations. Since the underlying idea of tableau algorithms has not yet been presented, we briefly discuss why axioms could lead to an exponential growth of search spaces for a tableau

algorithm. A tableau algorithm proves the satisfiability of a concept C by constructing a *tableau*, which is represented by a labeled graph (usually a tree) with nodes corresponding to individuals (elements in the domain) and edges corresponding to relationships between individuals. For a DL axiom of the form $C \sqsubseteq D$, to ensure this axiom holds, a tableau algorithm adds a disjunction $\neg C \sqcup D$ to *every* node of the tableau completion graph.

Consequently, for every node in the graph a tableau algorithm has to randomly choose either $\neg C$ or D; such a choice is an obviously nondeterministic behavior, which degrades the performance when there exist a large number of axioms and nodes. For example, for a completion graph with i nodes and one axiom, it already leads to 2^i choices. However, various absorption techniques can be used to maximally reduce this kind of nondeterminism.

Although a variety of absorption techniques have been presented in the literature (see Chapter 3), few of them are universally applicable and effective alone. Some absorptions can be applied to almost all known ontologies, however they cannot absorb all the axioms for some complicated ontologies. For this reason, it is desirable to design some absorption techniques that can eliminate general axioms. Horrocks and Tobies [2000] demonstrated through empirical studies that absorptions which rewrite as many general axioms as possible *generally* outperform other strategies.

1.1.2 Objectives

A normal question would be whether there exists some absorption that is able to absorb every axiom. Although no explicit proof or explanation has ever been presented, previous works [Horrocks and Tobies, 2000; Hudek and Weddell, 2006; Baader et al., 2006; Motik et al., 2007; Tsarkov et al., 2007] about optimizing DL reasoning over axioms have observed or implied that some axioms cannot be absorbed by any present absorptions. We conjecture that such axioms are likely *inherently unabsorbable* for DL reasoning, as we will detail in Section 3.3. The possibility that some general axiom cannot be absorbed is due to the characteristics of the domains to be modeled. So far as we know, one typical example, presented by Areces et al. [1999], is the BCS 5 ontology modeling feature interaction problem in the telecommunication domain, whose general axioms fail any known absorption techniques.

Why absorption techniques that have been used in DL reasoners are not always satisfactory? First, we are aware that absorptions are nondeterministic, that is, the same absorption can absorb an axiom in a number of ways, each of which might lead to a different level of difficulty for DL

reasoning. The randomness of axiom absorption grows markedly when various absorptions are applied to all axioms of some KB. We believe that among the many possibilities of absorptions some of them can result in a more preferable KB in terms of its runtime performance of DL reasoning. Secondly, not all absorptions are extensively implemented in present DL reasoners, for example, FaCT++ at present uses two main absorption techniques out of many choices. There could be many reasons why DL reasoners do not use all the absorption techniques. However, one typical reason may be that no single absorption always generates desired results for arbitrary input KBs. In order to utilize several absorption techniques in one DL reasoner in an effective manner, the developer must decide how to organize the applications of these techniques by using some heuristics based on the input KBs.

Bearing the above observations in mind, we, from a different perspective, present our solution featured by taking advantage of existing absorption techniques rather than inventing novel ones. The idea is to apply classical planning to different absorption techniques to solve the problems we have considered above. On the one hand, planning well offsets the limitation of one absorption technique by employing another one, for instance, if some axiom cannot be absorbed by the first absorption, planning will try another absorption. On the other hand, the cost metric that maps the hardness of an input knowledge base approximately can lead to heuristic absorptions for the input KB. In summary, a solution produced by planning will automatically organize absorptions in a certain way to meet a predefined goal.

1.2 Contributions

This thesis collects the researches done since the start of the author's program. The contributions can be summarized as follows:

1. Contributions to specific absorption techniques. First, the absorption technique **IRBA** (cf. Section 3.2.3) using the property of constraint back-propagation inverse roles has been derived from our research paper [Ding, Haarslev, and Wu, 2007]. Second, we have extended binary absorption [Hudek and Weddell, 2006] in several aspects to make it more suitable and adaptable in general cases, as described in Chapter 4.

2. The main contribution of this thesis is our proposed framework of applying classical planning to axiom absorption [Wu and Haarslev, 2008]. To the best of our knowledge, our framework

is the first to propose a cost-based organization of absorptions to facilitate DL reasoning. In addition, this thesis also provides experimental results using our system PAR, which implements our proposed framework.

1.3 Thesis Organization

There are occasions that readers may not read this thesis in the way as is presented. Hence, Figure 1 delineates the dependency between different chapters.

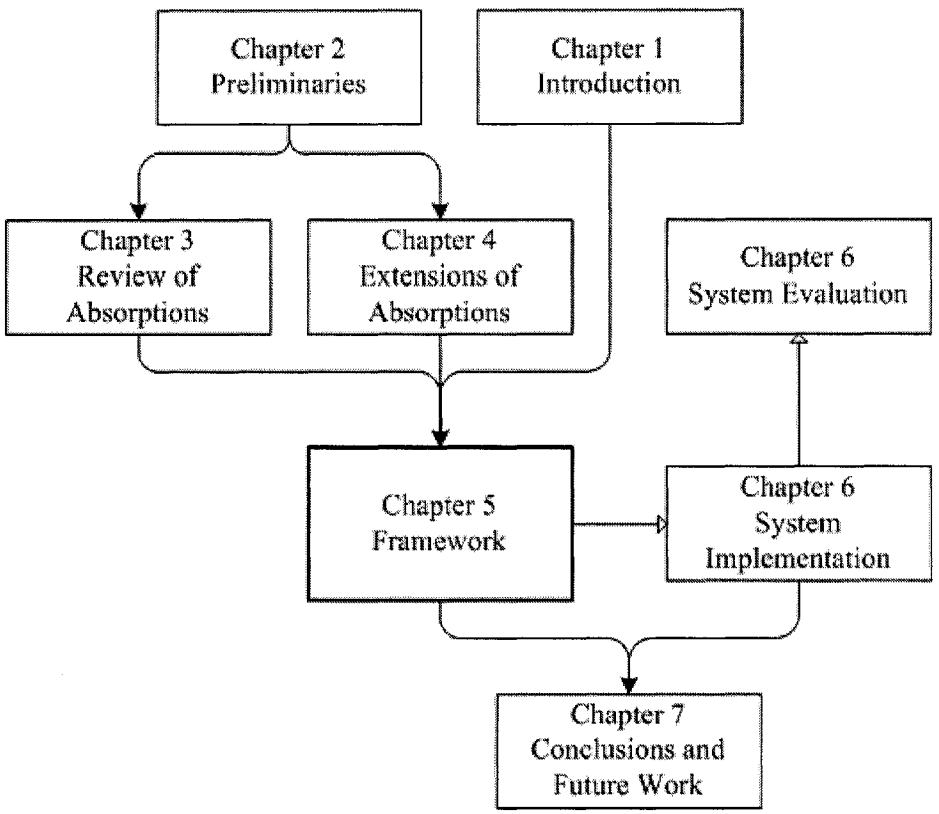

Figure 1: Thesis Organization

The thesis starts with the motivation (this chapter) to demonstrate the significance of studying absorption techniques using classical planning. Then, the thesis presents the necessary background (Chapter 2) on description logics and classical planning. If readers already have sufficient knowledge of these topics, they can directly jump to Chapter 3 for a complete survey of state-of-the-art absorptions. Chapter 4 details how binary absorption can be extended to a general n-ary variant. At this point all the preliminaries for the proposed framework have been presented. Consequently, Chapter 5 describes the framework for planning of axiom absorption, the central part of this thesis. An

implementation of the framework as well as empirical studies are described in Chapter 6. Chapter 7 concludes this thesis and discusses possible refinements to the framework.

Chapter 2

Preliminaries

This chapter mainly presents some basic notions for description logics and planning. Section 2.1 discusses how a DL knowledge base is formed (Section 2.1.1) and the syntax, semantics, and inferences of the fundamental DL $\mathcal{AL}$ and its extensions (Section 2.1.2). At the end of Section 2.1, a tableau algorithm for $\mathcal{ALC}$ is presented. Following that, Section 2.2 provides a summary of general planning, focusing on classical planning. Section 2.2.1 introduces classical planning and its restrictions, then Section 2.2.2 provides a comparison between state and plan space planning. Section 2.2.3 summarizes the complexity of classical planning.

2.1 Description Logics

Description logics, a family of knowledge representation (KR) formalisms, date back to the late 1970's with the initial name *terminological systems* to place emphasis on their definitions of the basic terminology in an application domain [see Baader et al., 2003, Chapter 1]. Later, DLs were also labeled *concept languages* to emphasize their concept constructs in the language. The recent advances in underlying logical systems introduce the name *Description Logics*, indicating that DLs are equipped with a formal logic semantics.

DLs mostly study a decidable fragment of the first-order predicate logic (FOL). Thus, decidability, complexity and expressiveness w.r.t. the reasoning problems characterize the theoretical research on DLs, for example, to seek *decision procedures* which always terminate in "reasonable" time. As briefly discussed in Section 1.1, the fundamental DL $\mathcal{ALC}$, in which atomic concepts correspond to unary predicates and atomic roles to binary predicates in FOL, is a notational variant of the

propositional *modal logic* $K_{(m)}$, revealed by Schild [1991]. By this correspondence to modal logic, theoretical results for $K_{(m)}$ are directly applicable to $\mathcal{ALC}$, and thus testing subsumption in $\mathcal{ALC}$ is PSPACE-complete [Schild, 1991]. It is easy to see that very expressive DLs will probably be even undecidable while weak DLs may be insufficient to model an application domain. Meanwhile, practical DL systems require an efficient reasoner without compromising the expressiveness of logics. In fact, this is achieved through the use of optimization techniques since a high worst-case complexity is rather common.

2.1.1 DL Knowledge Base

A *knowledge base* (KB) can be viewed as a mechanism storing machine-readable knowledge for reasoning. Such knowledge is defined by some languages, like DLs, to characterize a KB precisely. A KB distinguishes between *intensional knowledge* and *extensional knowledge* of a domain, which is clearly reflected in DL knowledge bases. For a DL knowledge base, intensional knowledge, contained in a TBox, describes the structure of the domain, while extensional knowledge, expressed as an ABox, describes instances in the domain [Baader et al., 2003]. Later we shall see that the distinction between a TBox and an ABox can be blurred by introducing the logic constructor $\mathcal{O}$.

TBox The TBox contains knowledge as a **T**erminology with concepts specifying common properties of instances. It is represented by a set of axioms. A concept in a TBox is defined in terms of previously defined concepts. For example, concepts GraduateStudent (1a) and Chair (1b) are defined as follows, taken from [Guo et al., 2005].

$$\text{GraduateStudent} \sqsubseteq \text{Person} \sqcap \exists\ \text{takesCourse.GraduateCourse} \tag{1a}$$

$$\text{Chair} \equiv \text{Person} \sqcap \exists\ \text{headOf.Department} \tag{1b}$$

In this example, GraduateStudent has only *necessary conditions*, but Chair has both necessary and *sufficient conditions*. In this thesis, concept definitions of the former form (i.e., only with necessary conditions) are called *inclusions*; concept definitions of the second form are *equivalences*, which are abbreviations for a pair of necessary and sufficient conditions for a concept. Both inclusions and equivalences are *axioms*. Further, there are two assumptions made about a DL TBox [see Baader et al., 2003, Chapter 1] for this TBox to be *unfoldable*:

- Only one unique definition for every concept name in this TBox is allowed. That is, a concept

name can occur at most once on the left-hand side (LHS) of all axioms in the TBox. Note that several axioms could be merged into one single axiom, such as $A \sqsubseteq C_1$ and $A \sqsubseteq C_2$ are combined into $A \sqsubseteq C_1 \sqcap C_2$.

- The TBox is *acyclic*. In other words, concepts are not defined in terms of themselves, whether directly or indirectly.

Suppose a TBox $\mathcal{T}_1$ only consists of axioms (1a) and (1b), then $\mathcal{T}_1$ is unfoldable, while any TBox containing the following axiom (2) becomes *not* unfoldable.

$$\text{Human} \sqsubseteq \exists \text{ hasParents.Human} \tag{2}$$

ABox ABoxes are formed by **A**ssertions about individuals of the universe of discourse. For instance, it can be asserted that JASON is a person who took a graduate course BIOINFORMATICS, as shown below.

$$\text{Person}(\text{JASON}), \quad \text{GraduateCourse}(\text{BIOINFORMATICS}), \tag{3a}$$

$$\text{takesCourse}(\text{JASON}, \text{BIOINFORMATICS}). \tag{3b}$$

Assertions of the form (3a) specifying that individuals are instances of some concept are called *concept assertions*. Differently, assertions of the form (3b) specifying the relationship between individuals are thus called *role assertions*, where *roles* indicate relationships between individuals.

2.1.2 Basic Logic $\mathcal{AL}$ and Its Extensions

So far, we already know that a DL KB consists of two (possibly empty) components: a TBox and an ABox. Description logics are then used to build these two components. However, DLs consist of a family of logics, which are distinguished by the constructors they provide. In this section, we first introduce the fundamental DL $\mathcal{AL}$, then additional concept constructors are introduced to describe more complex concepts.

$\mathcal{AL}$ Language

The fundamental DL $\mathcal{AL}$, the *attributive language* [Schmidt-Schauss and Smolka, 1991], describes concepts inductively as follows.

Syntax Note that A, B denote concept names or atomic concepts, C, D arbitrary concepts or concept expressions, R role names or atomic roles.

$$
\begin{aligned}
C, D := \; & A \mid && \text{(atomic concept)} \\
& \top \mid && \text{(top concept)} \\
& \bot \mid && \text{(bottom concept)} \\
& \neg A \mid && \text{(atomic negation)} \\
& C \sqcap D \mid && \text{(conjunction)} \\
& \forall R.C \mid && \text{(universal restriction)} \\
& \exists R.\top \mid && \text{(limited existential quantification)}
\end{aligned}
$$

In $\mathcal{AL}$, negations are only applied to atomic concepts, and existential quantification is limited to top concept over some role. However, these restrictions can be relaxed in more expressive logics.

Semantics The formal semantics of $\mathcal{AL}$ concepts is defined by an *interpretation* $\mathcal{I}$. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of the domain $\Delta^{\mathcal{I}}$, a non-empty set as the universe of discourse, and a mapping function $\cdot^{\mathcal{I}}$, which assigns to every concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every role R a set $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Specifically, the function $\cdot^{\mathcal{I}}$ is extended to concepts as follows.

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

Concept Constructors

More expressive DLs are obtained by adding more concept constructors to $\mathcal{AL}$. Though there exist a number of constructors, only those of our particular interest, like $\mathcal{I}$ [Horrocks et al., 1999] and $\mathcal{O}$ [Schaerf, 1994], are presented here in Table 1.

Constructor Name	Syntax	Semantics
Disjunction ($\mathcal{U}$)	$C \sqcup D$	$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Full Existential Quantification ($\mathcal{E}$)	$\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
Negation ($\mathcal{C}$)	$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
Inverse Role ($\mathcal{I}$)	R^-	$(a,b) \in R^{\mathcal{I}}$ iff $(b,a) \in R^{-\mathcal{I}}$
Nominal ($\mathcal{O}$)	$\{a_1, \ldots, a_n\}$	$\{a_1, \ldots, a_n\}^{\mathcal{I}} = a_1^{\mathcal{I}} \cup \ldots \cup a_n^{\mathcal{I}}$

Table 1: Concept Constructors. R^- is used to denote the inverse of role R and a_k $(1 \leq k \leq n)$ refers to some individual in the ABox.

A combination of $\mathcal{AL}$ with above constructors yields more expressive logics:

$$\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{C}][\mathcal{I}][\mathcal{O}],$$

however, it should be noted that some strings may identify the same logic. For example, negation ($\mathcal{C}$) can be used to express disjunction ($\mathcal{U}$) and full existential quantification ($\mathcal{E}$): $C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$ and $\exists R.C \equiv \neg\forall R.\neg C$. Hence, we could use $\mathcal{C}$ to replace $\mathcal{UE}$ in language names, for instance, we write $\mathcal{ALC}$ instead of $\mathcal{ALUE}$.

One more note is that by introducing nominals ($\mathcal{O}$) we are able to define *concepts* in terms of *individuals*, i.e., a TBox is then not separate from an ABox. Though such concepts are collections of individuals, they can be used wherever an ordinary concept can be used. An interpretation not only maps concepts and roles to sets and relations, but in addition maps individual names to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Moreover, an implicit assumption can be imposed on this mapping: the *unique name assumption* (UNA), i.e., $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if a, b are different individual names.

Semantics of Axioms and Assertions

We have seen in Section 2.1.1 examples for axioms and assertions. Here we present the semantics of axioms in a TBox and assertions in an ABox using an interpretation $\mathcal{I}$, as summarized in Table 2.

The semantics for axioms and assertions are quite straightforward, which together provide the semantics for the knowledge base. Conventionally, an axiom $C \sqsubseteq D$, where C and D are arbitrary concepts, is called a *general concept inclusion* (GCI). Then, the equivalence axiom $C \equiv D$ is considered as an abbreviation for the two GCIs $\{C \sqsubseteq D,\ D \sqsubseteq C\}$. In this sense, we say that a set of

Name	Syntax	Semantics
inclusions	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
equivalences	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
concept assertions	C(a)	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertions	R(a, b)	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

Table 2: Semantics of Axioms and Assertions. R denotes some role and a, b refer to some individuals in the ABox.

GCIs forms a TBox. Note that the semantics for a GCI is the same as for any other axiom, i.e., an interpretation $\mathcal{I}$ *satisfies* a GCI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.

Inferences

A description logics KB stores knowledge in its TBox and ABox, hence it is very likely to contain implicit knowledge that can be made explicit through inferences. For instance, we can infer that JASON is a graduate student because this individual complies with the definition of GraduateStudent from the example in Section 2.1.1. Formally, we can add an assertion to the ABox: GraduateStudent(JASON), which has never been explicitly stated in that KB.

Although a DL system is able to infer implicit knowledge from both the TBox and the ABox, in this thesis we are only interested in TBox inferences. Consequently, only reasoning tasks regarding TBoxes or concepts are discussed in this section. Intuitively, a concept only makes sense when it denotes a nonempty set in the domain of discourse. Following this intuition, we proceed by defining the satisfiability of a concept, which is the key inference for a TBox $\mathcal{T}$.

Satisfiability Concept C is *satisfiable* with respect to $\mathcal{T}$ ("w.r.t. a TBox $\mathcal{T}$" is dropped if $\mathcal{T}$ is clear from the context, as is the case in this thesis) if there exists an interpretation $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is nonempty. Such an interpretation $\mathcal{I}$ is a *model* of C.

Subsumption Concept C is *subsumed* by concept D if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for every model of $\mathcal{T}$, denoted by $\mathcal{T} \models C \sqsubseteq D$.

Equivalence Two concepts C and D are *equivalent* if $C^{\mathcal{I}} = D^{\mathcal{I}}$ holds for every model of $\mathcal{T}$, denoted by $\mathcal{T} \models C \equiv D$.

Disjointness Two concepts C and D are *disjoint* if $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$ holds for every model of $\mathcal{T}$.

The inference of satisfiability shows how to define a model of a concept. We can also state that an interpretation $\mathcal{I}$ satisfies a Tbox $\mathcal{T}$ if it satisfies every GCI in $\mathcal{T}$, and such an interpretation is called a *model* of $\mathcal{T}$. Moreover, $\mathcal{T}$ is said to be consistent if there exists a model of $\mathcal{T}$. Actually, a subsumption test could become the basic inference provided by DL reasoners due to the fact that other inferences can be reduced to the subsumption check between concepts, as shown in Proposition 2.1.1. It should be noted that all these inferences can be reduced to (un)satisfiability as well.

Proposition 2.1.1. *(Reduction to Subsumption) The following statements hold for arbitrary concepts (with respect to a TBox):*

- *C is unsatisfiable iff C is subsumed by $\bot$;*
- *C and D are equivalent iff C is subsumed by D and D is subsumed by C;*
- *C and D are disjoint iff $C \sqcap D$ is subsumed by $\bot$.*

Tableau Algorithm

We have seen so far the key inferences considered for TBox reasoning, which can all be reduced to testing subsumption, i.e., whether $C \sqsubseteq D$ holds. Alternatively, such a subsumption test amounts to check whether $C \sqcap \neg D$ is *unsatisfiable*. This underlying idea yields a widely used algorithm for DL reasoners, the tableau[1] algorithm.

A tableau-based satisfiability algorithm for $\mathcal{ALC}$ tries to construct a *tableau* by applying a set of tableau rules. We present below (Table 3) the tableau rules for basic $\mathcal{ALC}$. The tableau rules for more expressive logics are beyond the scope of this thesis.

A tableau algorithm works on a completion graph (usually a tree) [See Horrocks and Sattler, 2005, Definition 5], where every node a represents some element of $\Delta^{\mathcal{I}}$ and every edge $\langle a, b\rangle$ denotes some role relationship. Both nodes and edges are labeled by concepts and roles respectively, for example, $C \in \mathcal{L}(a)$ means $C(a)$; $R \in \mathcal{L}(\langle a, b\rangle)$ indicates $R(a, b)$. The completion graph is then expanded using tableau rules. Hence, these tableau rules are also called completion rules. Particularly, the **$\sqcup$-rule** is *non-deterministic* in the sense that a tableau has to randomly choose C_1 or C_2 as C. A tableau is called *complete* when either a clash, i.e., $\{A, \neg A\} \subseteq \mathcal{L}(a)$ or $\bot \subseteq \mathcal{L}(a)$, is encountered or no more tableau rules are applicable to it. In the latter case, the completion tree is clash-free, and we can state that the given concept is satisfiable.

[1] As a rule, *tableau* is used as an adjective and *tableaux* is used only to refer to more than one tableau, however, they are considered interchangeable in this thesis.

$\sqcap$-rule	
Conditions:	$C_1 \sqcap C_2 \in \mathcal{L}(a)$ and $\{C_1, C_2\} \not\subseteq \mathcal{L}(a)$
Actions:	$\mathcal{L}(a) = \mathcal{L}(a) \cup \{C_1, C_2\}$
$\sqcup$-rule	
Conditions:	$C_1 \sqcup C_2 \in \mathcal{L}(a)$ and $\{C_1, C_2\} \cap \mathcal{L}(a) = \emptyset$
Actions:	$\mathcal{L}(a) = \mathcal{L}(a) \cup \{\ C\}$ for some $C \in \{C_1, C_2\}$
$\exists$-rule	
Conditions:	$\exists R.C \in \mathcal{L}(a)$ and there is *no* individual c s.t. $R \in \mathcal{L}(\langle a, c\rangle)$ and $C \in \mathcal{L}(c)$
Actions:	create a new node b with $\mathcal{L}(\langle a, b\rangle) = \{R\}$ and $\mathcal{L}(b) = \{C\}$
$\forall$-rule	
Conditions:	$\forall R.C \in \mathcal{L}(a)$ and there is an individual b with $R \in \mathcal{L}(\langle a, b\rangle)$ and $C \notin \mathcal{L}(b)$
Actions:	$\mathcal{L}(b) = \mathcal{L}(b) \cup \{C\}$

Table 3: Tableau Rules for $\mathcal{ALC}$. R denotes some role, C, C_1, C_2 concepts and a, b, c individuals.

We conclude this section by showing soundness and completeness [Schmidt-Schauss and Smolka, 1991] of the tableau algorithm we have introduced.

Proposition 2.1.2. *An $\mathcal{ALC}$ concept is satisfiable iff it has a clash-free completion tree by applications of tableau rules in Table 3. Such a tableau algorithm is sound, complete and terminating.*

2.2 Planning

This section summarizes the background of planning that is needed to generalize and formalize our specific problems. First, the representation scheme and subcategories of the classical planning are presented in Section 2.2.1. Then, we introduce the ideas of state space and plan space planning (Section 2.2.2). At the end of this section, a brief investigation on the complexity of planning is offered to give readers an impression how hard planning itself could be (Section 2.2.3). Most of the notation and definitions involved in this section are attributed to [Ghallab et al., 2004], though some particular references are given as well.

Planning is an abstract process of organizing actions by anticipating some expected outcomes. Though there exists various forms of planning because of different types of actions, planning is generally concerned with choosing actions for changing the states of a system. Hence, a model called *state transition system* (Definition 2.2.1), a terminology commonly used in theoretical computer science, is used to describe most planning approaches.

Definition 2.2.1. *A state transition system is a 3-tuple* $\Sigma = (S, A, \gamma)$*, where:*

- $S = \{s_1, \ldots\}$ *is a finite set of states;*
- $A = \{a_1, \ldots\}$ *is a finite set of actions;*
- γ*:* $S \times A \rightarrow 2^S$ *is a state transition function.*

Given the above Σ, the purpose of planning is to find which actions to apply to which states toward an objective. Note that a state transition system can be represented by a directed graph whose nodes are states and arcs are state transitions.

2.2.1 Classical Planning

The conceptual model, i.e., state transition systems, is usually subject to some restrictive assumptions to be operational. Particularly, most present AI planners adapt the ideas from [Fikes and Nilsson, 1971], resulting in *restricted state transition systems.* A restrictive state transition system requires the planning domain to be deterministic, static, finite and fully observable [see Ghallab et al., 2004, Chapter 1]. *Classical planning*, or STRIPS planning [Fikes and Nilsson, 1971], then refers to planning for restricted state transition systems.

Offline Planning is one of the restrictive assumptions applied to classical planning as opposed to *online planning.* Offline planning ignores any change that may occur in the state transition system Σ during the planning. In other words, offline planning assumes that the knowledge about the planning problem and environment is *complete*, so it plans for the given states regardless of any current dynamics. On the contrary, online planning has to consider how the system will evolve, which can not be handled directly within a restricted model.

Classical representation

Representation of an input problem for classical planning is necessary to compute states and state transitions easily. Among possibly many ways of representing an input, the *classical representation* is chosen in the following descriptions. This is reasonable since different representation methods are of the same expressive power. In the subsequent sections, the classical representation scheme is defined via a (first order) language $\mathcal{L}$.

We stipulate that there are finitely many predicate symbols, constant symbols and *no* function symbols in $\mathcal{L}$. As defined in FOL, every *term* appearing in $\mathcal{L}$ is either a variable symbol or a constant

symbol. An *atom*, or an atomic formula, is a formula that contains no logical connectives, thus it is the simplest well formed formula of $\mathcal{L}$. A *ground term* contains no variables at all. Atoms and negated atoms are called positive and negative *literals* respectively.

States and Operators In the classical representation, a state s is a set of ground atoms. Note that atoms may have different truth values from state to state. An atom p holds in s iff $p \in s$. If g is a set of literals, s satisfies g when there is a substitution σ s.t. every positive literal of $\sigma(g)$ is in s and no negative literal of $\sigma(g)$ is in s. It is easy to see that the set of all possible states are finite due to the absence of function symbols in $\mathcal{L}$.

In the representation of states, it is assumed that the *closed-world assumption* is to be used. In other words, literals, unless explicitly expressed positively, are considered negative. For states, negative literals will be disallowed by strictly applying this assumption.

Operators in the classical representation are defined as follows.

Definition 2.2.2. *A planning* operator *can be represented as a triple* $\langle$*Name, Prec, Effc*$\rangle$*, where:*

- *Name is the operator name expressed as* $n(x_1, \ldots, x_k)$*, where* n *is a unique operator symbol,* $x_1, \ldots, x_k$ *are all the variable symbols occurring in the operator;*
- *Prec and Effc are sets of literals used to generalize the preconditions and effects, respectively.*

An *action* is just a ground instance of the planning operator. If the conditions of some state s are met in an action a, then we say a is applicable to s, and the result of the application is another state.

Planning Problems and Solutions The planning domain is independent of any particular goal or initial state, while the planning problems will include a domain, an initial state and a goal. First, the definition of a planning domain is given as follows.

Definition 2.2.3. *A classical planning domain in* $\mathcal{L}$ *is a restricted state transition system* $\Sigma = (S, A, \gamma)$*, where:*

- $S \subseteq 2^{\{all\ ground\ atoms\ of\ \mathcal{L}\}}$;
- A=$\{$*all ground instances of operators*$\}$;
- γ *is the state transition function which leads to a new state* $\gamma(s, a)$ *if* $a \in A$ *is applicable to* $s \in S$*. Note that* S *is closed under* γ.

A classical planning problem can thus be defined based on the classical planning domain.

Definition 2.2.4. *A classical planning problem is described as* $P = (\Sigma, s_0, g)$*, where:*

- Σ *is the classical planning domain;*
- s_0 *is the initial state, which could be any state in* S*;*
- g*, the goal, is a set of ground literals.*

When a state satisfies g, this state becomes a *goal state*. There could exist more than one goal state.

Now it comes to define a plan and a solution. A *plan* is a sequence of actions $\pi = \langle a_1, \ldots, a_i \rangle$, where $i \geq 0$, $a_i \in A$. The length of a plan is the number of actions. A plan π is a *solution* for a planning problem P if a goal state can be reached from the initial state s_0 by applying actions along π.

Up till now, the classical representation has been introduced. In practice, planning domains and problem descriptions are standardized by some description languages. One of the popular languages is PDDL [Gerevini and Long, 2005], developed for planning competitions[2], which contains STRIPS formalisms and more, so that classical planning problems can be stated in PDDL. In fact, most planners do not support full PDDL but only its STRIPS subset.

2.2.2 State Space and Plan Space Planning

Classical planning problems can be considered as the search for some path in the graph representing a state transition system Σ. State space search and plan space search are two categories of classical planning algorithms, as elaborated in this section.

State Space Planning These algorithms are the simplest classical planning algorithms, where the search space is a subset of the state space. Nodes of the search space are states of the domain; arcs are state transitions (or actions); a plan is a sequence of actions corresponding to a path from the initial state to a goal state.

State space planning is commonly (but not necessarily) associated with *total order* planning [Minton et al., 1994], such as STRIPS. A plan, considered as an ordered set of actions, is totally ordered if every action is ordered w.r.t. every other action. For state space planning, there are two main approaches as search: *progression* and *regression*.

[2]More information available at http://ipc.icaps-conference.org/

Progression, or forward search, is the simplest planning algorithm, which can be any search methods, such as breadth-first search (BFS), depth-first search (DFS) and A*, following these ideas: (1) starting from the initial state, compute whether or not a state is a goal state, (2) find the set of all actions applicable to a state, and (3) compute a successor state which is the result of applying an action to a state. Conversely, regression (or backward search) starts at the goal and applies the inverse[3] of the operators to produce subgoals, stopping if a set of subgoals satisfied by the initial state are produced. Both progression and regression algorithms are sound and complete.

Plan Space Planning These algorithms search in a space where nodes are *partial-order plans*. Arcs are plan refinement operations to further complete a partial plan. Intuitively, plan space planning is generally associated with partially ordered planning in the sense that a refinement operation does not add to the partial plan any constraints that are not strictly necessary. This principle is known as *the principle of least commitment* (Definition 2.2.5), the key difference between plan space and state space planning.

The principle of least commitment, also named *principle of procrastination*, was embraced early in planning research, stating that a planner should always avoid making decisions unless required to do so. In particular, partial-order planners have the objective to avoid unnecessary ordering commitments. We rephrased this principle, stated by Marr [1976], as follows.

Definition 2.2.5. *The Principle of Least Commitment states that one should never become prematurely committed to something that subsequent discoveries may force one to undo.*

As opposed to least commitment planning, state space planners make commitments about the order of action as they try to find a solution and therefore may make mistakes from poor guesses about the right order of actions.

Plan space planning also differs from state space planning in the definition of a solution plan. A plan is defined as a set of planning operators together with ordering constraints and binding constraints [see Ghallab et al., 2004, Chapter 5], which may not correspond to a sequence of actions. Partial order planning algorithms are also sound and complete, and POP and UCPOP algorithms can be considered major contributions to plan space planning.

Remarks It has been controversial whether plan space planning is more efficient than state space planning. Although plan space planning, for a while, outperformed state space planning, the former

[3]To apply the inverse of an operator means to choose an operator that will satisfy one of the goals, then replace that goal with the operator's preconditions.

ignores the notion of explicit states along the plan. As a result, state space planner are able to make use of domain-specific heuristics to scale up to very large problems. The framework presented is not affected by the choice between these two approaches, but really depends on concrete implementations.

2.2.3 Complexity of Planning

The computational complexity regarding planning is one of the main research topics. One should note that it is the expressive power of the representation schemes that matters. That is, planning algorithms are independent of the representation schemes themselves. This section adapts studied results on complexity of planning. First, let us simplify a planning problem into a decision problem:

> PLAN-EXISTENCE(P) indicates whether P represents a solvable planning problem. When it is obvious what P is, it is omitted.

It is established [see Ghallab et al., 2004, Chapter 3] that PLAN-EXISTENCE in ordinary classical planning remains decidable, but extending classical planning to allow function symbols makes PLAN-EXISTENCE *semidecidable*, i.e., a procedure can be found to terminate and return **yes** if P is solvable, but never return **yes** if P is unsolvable. Table 4 summarizes the computational complexity of classical planning problems in the classical representation scheme, depending on different conditions.

How operators are given	Negative effects?	Negative preconditions?	Complexity
In the input	Yes	Yes/No	EXPSPACE-Complete
	No	Yes	NEXPSPACE-Complete
		No	EXPSPACE-Complete
		No*	PSPACE-Complete
In advance	Yes	Yes/No	PSPACE
	No	Yes	NP
		No	P
		No*	NLOGSPACE

* No operator has more than one precondition.

Table 4: Complexity of PLAN-EXISTENCE for Classical Planning [Ghallab et al., 2004, Chapter 3].

The results we have shown state that the worst-case complexity of classical planning is rather high, even if restrictions are made. However, the results do not necessarily depict the complexity of any particular planning domain since they are only worst-case results.

Chapter 3

Lazy Unfolding and Axiom Absorption

In this chapter, we will survey a number of related works on absorption techniques, which are optimizations on handling axioms. Section 3.1 first states why axioms need special care in DL reasoning, followed by existing absorptions covered in Section 3.2. Finally, Section 3.3 shows a map of various absorptions, and discusses a general form of axiom absorption in the hypertableaux reasoning.

The definitions and notations shown in the following paragraph will be used throughout the thesis unless otherwise specified. Special attention should be paid to the division of a TBox of a DL KB, which will be frequently referred to in this thesis.

Definitions and Notations The set of concept names is denoted by **AC** including $\top$ and $\bot$, and the set of negated concept names by **NA**. A concept is called a *simple concept* if it is a member of either **AC** or **NA**. We denote by A, B and A_i ($1 \leq i \leq n$ for some integer n) concept names in **AC**, by C, D and C_i ($1 \leq i \leq n$ for some integer n) arbitrary concept expressions, and by R, S some roles and R^-, S^- the inverse relations of R, S, respectively. $\mathrm{Sig}(\mathcal{T})$ (or $\mathrm{Sig}(C)$) is the signature, i.e., the set of concept and role names, of some TBox $\mathcal{T}$ (or concept expression C).

A Tbox $\mathcal{T}$ is divided such that $\mathcal{T} \equiv \mathcal{T}_u^A \cup \mathcal{T}_u^{\neg A} \cup \mathcal{T}_u^{\sqcap} \cup \mathcal{T}_e \cup \mathcal{T}_g$, where $\mathcal{T}_u^A$ and $\mathcal{T}_u^{\neg A}$ contain axioms of the forms $A \sqsubseteq C$ and $\neg A \sqsubseteq C$ respectively (Section 3.2.1), $\mathcal{T}_u^{\sqcap}$ consists of axioms in the form of $A_1 \sqcap \ldots \sqcap A_i \sqsubseteq C$, where $i \geq 2$ (Proposition 4.1.1), $\mathcal{T}_e$ is composed of equivalence axioms of the form

$A \equiv C$ (Section 3.2.7), and $\mathcal{T}_g$ consists of all other axioms that are normally called general axioms. In contrast to $\mathcal{T}_g$, all TBoxes $\mathcal{T}_u^A$, $\mathcal{T}_u^{\neg A}$, $\mathcal{T}_u^{\sqcap}$ and $\mathcal{T}_e$ are unfoldable, i.e., each of them is a subset of $\mathcal{T}_u$. The *unfoldable* TBox $\mathcal{T}_u$ and the *general* TBox $\mathcal{T}_g$ are such that $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$ and $\mathcal{T}_u \cap \mathcal{T}_g = \emptyset$. Moreover, it is assumed that every axiom belongs to $\mathcal{T}_g$ before the input KB is preprocessed. At this stage, every axioms can be considered for absorptions. General axioms become unfoldable when they have been successfully absorbed into some $\mathcal{T}_u$.

3.1 Lazy Unfolding

As Tsarkov et al. [2007] argued, TBox axioms, if dealt with in a cursory way, i.e., by applying the $\sqsubseteq$-rule as shown in Figure 2, can lead to a catastrophic degradation in reasoning performance.

> **$\sqsubseteq$-rule:**
> if $(C_1 \sqsubseteq C_2) \in \mathcal{T}$,
> and $\{\neg C_1, C_2\} \cap \mathcal{L}(\mathrm{x}) = \emptyset$,
> then $\mathcal{L}(\mathrm{x}) = \mathcal{L}(\mathrm{x}) \cup \{C\}$ for some $C \in \{\neg C_1, C_2\}$

Figure 2: Tableau Expansion Rules for Axioms

For example, axiom $A \sqsubseteq C$ will be first transformed to $\top \sqsubseteq \neg A \sqcup C$, then concept $\neg A \sqcup C$ is added to every node of the completion graph. It is easy to see that each such axiom would enforce an addition of a disjunction, which results in an enormous non-deterministic expansion of the search space. Conversely, there might exist a solution such that all axioms can be *lazily unfolded* during the tableau expansion without resorting to the $\sqsubseteq$-rule [Baader et al., 1994].

Definition 3.1.1. *Lazy unfolding is used to deal with axioms of the form $A \sqsubseteq C$ or $A \equiv C$ in a TBox $\mathcal{T}$ such that:*

Case 1 *if $A \in \mathcal{L}(x)$, $(A \equiv C)$ or $(A \sqsubseteq C) \in \mathcal{T}$ and $C \notin \mathcal{L}(x)$, then $\mathcal{L}(x) = \mathcal{L}(x) \cup \{C\}$; Or*

Case 2 *if $\neg A \in \mathcal{L}(x)$, $(A \equiv C) \in \mathcal{T}$ and $\neg C \notin \mathcal{L}(x)$, then $\mathcal{L}(x) = \mathcal{L}(x) \cup \{\neg C\}$.*

Instead of adding unnecessary disjunctions caused by axioms, lazy unfolding is capable to handle axioms in a deterministic manner such that the possible exponential explosion of the search space is prevented. Lazy unfolding, in its essence, is a typical application of *the principle of least commitment* (see Definition 2.2.5).

It is the appropriate time to introduce lazy unfolding for axioms in different TBoxes. $\mathcal{T}_u$ consists of axioms that can be unfolded, while $\mathcal{T}_g$ contains axioms which lazy unfolding is not applicable to. With the help of these notions, lazy unfolding, formulated in Figure 3 as two tableau expansion rules for $\mathcal{T}_u$ and $\mathcal{T}_g$ respectively, becomes a proper substitute for the $\sqsubseteq$-rule in Figure 2 [Horrocks, 2003].

$\sqsubseteq_u$**-rule:**	$\sqsubseteq_g$**-rule:**
if $C_1 \in \mathcal{L}(\mathrm{x})$, $(C_1 \sqsubseteq C_2) \in \mathcal{T}_u$,	if $(C_1 \sqsubseteq C_2) \in \mathcal{T}_g$,
and $C_2 \cap \mathcal{L}(\mathrm{x}) = \emptyset$,	and $\{\neg C_1, C_2\} \cap \mathcal{L}(\mathrm{x}) = \emptyset$,
then $\mathcal{L}(\mathrm{x}) = \mathcal{L}(\mathrm{x}) \cup \{C_2\}$	then $\mathcal{L}(\mathrm{x}) = \mathcal{L}(\mathrm{x}) \cup \{C\}$ for some $C \in \{\neg C_1, C_2\}$

Figure 3: Tableau Expansion Rules for Axioms in $\mathcal{T}_u$ and $\mathcal{T}_g$

Note that Figure 3 does not show the case for equivalence axioms of the form $A \equiv C$ as discussed in Definition 3.1.1. The reason is that an axiom of the form $A \equiv C$ is just an abbreviation of two inclusion axioms $A \sqsubseteq C$ and $C \sqsubseteq A$, which the tableau rules in Figure 3 are applicable to.

3.2 Absorption Techniques

Since lazy unfolding, in contrast to the application of the $\sqsubseteq$-rule, is quite effective in practice, naturally we expect that as many axioms as possible can be lazily unfolded during tableau expansions, i.e., the size of $\mathcal{T}_g$ can be shrunk minimally. To facilitate lazy unfolding, researchers later implemented an optimization technique, *absorption*, which *rewrites* axioms in order to fit them to the definition of some $\mathcal{T}_u$. A way toward defining the absorption technique is provided in Definition 3.2.1.

Definition 3.2.1. *Absorption is a rewriting technique that transforms any axiom $C \sqsubseteq D$ to a suitable form for any unfoldable TBox $\mathcal{T}_u$ while the following two conditions hold:*

- *the semantics of the axiom is preserved after transformation;*
- *$\mathcal{T}_u$ remains unfoldable when the absorbed axiom is added.*

Roughly speaking, absorptions aim to relocate general axioms from $\mathcal{T}_g$ to some $\mathcal{T}_u$. For some axiom, there may exist more than one way to absorb it to some $\mathcal{T}_u$ depending on how the $\mathcal{T}_u$ is defined. More precisely, the LHS of axioms defined in $\mathcal{T}_u$ determines what kind of absorptions

should be employed for axioms. Corresponding to the various forms of LHS admissible in $\mathcal{T}_u$ is a list of different absorptions, as detailed in subsequent sections.

The advantages of absorptions can be viewed in several ways. Practically, reasoning systems that employ absorptions outperforms those that do not because the search space of the completion graph is significantly reduced. Considering a completion graph containing n nodes and a KB with k general axioms, a total of $(k \times n)$ disjunctions will be added and expanded in $2^{k \times n}$ ways non-deterministically if the $\sqsubseteq$-rule is naively used rather than absorptions.

Although a number of absorptions have been presented in the literature [Horrocks and Tobies, 2000; Haarslev and Möller, 2001a; Tsarkov and Horrocks, 2004; Hudek and Weddell, 2006; Zuo and Haarslev, 2006; Ding et al., 2007; Sirin et al., 2006], they come into effect insofar as the general axioms are of specified uniform forms and/or of a moderate quantity. Ipso facto, some absorptions enjoy prevalence among almost all known ontologies, but their effectiveness might deteriorate on some emerging large and complicated ontologies, for example, some axioms in BCS5 cannot be absorbed by present absorptions. It then becomes natural to seek a good absorption among all the possibilities. Yet, there is even no exact description of what a good absorption ought to be. A *rule of thumb* to guide absorption behaviors, as Horrocks and Tobies [2000] observed, is to leave $\mathcal{T}_g$ empty or as small as possible. This coincides with the discussion in Section 3.1 that lazy unfolding should be exploited in a TBox for better reasoning performance.

3.2.1 Concept Absorption (CA)

Concept absorption is formally presented in [Horrocks and Tobies, 2000; Horrocks, 2003] to absorb a general axiom into another one whose LHS is either a concept name A or a negated concept name $\neg A$. Horrocks [2003] also provided a restrictive but effective procedure to apply absorptions. The transformation rule w.r.t. concept absorption is summarized below.

Proposition 3.2.1. *Concept absorption is based on the following transformation rule:*

$$\begin{aligned} C_1 \sqcap C_2 \sqsubseteq D &\Leftrightarrow C_1 \sqsubseteq \neg C_2 \sqcup D \\ C \equiv D &\Leftrightarrow C \sqsubseteq D \text{ and } \neg C \sqsubseteq \neg D \end{aligned} \tag{4}$$

An observation is that the transformed axiom should have its LHS in conformity with the definition of some $\mathcal{T}_u$. In this case, absorptions can absorb axioms into a $\mathcal{T}_u$ with its LHS being concept names. Some systems, such as RacerPro [cf. Haarslev and Möller, 2001a], have been extended to

support negated concept names as well. To ease our presentation, the concept absorption that absorbs into concept names only are denoted by **CAP**, while the one that absorbs into negated concept names are denoted by **CAN**. The case that both A and $\neg A$ are defined in the same TBox is valid only when these two axioms could be merged into a definition of the form $A \equiv C$ (i.e., an equivalence axiom).

3.2.2 Enhanced Concept Absorption (ECA)

An immediate question arising from the concept absorption in Section 3.2.1 is whether absorptions can have both A and $\neg A$ be defined in the same TBox even if they can not form the axiom $A \equiv C$. Zuo and Haarslev [2006] generalized this idea by constructing new axioms to ensure the correctness of absorptions. The introduction of fresh axioms, at first sight, seems ineffectual. According to the results in [Zuo and Haarslev, 2006], however, this absorption surpasses the others in certain cases. As a complementary strategy to concept absorption, the main idea of the so-called enhanced concept absorption has been recapped as follows.

Proposition 3.2.2. *Enhanced concept absorption can be applied in the following situations:*

1. *If an axiom $\neg A \sqsubseteq C$ already exists in $\mathcal{T}_u$, then the absorption that absorbs some axiom as an axiom of the form $A \sqsubseteq D$ (into $\mathcal{T}_u$) is correct if another axiom $\top \sqsubseteq C \sqcup D$ is added to $\mathcal{T}_g$.*

2. *If axioms $A \sqsubseteq C_1$ and $\neg A \sqsubseteq C_2$ already exist in $\mathcal{T}_u$, then the absorption that absorbs some axiom as an axiom of the form $A \sqsubseteq D$ (into $\mathcal{T}_u$) is correct if another axiom $\top \sqsubseteq C_2 \sqcup D$ is added to $\mathcal{T}_g$.*

A procedure for this absorption is illustrated in [Zuo, 2006] where potential defects of the procedure are exhibited as well. One imperfection of this idea is obviously that newly introduced axioms may be even more difficult for absorptions than the original ones. If so, the effect of this absorption is then detrimental. Another consequence of this procedure is that non-termination will probably occur if not all the new axioms can be completely absorbed. After all, this absorption is of potential use if there are remedies for its drawbacks.

Another contribution of Zuo [2006] is to improve heuristic choices during absorptions. As shown from the beginning, absorption procedures are themselves non-deterministic. Hence, the heuristic is applied to the entire TBox instead of a single axiom to decrease the non-determinism of absorptions. Through the use of the occurrence-based counting of (negated) concept names, the absorption tends

to choose the most frequently occurring (negated) concept name in order to maximize the number of absorptions.

3.2.3 Inverse Role Based Absorption (IRBA)

The expressiveness of *inverse roles* ($\mathcal{I}$) consists in the capability of propagating constraints to the predecessors of nodes of the completion graph. It is the propagation that yields a dual representation of exactly the same information. For DLs in the presence of inverse roles, Ding and Haarslev [2005] briefly mentioned an elegant equivalence, which turns out to be quite useful in practice.

Proposition 3.2.3. *For DLs having $\mathcal{I}$, the equivalence*

$$C \sqsubseteq \forall R.D \Leftrightarrow \neg D \sqsubseteq \forall R^{-}.\neg C \tag{5}$$

holds for some role R and its inverse R^{-}.

Hudek and Weddell [2006] made the first attempt to use a variant of this equivalence in their absorption framework. Two features of this variation as an absorption catch the attention. First, fresh concept names are used to supplement binary absorptions (Section 3.2.5). Second, new axioms are also introduced to deal with back-propagation information. They are explained in Figure 4 showing how inverse roles are used in absorptions.

$$\top \sqsubseteq \neg A_1 \sqcup \neg A_2 \sqcup \neg B_1 \sqcup \underline{\forall R.\forall S.\neg B_2} \sqcup \exists R.C$$

$$\rightsquigarrow \begin{cases} \top \sqsubseteq \neg A_1 \sqcup \neg A_2 \sqcup \neg B_1 \sqcup \neg A' \sqcup \exists R.C & \textit{Introduce a fresh concept } A' \in \boldsymbol{AC} \\ \top \sqsubseteq \underline{\forall S.\neg B_2} \sqcup \forall R^{-}.A' & \textit{Introduce a new axiom} \end{cases}$$

$$\rightsquigarrow \begin{cases} A' \sqsubseteq \neg A_1 \sqcup \neg A_2 \sqcup \neg B_1 \sqcup \exists R.C & \textit{Safely absorb to } A' \textit{ by concept absorption} \\ \top \sqsubseteq \neg A'' \sqcup \forall R^{-}.A' & \textit{Introduce a fresh concept } A'' \in \boldsymbol{AC} \\ \top \sqsubseteq \neg B_2 \sqcup \forall S^{-}.A'' & \textit{Introduce a new axiom} \end{cases}$$

$$\rightsquigarrow \begin{cases} A' \sqsubseteq \neg A_1 \sqcup \neg A_2 \sqcup \neg B_1 \sqcup \exists R.C & \textit{Safely absorb to } A' \textit{ by concept absorptions} \\ A'' \sqsubseteq \forall R^{-}.A' & \textit{Safely absorb to } A'' \textit{ by concept absorptions} \\ B_2 \sqsubseteq \forall S^{-}.A'' & \textbf{\textit{Axiom 1}} \end{cases}$$

Figure 4: Usage of Inverse Roles in Absorptions [Hudek and Weddell, 2006].

We add a few comments about Figure 4 to present the basic idea. For every universal restriction of the form $\forall R.C$, where R is some role and C is an arbitrary concept expression, a fresh concept

name A (corresponds to A' and A'' in Figure 4) will be introduced such that its negation replaces the universal restriction in the original general axiom, and an additional general axiom of the form $\top \sqsubseteq C \sqcup \forall R^{-}.A'$ will be generated. If C is still a universal restriction (like $\forall S.\neg B_2$), then it is subject to the same processing as well.

Similarly, Ding et al. [2007] generalized the usage of equivalence (5) to map the DL $\mathcal{ALCI}$ to $\mathcal{ALC}$ by introducing new named concepts. Another absorption could be derived from that paper by using inverse roles, as described in Figure 5.

$$\top \sqsubseteq \neg A_1 \sqcup \neg A_2 \sqcup \neg B_1 \sqcup \underline{\forall R.\forall S.\neg B_2} \sqcup \exists R.C$$
$$\rightsquigarrow \quad B_2 \sqsubseteq \forall S^{-}.\forall R^{-}.(A_1 \sqcup \neg A_2 \sqcup \neg B_1 \sqcup \exists R.C) \quad \textbf{\textit{Axiom 2}}$$

Figure 5: Usage of Inverse Roles in Absorptions [Ding et al., 2007].

Through a comparison between Figure 4 and Figure 5 it is easy to see that on the one hand, *Axiom* 1 and *Axiom* 2 can *possibly* be achieved by concept absorption. If both of them need to be safely absorbed, concept name B_2 must comply with the conditions for concept absorptions. On the other hand, *Axiom* 2 in Figure 5 is essentially the same as *Axiom* 1 in Figure 4, because it can be obtained from *Axiom* 1 by unfolding concept names (i.e., A' and A'') within the TBox in Figure 4.

3.2.4 Domain and Range (Role) Absorptions (DRA)

The domain and range are important properties of roles in DLs. Haarslev and Möller [2001a] first presented that the domain and range of some role R can be determined by axioms, as shown in (6).

Proposition 3.2.4. *Let C and D be some concepts, R some role, then:*

$$\begin{aligned} &\textit{if } C \textit{ is the domain of } R, \textit{ then } \exists R.\top \sqsubseteq C \\ &\textit{if } D \textit{ is the range of } R, \textit{ then } \top \sqsubseteq \forall R.D \end{aligned} \tag{6}$$

Note that such general axioms are not amenable to concept absorptions, thus Haarslev and Möller [2001a] used a special treatment to deal with domain restrictions by adding C to the domain of R. Every existential restriction of the form $\exists R.C_1$ adjoins the domain of R, C, as a conjunction $C \sqcap \exists R.C_1$ during unfolding. Despite the fact that a range restriction adds no disjunctions, it was pointed out that the range restriction of R could also be resolved by associating with every existential

restriction, $\exists R.C_1$, the range restriction to form a conjunction $\forall R.D \sqcap \exists R.C_1$. Afterward, Tsarkov and Horrocks [2004] formalized and extended this idea to the *role absorption*, as depicted in (7).

Proposition 3.2.5. *Extended role absorptions can rewrite axioms as follows:*

$$\begin{aligned} &\exists R.C \sqsubseteq D \text{ is absorbed into } D \sqcup \forall R.\neg C, \text{ added to the domain of } R \\ &D \sqsubseteq \forall R.C \text{ is absorbed into } \neg D \sqcup \forall R.C, \text{ added to the domain of } R \end{aligned} \tag{7}$$

We can observe from (7) that role absorption does not *completely* absorb axioms in the sense that nondeterminism is merely localized in the domain of roles.

3.2.5 Binary Absorption (BA)

Hudek and Weddell [2006] proposed a framework of absorptions with a new absorption intended for axioms of the form $A \sqcap B \sqsubseteq C$. This absorption, called binary absorption, apparently cuts down the number of disjunctions in general axioms. It is proved that new internal named concepts introduced in the course of binary absorptions can be utilized to further absorb axioms. We take the same example from Section 3.2.3 to show how binary absorption works with the aid of inverse role based absorption (see Figure 6).

$\top \sqsubseteq \underline{\neg A_1 \sqcup \neg A_2} \sqcup \neg B_1 \sqcup \forall R.\forall S.\neg B_2 \sqcup \exists R.C$

$\rightsquigarrow \begin{cases} A_1 \sqcap A_2 \sqsubseteq A' & \textit{Introduce a fresh concept } A' \in \boldsymbol{AC} \\ \top \sqsubseteq \underline{\neg A' \sqcup \neg B_1} \sqcup \forall R.\forall S.\neg B_2 \sqcup \exists R.C & \textit{The axiom changes} \end{cases}$

$\rightsquigarrow \begin{cases} A_1 \sqcap A_2 \sqsubseteq A' & \\ A' \sqcap B_1 \sqsubseteq A'' & \textit{Introduce a fresh concept } A'' \in \boldsymbol{AC} \\ \top \sqsubseteq \neg A'' \sqcup \underline{\forall R.\forall S.\neg B_2} \sqcup \exists R.C & \textit{The axiom changes} \end{cases}$

$\rightsquigarrow \begin{cases} A_1 \sqcap A_2 \sqsubseteq A' & \\ A' \sqcap B_1 \sqsubseteq A'' & \\ \top \sqsubseteq \underline{\neg A'' \sqcup \neg A'''} \sqcup \exists R.C & \textit{Introduce a fresh concept } A''' \in \boldsymbol{AC} \\ \top \sqsubseteq \forall S.\neg B_2 \sqcup \forall R^-.A''' & \textit{Introduce a new axiom, as shown in Figure 4} \end{cases}$

..... *Repetition of steps shown above and in Figure 4*

Figure 6: Binary Absorptions with Inverse Role Based Absorptions. We assume that A_1, A_2, and B_1 have no equivalence definitions in the TBox, i.e., they are real candidates for binary absorptions.

The basic idea of binary absorption is that whenever a disjunction of two negated concept names

such as $\neg A \sqcup \neg B$ is found on the RHS of some general axiom, binary absorption will be considered[1]. This pair of negations form a *binary axiom* of the form $A \sqcap B \sqsubseteq A'$, where A' is a fresh concept name introduced by the absorption. At the same time, the negation of the fresh concept name, i.e., $\neg A'$, will replace the appropriate part (the disjunction of the pair) of the original axiom, as can be seen in Figure 6.

Figure 6 does not show the full picture of the process of binary absorptions, but it is clear that inverse role based absorptions supplement binary absorptions by introducing unique fresh concept names, which can be safely used by all absorptions. From this perspective, inverse role based absorption described in Figure 5 may not be preferred due to its inability to support binary absorptions.

Binary absorptions impose a new requirement for lazy unfolding: DL reasoners need to deal with axioms of the form $A \sqcap B \sqsubseteq C$ in a deterministic fashion. In other words, lazy unfolding will be applicable not only to axioms with (negated) concept names on the LHS, but also to axioms having a conjunction of two concept names. Technically, the $\sqsubseteq_u$-rule in Figure 3 allows C_1 to be concepts of the form $A_1 \sqcap A_2$, where A_1 and A_2 are concept names. Additionally, another question about how to efficiently utilize this kind of lazy unfolding is posed. For instance, DL systems must single out all the pairs of concept names that can be unfolded from a pool of concept names. So far, existing DL reasoners have not yet widely taken advantage of binary absorptions.

3.2.6 Nominal Absorption (OA)

Nominals, also named *enumerated classes*, form the important concept constructor $\mathcal{O}$ for DLs. Sirin et al. [2006] presented a suite of optimization techniques for nominals, including nominal absorptions. Nominal absorption, as the name implies, tries to resolve the non-determinism from general axioms containing nominals. Two transformation rules are described below.

Proposition 3.2.6. *Nominal absorption states that*

$$\begin{aligned} &C \equiv \{a_1, \ldots a_n\} \textit{ is logically equivalent to} \\ &C \sqsubseteq \{a_1, \ldots a_n\} \textit{ and } C(a_1), \ldots, C(a_n) \end{aligned} \tag{8}$$

$$\exists R.\{o\} \sqsubseteq C \textit{ is logically equivalent to } \{o\} \sqsubseteq \forall R^{-}.C \tag{9}$$

Since nominals can be viewed as concepts sui generis, the above absorptions resemble typical

[1]Restrictions on such negated concept names apply, as reported in [Hudek and Weddell, 2006].

absorptions for some ordinary concepts C. For example, (8) acts the same way as how concept absorptions treat equivalence axioms, while (9) is an application of inverse role based absorption.

3.2.7 Equivalence Absorption (EA)

It can be shown that concept names, when chosen as the LHS of some absorbed axioms, should not have an equivalence definition in order to maintain the unfoldability of the TBox. Given this requirement, the treatment for equivalence axiom turns out to be quite important. An immediate solution to equivalence definitions could be to leave them intact in $\mathcal{T}_e$. The number of equivalence definitions in a TBox is then inversely proportional to the number of concepts that could be used in absorptions. Hence, it is better if the size of $\mathcal{T}_e$ can be reduced.

To remedy the deficiency caused by equivalence definitions, the preprocessing can revert them into ordinary subsumption axioms as shown in (4), i.e., $C \equiv D$ is of the same effect as $C \sqsubseteq D$ and $\neg C \sqsubseteq \neg D$. Being traded for two general axioms, an equivalence axiom probably brings in a higher degree of nondeterminism. Nevertheless, an additional "absorption" can always be used to cope with these general axiom. The strategy is to piece together axioms to form as much new equivalence definitions as possible.

We abuse the term *absorption* for dealing with equivalence axioms, which refers to this special case of the enhanced concept absorption: through the enhanced concept absorption the axioms $A \sqsubseteq C$ and $\neg A \sqsubseteq C$ can be absorbed into an equivalence $A \equiv C$ whereas the introduced (trivial) axiom $\top \sqsubseteq A \sqcup \neg A$ can be discarded. Equivalence absorption differs from general ECA in that the former generates no new axioms and thus complies with classical planning.

3.3 Remarks

We have discussed a number of absorptions developed for tableau-based DL reasoning, of which some are well known and extensively applied, meanwhile, others are restrictively employed in existing reasoners. In real applications, normalization and/or other syntactic transformations usually precede absorptions. Almost all the absorptions discussed earlier assume that the concepts in the axioms are in NNF, that is, negations occur only in front of concept names. We refer readers to [Horrocks, 2003; Tsarkov et al., 2007] for a full description of common normalizations and simplifications.

Another point is that axioms are usually to be re-structured to facilitate absorptions. For example, a general axiom is transformed in such a way that the LHS becomes $\top$ and the RHS is

decomposed into a set of basic units [see Horrocks, 2003]. Absorptions are, in principle, applied by examining individual elements of this set. To distinguish between a concept and the elements in such a set, the latter are called *items*. For instance, axiom $\exists R.C \sqsubseteq A$ is rewritten as $\top \sqsubseteq \forall R.\neg C \sqcup A$, where $\forall R.\neg C$, A are items in the set ($\forall R.\neg C$, A).

To adopt for a particular reasoner the most appropriate absorptions, as we argue here, one probably has to consider some properties of the knowledge base. We construct the map (Table 5) to summarize these absorptions.

Acronym	Target Items	Logic Constructs	LHS of Generated Axioms	Notes
CA	A or $\neg A$	Any	$\neg A$ or A	Absorb to A or $\neg A$ exclusively
ECA	A and $\neg A$	Any	$\top$	Termination not guaranteed; New general axioms introduced
IRBA	$\forall R.C$	$\mathcal{I}$	$\neg C$	
DRA	$\forall R.C$	Any	None	Disjunctions added
BA	$\neg A$ and $\neg B$	Any	$A \sqcap B$	A new tableau rule required
OA	$\forall R.\neg\{o\}$	$\mathcal{IO}$	$\{o\}$	
EA	A and $\neg A$	Any	A	Special case of **ECA**; No general axioms introduced

Table 5: Map of Known Absorptions. Target items refer to items that are to be selected for absorptions.

Note that this map uses the abbreviations to denote absorptions. Through the survey on currently available absorptions, it could be argued that absorptions may be dependent on some logical constructs (and thus certain logic), for example **IRBA** requires the concept constructor $\mathcal{I}$.

An important observation from Table 5 is that presently there exist no absorption techniques to target items in the form of existential restrictions ($\exists R.C$). We conjecture that no absorption can resolve existential restrictions (universal restrictions) on the RHS (LHS) of any axiom because they are inherently unabsorbable.

IRBA, DRA and OA Inverse role based absorption (**IRBA**) shows that the inverse roles play a key role in aiding the absorption of general axioms. Following the equivalence described in Section 3.2.3, the universal qualification can be directly used, if it is a concept name or a negated concept name, to avoid generating fresh concepts. Then, a universal qualification now plays the same role

as a (negated) named concept does. Consequently, the possibility of extracting qualifications in universal restrictions is the key for inverse role based absorptions.

Domain and range absorptions (**DRA**) displaces the disjunctions to the domain of roles rather than eliminate them. In (7), $\exists R.C \sqsubseteq D$ is essentially identical to the general axiom $\top \sqsubseteq \forall R.\neg C \sqcup D$, whose disjunctions will be moved to the domain of R instead. If such a general axiom is handled by inverse role based absorption, no disjunction is retained as long as the universal qualification is a (negated) concept name: $\neg D \sqsubseteq \forall R^{-}.\neg C$ for D a (negated) concept name. In addition, it is possible to recursively extract the universal qualifications, if necessary, until some (negated) concept name is found.

For nominal absorptions, the behaviors of equivalence (9) can also be imitated by concept absorptions and inverse role based absorptions if nominals are treated as ordinary concepts.

3.3.1 Hyperresolution and Absorption

Recently, Motik et al. [2007] developed for DL reasoning a *hypertableaux* calculus based on *hyperresolution*, which can be viewed as a hybrid of resolution and tableau. In their calculus, axioms in a DL TBox are first transformed to special Horn DL-clauses: universally quantified implications of the form $\bigwedge U_i \rightarrow \bigvee V_j$, where hyperresolution inferences derive some V_j only if all U_j are satisfied by some individuals.

During the translation into DL clauses, axioms are actually *absorbed* in a special way. For instance, an axiom $\exists R.A \sqsubseteq B$ is translated to a DL clause $R(x,y) \wedge A(y) \rightarrow B(x)$; an axiom $\exists R.\neg A \sqsubseteq B$ is translated to $R(x,y) \rightarrow A(y) \vee B(x)$; an axiom $\top \sqsubseteq \neg A_1 \sqcup \neg A_2 \sqcup \exists R.A_3$ is translated to $A_1(x) \wedge A_2(x) \rightarrow \exists R.A_3(x)$. Such translations in the hypertableaux calculus clearly resemble absorptions in a tableau calculus. The difference is that absorptions can allow negated concept names on the LHS. Furthermore, the translations in hypertableaux still cannot "absorb" existential restrictions on the RHS of some axioms. Since the discussion of hypertableaux and hyperresolution is outside the scope of this thesis, readers are referred to [Motik et al., 2007] for details.

Chapter 4

Extension of Absorptions

The last chapter (Chapter 3) provided a brief overview of present absorptions, particularly, binary absorption (Section 3.2.5) draws our attention. Although binary absorption seems to function well in DL reasoning, it can be extended to allow for other forms of absorptions. Additionally, absorptions have to conform to certain conditions in order to be properly used as operators in a classical planner, like *offline planning*, while binary absorptions violates offline planning by adding new concepts to the planning domain. This chapter first extends binary absorption to conjunctive absorption (Section 4.1), then it generalizes conjunctive absorptions to allow negated concept names on the LHS (Section 4.2).

4.1 Conjunctive Absorption

4.1.1 From Binary to Conjunctive Absorption

Recall that binary absorptions, motivated to utilize axioms of the form $A_1 \sqcap A_2 \sqsubseteq C$, where only concept names are allowed on the LHS, seem to be very straightforward. As we have pointed out, new named concepts will be needed to supplement the procedure of binary absorptions. An interesting observation is that one binary absorption cascades another (via the connection of newly introduced concepts) to deal with an axiom that demands more than one binary absorption, as illustrated in Figure 7a. Naturally, a one-step absorption could be formed in this case to avoid unnecessary internal named concepts. A theorem (Theorem 4.1.1) is then given to show an alternative solution to axioms that can be absorbed by binary absorptions.

Theorem 4.1.1. *An arbitrary concept D is satisfiable with respect to TBox $\mathcal{T}_1 = \mathcal{T} \cup \{A_1 \sqcap A_2 \sqcap A_3 \sqsubseteq C\}$ iff D is satisfiable with respect to TBox $\mathcal{T}_2 = \mathcal{T} \cup \{A_1 \sqcap A_2 \sqsubseteq B_1, B_1 \sqcap A_3 \sqsubseteq C\}$, where $B_1 \notin Sig(\mathcal{T}_1)$.*

Theorem 4.1.1, with its formal proof, were first presented in Hudek and Weddell [2006]. Thus, the proof is omitted here. Instead of cascading binary absorptions to deal with the axioms in Figure 7a, this theorem provides us with another absorption to resolve a qualified axiom without introducing new concept names, as described in Figure 7b. Consequently, binary absorption can be safely extended to the latter one that is named *conjunctive absorption.* A definition of conjunctive absorption follows below (Definition 4.1.1).

$A_1 \sqcap A_2 \sqsubseteq B_1$ $A_3 \sqcap B_1 \sqsubseteq B_2$ $B_2 \sqsubseteq \forall R.C$	$A_1 \sqcap A_2 \sqcap A_3 \sqsubseteq \forall R.C$
(a) Binary Absorption	(b) Conjunctive Absorption

Figure 7: Binary vs. Conjunctive Absorptions on axiom $(\neg A_1, \neg A_2, \neg A_3, \forall R.C)$. Note that B_1 and B_2 denote internal concept names.

Definition 4.1.1. *Let $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$, where $\mathcal{T}_u = \emptyset$ and $\mathcal{T}_g = \{\top \sqsubseteq \neg A_1 \sqcup \ldots \sqcup \neg A_i \sqcup C, i \geq 2\}$ for $A_k \in \boldsymbol{AC}$, $1 \leq k \leq i$; then $\mathcal{T}' = \mathcal{T}_u' \cup \mathcal{T}_g'$ where $\mathcal{T}_u' = \{A_1 \sqcap \ldots \sqcap A_i \sqsubseteq C, i \geq 2$ and $\exists j, 1 \leq j \leq i$ such that $\neg A_j$ never occurs on the LHS of $\mathcal{T}_u\}$ and $\mathcal{T}_g' = \emptyset$ is a correct absorption of $\mathcal{T}$. Such absorptions are named conjunctive absorptions.*

Notably, the extension does not generate new internal concept names, so, it conforms to the *offline* requirement during planning. This is especially desirable for a classical planner.

Since conjunctive absorption is directly derived from binary absorption, it makes sense to think that conjunctive absorption is a special case of binary absorption. A more general statement is made about the relationships among conjunctive absorption (**CJA**), binary absorption (**BA**) and concept absorption (restricted to allow concept names on the LHS, i.e., **CAP**), as described in Proposition 4.1.1.

Proposition 4.1.1. ***CJA*** *is a special case of* ***BA***, *and both of them are special cases of* ***CAP***. *The three absorptions* only *differ in the number of disjunctions on the RHS they can resolve in one absorption.*

Proposition 4.1.1 implies that **CJA** and **BA** can be derived from **CAP**. Intuitively, any general axiom that can be absorbed by conjunctive or binary absorptions (**CJA** or **BA**) is a candidate for

concept absorptions (**CAP**) as well. Hence, the extension of conjunctive absorption actually cannot guarantee to absorb more general axioms than the other two absorptions, but it enjoys the advantage of reducing the number of disjunctions in a more straightforward way.

Prior to this extension in tableau based DLs, some other works already presented a similar idea of conjunctive absorptions, although not necessarily for tableau based reasoning. First, Baader et al. [2006] presented one normal form for $\mathcal{EL}$ in a conjunctive format, i.e., $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B$, where $A_k, 1 \leq k \leq n$ and B are all concept names. Such a normal form has its origin in a binary normal form in Baader et al. [2005]. Further, as discussed in Section 3.3.1, Motik et al. [2007] generalized the idea of conjunctive absorption in hyperresolution by translating axioms into DL clauses of the form $\bigwedge U_i \rightarrow \bigvee V_j$, where U_i can be $A(x)$ or $R(x, y)$. However, none of the above generalization is aimed for tableau-based reasoning, thus the extension presented in this chapter becomes necessary for tableau-based DL reasoners.

4.1.2 Subset Testing during the Tableaux Expansion

Conjunctive absorptions, as well as binary absorptions, require a new unfolding rule. These two rules are denoted by *conjunctive and binary expansion rules* respectively. The only difference is that conjunctive rules enables C_1 to be concepts of the form $A_1 \sqcap \ldots \sqcap A_n$, where A_1 to A_n are concept names, as in the $\sqsubseteq_u$-rule in Figure 3.

For both binary and conjunctive absorptions, DL reasoning needs to take some special actions when some named concept is assigned to the label of one node during tableau expansions. The reason is that binary or conjunctive absorptions introduce a number of axioms (to ease our presentation, such axioms are called *binary axioms* and *conjunctive axioms*, respectively), which have several concept names on the LHS. Normally, a DL reasoner has to decide whether the set of named concepts currently belonging to the label of this node covers any set of named concepts occurring on the LHS of these axioms. Without loss of generality, the following example details the subset testing (or matching) in a tableau expansion.

Assuming that binary and conjunctive absorptions are respectively applied to an axiom in the set notation $(\neg A_1, \neg A_2, \neg A_3, \neg A_4, \neg A_5, C)$[1]. In this example, conjunctive absorptions produce one conjunctive axiom while binary absorptions produce 4 binary axioms.

We now consider the number of matches required for different nodes in tableau expansions. The scenario is described as follows. For some node in the tableau model, it initially has only A_1 in its

[1]Hence the axiom is of the form $\top \sqsubseteq \neg A_1 \sqcup \neg A_2 \sqcup \neg A_3 \sqcup \neg A_4 \sqcup \neg A_5 \sqcup C$.

label. Every time a new named concept is added to this node, a reasoner has to check if all named concepts in the label of this node cover any LHS of some binary or conjunctive axioms. Let us assume that every conjunctive axiom in the TBox has at least 3 and at most 5 named concepts on the LHS, that is, subset testing for conjunctive axioms will only be triggered if a node has 3 or more named concepts in the label. Table 6 records the number of subset tests for both binary axioms (produced by **BA**) and conjunctive axioms (produced by **CJA**).

Node Labels (new concepts are appended to the end of the labels)	**BA**	**CJA**
	# of subset testing/matching	
$\{A_1\}$	0	0
$\{A_1, A_2\}$	1	0
$\{A_1, A_2, A_3\}$	2	$C_2^2 = 1$
$\{A_1, A_2, A_3, A_4\}$	3	$C_3^3 + C_3^2 = 4$
$\{A_1, A_2, A_3, A_4, A_5\}$	4	$C_4^4 + C_4^3 + C_4^2 = 11$
TOTAL	10	16

Table 6: Naïve Matching Strategy for Binary and Conjunctive Axioms

Table 6 raises the question about the practicality of subset testing (or *matching*) in tableau-based DL reasoners. For binary absorptions, binary axioms uniformly have their LHS consist of two concept names. In every subset testing, the reasoner only needs to match every pair of the newly added named concept and an existing concept name in the label with the LHS of all binary axioms. Conjunctive axioms, in contrast to binary axioms, have different numbers of concept names on their LHS, for example, the above example presumes that conjunctive axioms have 3, 4 or 5 concept names on the LHS. In either case, this naïve subset testing algorithm hardly achieves satisfactory performance in real reasoners. A remedy could be obtained from Yellin [1992], which indexes binary (conjunctive) axioms through all involved concept names.

We continue by giving another practical algorithm (Figure 8) to deal with subset testing, which stems from the *watched literal* algorithm initially used in the SAT solver Chaff [Moskewicz et al., 2001].

The original algorithm is slightly modified in order to suit our needs. The basic idea is that every conjunctive axiom is *minimally* indexed in the sense that it is indexed by only one (possibly different) concept name occurring in it at all times. Consequently, a set of concept names indexes all

Input: Conjunctive axioms $\text{Axiom}_0 \ldots \text{Axiom}_m$, where concept names are elements in a set;
A queue of concept names **CN** in the label of node n;
Output: Conjunctive axiom(s) to be unfolded in node n;

```
1: {Initialize indexes of every axiom;}
2: for all i such that 0≤ i ≤m do
3:   A' := pickOneElement(set(Axiom_i));
4:   indexes(A') := indexes(A') ∪ {i};
5: end for
6: {Update indexes of axioms dynamically;}
7: while not empty(CN) do
8:   B ← pop(CN);
9:   CheckedConcepts := CheckedConcepts ∪ {B};
10:  while not empty(indexes(B)) do
11:    k ← fetch(indexes(B));
12:    hit ← true;
13:    for all concept names A in Axiom_k do
14:      if A ≠ B and A not in CheckedConcepts then
15:        indexes(B) := indexes(B) − {k};
16:        indexes(A) := indexes(A) ∪ {k};
17:        hit ← false;
18:      end if
19:    end for
20:    if hit then
21:      Results := Results ∪ {k};
22:    end if
23:  end while
24: end while
25: return Results;
```

Figure 8: Online Indexing Algorithm for Subset Testing

the axioms. During a tableau expansion, a new concept added to the label of some node needs not to be checked unless it indexes some axiom. Once a concept name indexing some axioms has been checked, these axioms will then be indexed by another concept name that has not yet occurred in the label of this node. When all concept names in some conjunctive axiom have been checked, this axiom should be triggered in this node, and thus the conjunctive rule applies. Axioms are indexed in such a circulating fashion that unnecessary subset testing can be avoided.

An illustration of the algorithm is provided in Figure 9. Suppose that there are two conjunctive axioms in the present node, concept names are added to the label of the node in the sequence of A_1, A_3, A_2. Finally, the algorithm reveals that the first conjunctive axioms should be applied to this node.

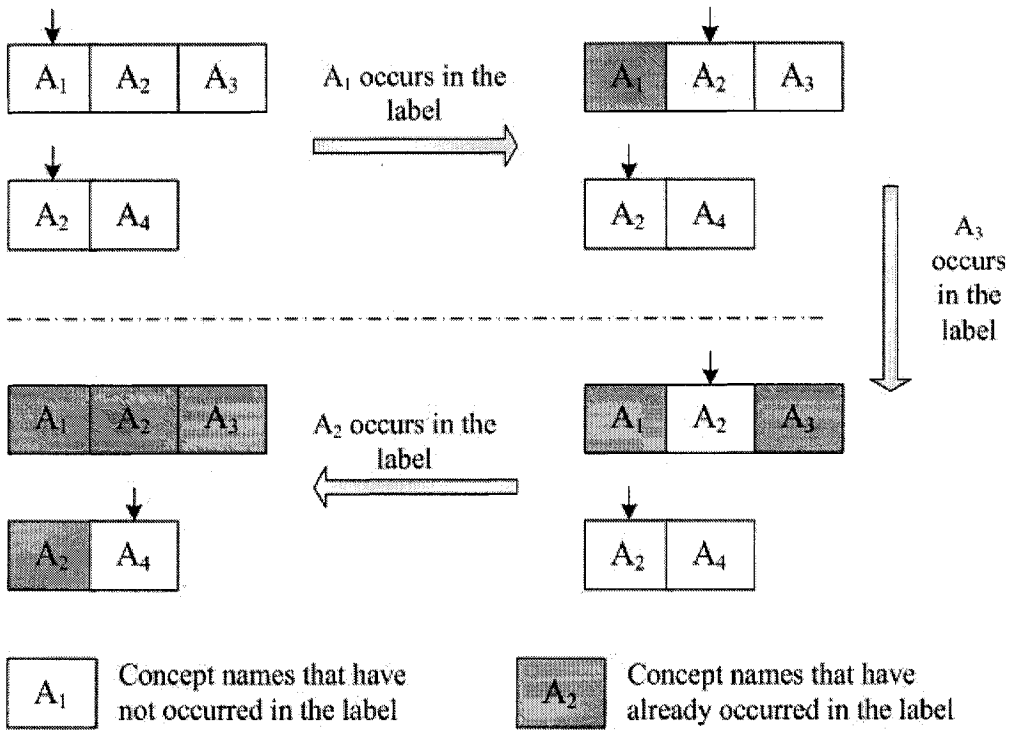

Figure 9: Matching Algorithm for Conjunctive Axioms in Tableau Expansions

Binary versus Conjunctive Absorption

A misleading conclusion might be drawn from Table 6 that conjunctive absorptions push more overhead to the reasoning than binary absorptions[2]. However, it may be the case that conjunctive absorptions are favored. Observe that the number of subset tests in Table 6 only describes *local* information, i.e., subset testing is restricted to one node. Conversely, a *global* consideration of subset testing in all nodes could, in some case, prefer to use conjunctive axioms. A brief explanation toward this statement is that the chance of executing the conjunctive expansion rule is smaller than that of the binary expansion rule. An explanatory scenario follows (Figure 10).

It is easy to see that the general axiom can be absorbed by conjunctive absorption. Obviously, the produced conjunctive axioms will be triggered only if there are at least four concept names in the node labels during concept satisfiability tests. For this KB, no subset testing (and thus no application of the conjunctive expansion rule) will be triggered to check satisfiability of concepts D, E, F. On the contrary, subset testing is always necessary if the general axioms are absorbed into 3 binary axioms. Nevertheless, to degrade the performance of conjunctive absorptions of this

[2]Naively implemented, the number of subset tests for binary axioms is quadratic to the total number of concept names in the label of some node, while it could be exponential for conjunctive axioms.

$$\mathcal{T}_g = \{\top \sqsubseteq \neg A_1 \sqcup \neg A_2 \sqcup \neg A_3 \sqcup \neg A_4 \sqcup C\}$$
$$\mathcal{T}_u^A = \{$$
$$D \sqsubseteq A_1 \sqcap A_2;$$
$$E \sqsubseteq A_1 \sqcap A_2 \sqcap A_3;$$
$$F \sqsubseteq A_1 \sqcap A_2 \sqcap A_4;$$
$$\ldots\}$$

Figure 10: A Sample KB Suitable for Conjunctive Absorptions

KB is quite easy: add to $\mathcal{T}_g$ an axiom of the form $\top \sqsubseteq \neg A_1 \sqcup \neg A_2 \sqcup C_1$, then subset testing become unavoidable for all concept satisfiability tests.

To sum up, the choice between conjunctive absorption and binary absorption largely depends on the characteristics of the axioms of an input KB. It should be pointed out that our experiments implied that for either binary absorption or conjunctive absorption the overhead of subset tests can not be ignored in practice.

4.2 Extended Conjunctive Absorption

Note that binary absorptions do not permit negated concept names to occur on the LHS of binary axioms, moreover, one of the concept names, let us say A, on the LHS of some binary axiom should not have its negation $\neg A$ defined in any unfoldable TBoxes. The same rules apply to the conjunctive absorption: not all the negations of the conjuncts on the LHS are allowed to be defined in an unfoldable TBox, as seen in Proposition 4.1.1. In other words, at least one of the negated conjuncts never occurs on the LHS of any unfoldable TBox. Such a restriction, however, can be relaxed by using the enhanced concept absorption (see Section 3.2.2), which resolves negations by introducing extra axioms.

Is it really necessary to introduce new, very likely not absorbable, axioms to ensure the correctness of absorptions when both concept names and their negations are defined in unfoldable TBoxes due to absorptions? A counterexample can show a plausible reason why additional axioms need to be introduced, as described in Figure 11.

The last concept in Figure 11 is actually *unsatisfiable* in the TBox before absorption, but it turns out to be *satisfiable* w.r.t. the TBox in tableau based reasoning due to the problematic absorptions.

$\mathcal{T}_g = \{\top \sqsubseteq \neg A \sqcup B \sqcup C_1\}$
$\mathcal{T}_u^A = \{\ A \sqsubseteq C_2;\ B \sqsubseteq C_3;\}$

Absorption adding no axiom:

$\mathcal{T}_g = \emptyset$
$\mathcal{T}_u^A = \{A \sqsubseteq C_2;\ B \sqsubseteq C_3;\}$
$\mathcal{T}_u^{\sqcap} = \{A \sqcap \neg B \sqsubseteq C_1;\}$
Satisfiability of concept $A \sqcap \neg C_1 \sqcap \neg C_3$?

Figure 11: A Counter Example

However, the correctness of the absorption is regained by adding an extra axiom $A \sqsubseteq C_1 \sqcup C_3$, which is exactly the negation[3] of the target concept. This observation is generalized in Theorem 4.2.1.

Theorem 4.2.1. *Let* $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$, *where* $\mathcal{T}_u = \{\neg A_1 \sqsubseteq C_1, B_1 \sqsubseteq C_2\}$ *and* $\mathcal{T}_g = \{\top \sqsubseteq \neg A_1 \sqcup \neg A_2 \sqcup B_1 \sqcup C\}$ *for* $A_1, A_2, B_1 \in \boldsymbol{AC}$; *then* $\mathcal{T}' = \mathcal{T}_u' \cup \mathcal{T}_g'$ *where* $\mathcal{T}_u' = \{\neg A_1 \sqsubseteq C_1, B_1 \sqsubseteq C_2, A_1 \sqcap A_2 \sqcap \neg B_1 \sqsubseteq C\}$ *and* $\mathcal{T}_g' = \{A_2 \sqsubseteq C \sqcup C_1 \sqcup C_2\}$ *is a correct absorption of* $\mathcal{T}$.

Proof. This amounts to prove that both $\mathcal{T} \models \mathcal{T}'$ and $\mathcal{T}' \models \mathcal{T}$ hold.

First, let us prove $\mathcal{T}' \models \mathcal{T}$. Every axiom in $\mathcal{T}$ trivially follows from its corresponding axiom in $\mathcal{T}'$. Hence, TBox $\mathcal{T}$ follows from $\mathcal{T}'$.

Then we prove $\mathcal{T} \models \mathcal{T}'$. Every axiom in $\mathcal{T}'$ except $A_2 \sqsubseteq C \sqcup C_1 \sqcup C_2$ trivially follows from its corresponding axiom in $\mathcal{T}$, so we only need to prove axiom $A_2 \sqsubseteq C \sqcup C_1 \sqcup C_2$ follows from $\mathcal{T}$ as well. Since the enhanced concept absorption is a correct absorption, we base our remaining proof on that absorption.

We rewrite (similar to applying concept absorption **CAP**) the general axioms in $\mathcal{T}$ so that $\mathcal{T}_u = \{\neg A_1 \sqsubseteq C_1, B_1 \sqsubseteq C_2\}$ and $\mathcal{T}_g = \{A_1 \sqsubseteq \neg A_2 \sqcup B_1 \sqcup C\}$, to which the enhanced concept absorption is applicable. After an application of the enhanced concept absorption, $\mathcal{T}_1 = \mathcal{T}_u : \{\neg A_1 \sqsubseteq C_1, B_1 \sqsubseteq C_2, A_1 \sqsubseteq \neg A_2 \sqcup B_1 \sqcup C\} \cup \mathcal{T}_g : \{\top \sqsubseteq C_1 \sqcup \neg A_2 \sqcup B_1 \sqcup C\}$. Again, the general axiom could be rewritten (similar to applying concept absorption **CAN**) as $\neg B_1 \sqsubseteq C_1 \sqcup \neg A_2 \sqcup C$ to enable another application of the enhanced concept absorption, which results in the following TBox: $\mathcal{T}_2 = \mathcal{T}_u : \{\neg A_1 \sqsubseteq C_1, B_1 \sqsubseteq C_2, A_1 \sqsubseteq \neg A_2 \sqcup B_1 \sqcup C, \neg B_1 \sqsubseteq C_1 \sqcup \neg A_2 \sqcup C\} \cup \mathcal{T}_g : \{\top \sqsubseteq C_1 \sqcup \neg A_2 \sqcup C_2 \sqcup C\}$.

Now it is evident that axiom $A_2 \sqsubseteq C \sqcup C_1 \sqcup C_2$ trivially follows from $\mathcal{T}_2$ because of that general

[3] Because the negation of the target concept is $\neg(A \sqcap \neg C_1 \sqcap \neg C_3)$, i.e., $\top \sqsubseteq \neg A \sqcup C_1 \sqcup C_3$.

axiom in $\mathcal{T}_2$. Since $\mathcal{T}_2$ is merely a variant of the original TBox $\mathcal{T}$, axiom $A_2 \sqsubseteq C \sqcup C_1 \sqcup C_2$ follows from $\mathcal{T}$ as well, which proves that every axiom in $\mathcal{T}'$ follows from $\mathcal{T}$, i.e. $\mathcal{T} \models \mathcal{T}'$.

□

To add additional axioms is not desirable, and it can be avoided sometimes. A special case avoiding adding axioms is described in Corollary 1.

Corollary 1. *Let* $\mathcal{T} = \mathcal{T}_u \cup \mathcal{T}_g$, *where* $\mathcal{T}_u = \{\neg A_1 \sqsubseteq C_1, B_1 \sqsubseteq C_2\}$ *and* $\mathcal{T}_g = \{\top \sqsubseteq \neg A_1 \sqcup B_1 \sqcup \neg C_1 \sqcup \neg C_2\}$ *for* $A_1, B_1 \in \boldsymbol{AC}$; *then* $\mathcal{T}' = \mathcal{T}_u' \cup \mathcal{T}_g'$, *where* $\mathcal{T}_u' = \{\neg A_1 \sqsubseteq C_1, B_1 \sqsubseteq C_2, A_1 \sqcap \neg B_1 \sqsubseteq \neg C_1 \sqcup \neg C_2\}$ *and* $\mathcal{T}_g' = \emptyset$, *is a correct absorption of* $\mathcal{T}$.

Proof. This is an immediate consequence following from Theorem 4.2.1 because the additional axiom $\top \sqsubseteq \top$ is trivial and unnecessary. □

It is worth noting that the enhanced concept absorption (**ECA**) naturally follows from the proof of Theorem 4.2.1. Evidently, as long as some concept name and its negation occur on the LHS of two axioms (in one unfoldable TBox), it is valid to "resolve" both axioms by introducing an additional axiom whose LHS is the *resolvent* of the LHS of the previous two axioms and whose RHS is the disjunction of the RHS of those two axioms.

Due to their tight relationship with the resolution method, we refer to the extended conjunctive absorption and the enhanced concept absorption as the *resolution based* absorptions. Given the purpose of absorptions, the resolution based absorptions seem to be out of the scope of absorptions because they add extra, very likely complicated, general axioms to a TBox. Nonetheless, they can serve as auxiliary absorptions once the new general axioms, if introduced, can be easily absorbed by other absorption techniques.

Chapter 5

A Framework for Planning of Absorption

As the first attempt to apply planning to axiom absorption, our research paves the way for further improvement of optimization techniques, especially absorptions. This chapter presents a framework for planning of axiom absorptions, which has two main components: the classical planner and the dedicated DL reasoner. Based on the proposed framework, implementation can be easily achieved.

To best illustrate this framework, this chapter is structured as follows. The next section (Section 5.1) describes the overall architecture of the framework as well as a glimpse of individual components' functions. Following that, other sections elaborate the preprocessing (Section 5.2), the planner (Section 5.3) and the DL reasoner (Section 5.4).

5.1 Architecture

The framework, a typical process of dealing with axioms of a knowledge base, in itself is easy to follow, as shown in Figure 12. A TBox in a KB can be viewed as a set of axioms. Normally, the terminological axioms are first preprocessed, such as simplification and normalization, to facilitate processes afterward. The preprocessed axioms are then fed into the planner for planning. This planning process is the key component of the framework, which produces a solution (plan) depending on whether a preset goal has been reached. In any case, the output plan is deemed to be the most economical one in terms of cost estimations. Meanwhile, the planner puts those axioms into

appropriate sub-Tboxes, i.e., $\mathcal{T}_g$ and $\mathcal{T}_u$, so that a DL reasoner can seamlessly take over the whole knowledge base for reasoning.

Being quite self-evident, both preprocessing and planning of the framework give rise to an (internal) knowledge base that may be different from the input. However, note that all processes are satisfiability-preserving syntactical transformations.

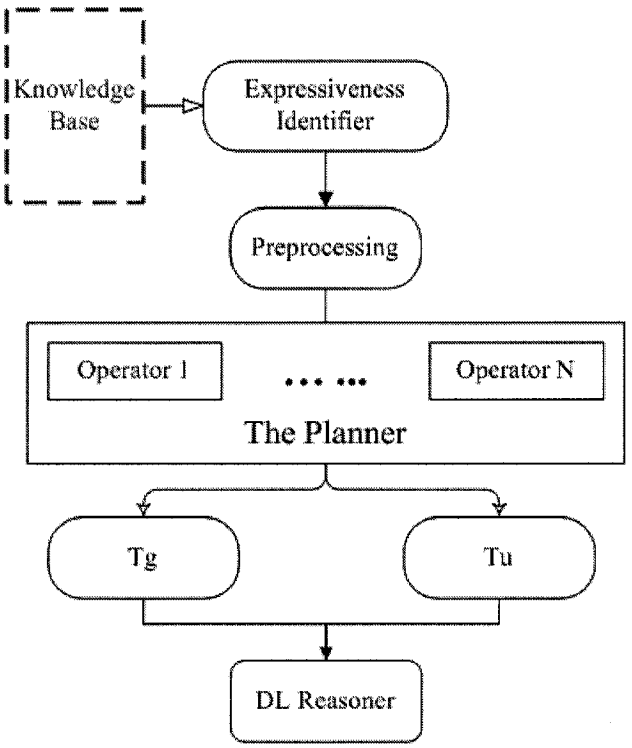

Figure 12: Framework Architecture

Expressiveness Identifier The component named *expressiveness identifier* is designed to filter KBs that could be handled differently because of their expressiveness. This expressiveness checker identifies the expressiveness of an input KB before the KB is fed into the planner and reasoner. The expressiveness of a KB matters in the following two aspects.

On the one hand, some absorptions only work with special concept constructors, for instance, nominal absorption requires the logic to support $\mathcal{O}$, so, the identification of concept constructors can decide which absorptions should be used as operators during the planning. Unnecessary operators will thus remain inactive in the planning in order to avoid potential overhead or even misleading solutions.

On the other hand, some applications or reasoners may require or prefer a particular set of logics

of the input KB. For example, Baader et al. [2005] proposed the logic $\mathcal{EL}$, which is prevailing in life science domains. A notable feature of $\mathcal{EL}$ is that it supports no disjunctions, which means that applying absorptions to $\mathcal{EL}$ KBs jeopardize their original characteristics because disjunctions will be added normally. A KB affected in this way not only changes its expressiveness but also fails to maintain its tractable reasoning in the dedicated $\mathcal{EL}$++ reasoner CEL [Baader et al., 2006]. From this perspective, the expressiveness identifier also helps to preserve the logic during absorptions.

5.2 Preprocessing

Preprocessing is motivated to minimize the computational overhead which would otherwise be amortized in subsequent reasoning services. In the scope of DL reasoning, preprocessing has been devised to change the KB due to the fact that a DL KB, usually designed by users of a particular domain, makes sense for human beings but may deter easy manipulation for machines. Normalization and absorption are two main techniques in this sense.

In this framework, preprocessing can facilitate both DL reasoning and planning. Consequently, various preprocessing techniques could be adopted for planners. Well-founded preprocessing techniques of DL reasoning are usable as well. In this framework, preprocessing with the planner serve as the preprocessing for DL reasoning.

We maintain that preprocessing techniques to be used rely upon the planner, but a few common ones can be named. This section thus discusses some key preprocessing techniques for the framework, though concrete implementation can use as many preprocessors [see Tsarkov et al., 2007, Section 3] as necessary.

5.2.1 Normalization

Normalization, on top of everything else, can put concepts into normal forms to ease certain operations over axioms, for example, structural analyses of axioms. Prior to normalization, all axioms could be uniformly rewritten so that the LHS of every axiom is $\top$, and the RHS is ready for normalization (see Figure 13). To refine the process of normalization, it is acceptable to denote the RHS in sets.

Negation Normal Form is a well-known normal form such that negations occur directly before concept names. This is achieved through a translation between dual pairs of operators ($\exists$ and $\forall$, $\sqcap$ and $\sqcup$, etc.) by applying De Morgan's Laws. Horrocks [2003] presented two functions to obtain

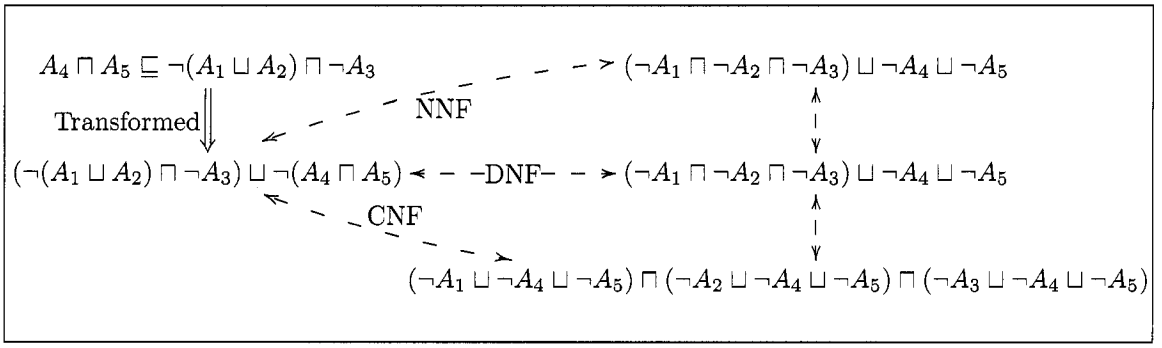

Figure 13: Normal Forms in Preprocessing

NNF with additional simplification given some concept expressions.

CNF and DNF have been quite influential as well. The former consists of a conjunction of *clauses*, and the latter is made up of a disjunction of *terms*. Every concept, in principle, can be converted to its *equivalent* concept in CNF (DNF).

We assume that above normal forms suffice for this framework, but other normal forms may be needed in some cases. Fortunately, Darwiche and Marquis [2002] studied a considerable number of normal forms to help decide which to use for different applications.

5.2.2 Synonym Replacement

Synonyms refer to concept names A that have a definition of the form $A \equiv B$, where B is a concept name as well. Obviously, A and B are synonyms because they are just two names for the same concept. Without proper treatment, these definitions are deemed as equivalence axioms, each of which might contribute two unnecessary general axioms. Moreover, synonyms may introduce fake cycles or unnecessary dependencies into the TBox.

Actually, a TBox could be further simplified by replacing every occurrence of B with A. For instance, if a TBox contains the following axiom $\{A \equiv B, \top \sqsubseteq B \sqcup \exists R.C\}$, it is reduced to $\{\top \sqsubseteq A \sqcup \exists R.C\}$ by substituting B with A. Practically, every synonym definition of A is removed from the TBox, and all occurrences of the synonyms of A are superseded by A.

During synonym replacement, a TBox must always satisfy the default assumptions introduced in Section 2.1.1. In other words, applications of synonym replacement should not break the assumptions for an unfoldable TBox.

5.3 Planner Architecture

Preprocessing tailors an input KB for the subsequent planning that may stipulate axioms in certain format. No specific format is suggested in this framework, yet it is possible to select a representative for our discussions. In the following sections, after the normalization phase, all the axioms are assumed to be represented as sets whose elements are disjuncts of those axioms. In fact, one axiom in CNF can produce more than one set by separating its conjuncts. The consequence of this notation is that disjunctions, which are frequently referred to during planning, can be easily identified. The elements in a set, when no ambiguity is caused, are called *items* of the original axiom.

Any general planner is applicable to this problem despite a few minor changes to be noted. Without loss of generality, Figure 14 delineates an adaptable generic planner.

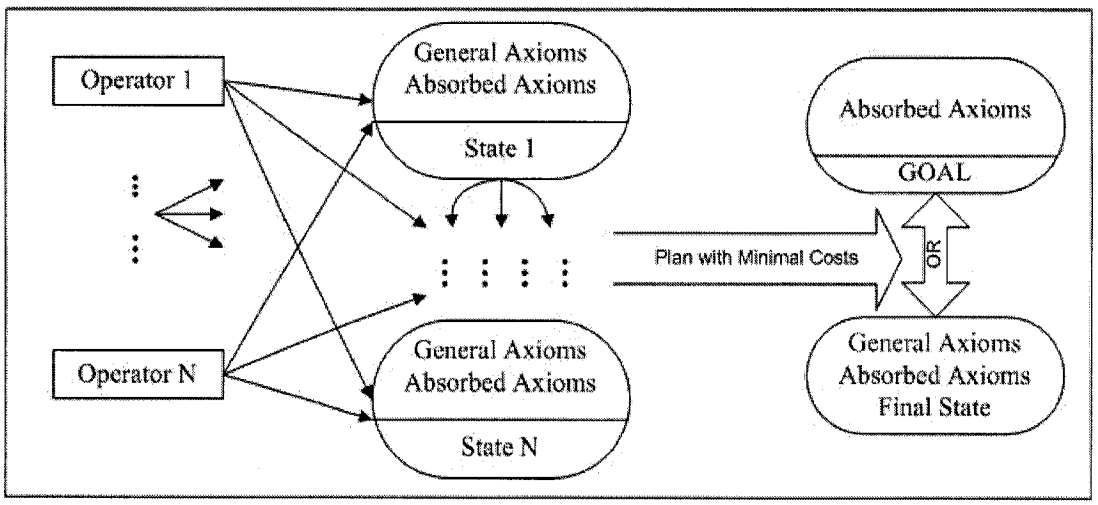

Figure 14: The Architecture of a Generic Planner

The following sections manifest the architecture of the planning framework by introducing the representation of the planning problem. At the end of this section, a planning procedure simulating the work flow of this planning system is presented, along with specific considerations to build a practical planner.

5.3.1 States

As discussed in Section 2.2.1, literals are shown later in constructing states and the preconditions and effects of operators.

In our framework, variable symbols are denoted by three or less characters, possibly with subscripts, in the slanted shape, such as *gci*. Constant symbols can be easily identified because they are

just **AC**, **NA**[1] and names of TBoxes, for example $\mathcal{T}_g$. Predicates are denoted in `typewrite` font; a typical predicate would be `isGeneral`. As a convention in FOL, an atomic formula (atom) is formed by applying some predicate to constants and/or variables. Furthermore, atoms and negated atoms are called *literals*, for example, `isGeneral`(*gci*) is a positive literal. The list of literals devised in this thesis are depicted in Table 7.

Literal Name	Explanation
`equals`(x, y)	true if concept x is the same as concept y.
`belong`(x, **AC** (or **NA**))	true if concept x is a (or negated) concept name.
`simpleCon`(x)	true if either `belong`(x, **AC**) or `belong`(x, **NA**) hold.
`univRes`(x)	true if concept x is a universal restriction.
`hasQual`(x, y)	true if `univRes`(x) holds and y is the qualification of x.
`isGeneral`(*gci*)	true if axiom *gci* is in $\mathcal{T}_g$, i.e., *gci* is a general axiom.
`hasDisjunct`(*gci*, x)	true if concept x is a disjunct of axiom *gci*.
`definedIn`(x, $\mathcal{T}$)	true if concept x is defined in TBox $\mathcal{T}$, i.e., it appears on the LHS of some unfoldable axiom(s) in $\mathcal{T}$. $\mathcal{T}$ could be any unfoldable TBoxes.
`extract`(x, y)	true if either of the following hold: 1. both `hasQual`(x, y) and `simpleCon`(y) hold; or 2. both `hasQual`(x, z) and `extract`(z, y) hold.

Table 7: Definitions of Literals

A typical state looks like $s_1 = \{$`isGeneral`(*axiom*), `hasDisjunct`(*axiom*, A), `simpleCon`(A)$\}$.

Initial State

An *initial state* can be any state, which usually contains all the necessary ground atoms to express relevant background knowledge. In this framework, an initial state is always the one describing the original input knowledge base. Designers should ensure that all the necessary knowledge will be encoded in states while irrelevant conditions will not.

For our planning problem, an initial state usually describes every axiom occurring in an input KB. Each axiom could be considered as a candidate for absorptions unless it is naïvely put into some unfoldable TBox through preprocessing. For example, the preprocessor may move all equivalence definitions directly to $\mathcal{T}_e$. This preprocessor augments $\mathcal{T}_e$, hence an initial state should also depict $\mathcal{T}_e$ appropriately. An initial state must encode TBox information, such as which TBox every axiom belongs to. Other than that, an initial state has to encode elements of every general axiom represented as set. Such elements are the direct sources of absorptions. Note that other extensions to this framework might require additional KB descriptions such as ABox assertions. Figure 15 serves

[1]Recall that **AC** is the set of concept names, and **NA** is the set of negated concept names.

as a template for designing initial states in implementations. At present, ABox assertions are not considered by our framework.

```
1: {Describes the status of axioms; n: # of general axioms, n > 0}
2: for all k such that 1≤ k ≤n do
3:    isGeneral(gci_k) ← true
4:    {Describes items in each general axiom; m: # of items of axiom k, m ≥ 0}
5:    for 0≤ i ≤ m do
6:       hasDisjunct(gci_k, x_i) ← true
7:    end for
8: end for
9: {Describes the status of unfoldable TBoxes}
10: for all non-empty unfoldable TBoxes T do
11:    for all x defined in T do
12:       definedIn(x, T) ← true
13:    end for
14: end for
```

Figure 15: Template for Designing an Initial State

5.3.2 Operators and Actions

An operator is a representation of some absorption using defined literals. The following discussion details the formulation of operators, which includes concept absorption (**CA**), conjunctive absorption (**CJA**), and inverse role based absorption (**IRBA**). Note that the enhanced concept absorption (**ECA**) is not part of the formulation due to its violation of classical planning restrictions. The same reason goes for the extended conjunctive absorption. Nominal absorption (**OA**), although suitable for a classical planner, are omitted in the formulation because they can be simulated by **CA** and **IRBA**. To be more rigorous, we have incorporated the absorption on equivalence axioms.

Operator Formulation

Planning operators are instantiated into *actions* to be applied to states. Intuitively, every absorption can be formulated as one or more operators depending on the preconditions and effects of these operators. Similar to the representation of states, preconditions and effects are just sets of literals. Observe that there is no requirement to prohibit positive literals here, but rigid relations, which are *invariant* over all states, never appear in the effects.

Table 8 formally presents operators to cover currently known absorptions through the use of literals defined in Table 7. In this table, $\neg x$ refers to the normalized negation of some concept name

x. For instance, if x is $\neg A$, then $\neg x$ becomes A instead of $\neg\neg A$. Thus, if x belongs to **NA**, $\neg x$ belongs to **AC**, and vice versa.

CONCEPTABSPOS (gci, x)	To move an axiom gci from $\mathcal{T}_g$ to $\mathcal{T}_u^A$;
Preconditions	isGeneral(gci); hasDisjunct(gci, x); belong(x, **NA**); ¬definedIn($\neg x$, $\mathcal{T}_e$); ¬definedIn(x, $\mathcal{T}_u^{\neg A}$)
Effects	definedIn(x, $\mathcal{T}_u^A$); ¬isGeneral(gci)

CONCEPTABSNEG (gci, x, x_1, ..., x_n)	To move an axiom gci from $\mathcal{T}_g$ to $\mathcal{T}_u^{\neg A}$;
Preconditions	isGeneral(gci); hasDisjunct(gci, x); belong(x, **AC**); ¬definedIn(x, $\mathcal{T}_e$); ¬definedIn(x, $\mathcal{T}_u^A$); **when** definedIn($x \sqcap x_1 \sqcap \ldots \sqcap x_n$, $\mathcal{T}_u^{\sqcap}$), **then** ∃ i: ¬definedIn($\neg x_i$, $\mathcal{T}_u^{\neg A}$), 1≤i≤n
Effects	definedIn(x, $\mathcal{T}_u^{\neg A}$); ¬isGeneral(gci)

CONJUNCTIVEABS (gci, x_1, ..., x_n)	To move an axiom gci from $\mathcal{T}_g$ to $\mathcal{T}_u^{\sqcap}$;
Preconditions	isGeneral(gci); hasDisjunct(gci, x_1); ... ; hasDisjunct(gci, x_n) where n≥2; belong(x_i, **NA**), $1 \leq i \leq n$; ¬definedIn($\neg x_i$, $\mathcal{T}_e$), $1 \leq i \leq n$; ∃j, 1≤ j≤ n, ¬definedIn(x_j, $\mathcal{T}_u^{\neg A}$)
Effects	definedIn($x_1 \sqcap \ldots \sqcap x_n$, $\mathcal{T}_u^{\sqcap}$); ¬isGeneral(gci)

INVERSEROLEABS (gci, x, y)	To move an axiom gci from $\mathcal{T}_g$ to $\mathcal{T}_u^A$ [or $\mathcal{T}_u^{\neg A}$];
Preconditions	isGeneral(gci); hasDisjunct(gci, x); extract(x, y); belong(y, **AC** [or **NA**]); ¬definedIn(y [or $\neg y$], $\mathcal{T}_e$); ¬definedIn(y, $\mathcal{T}_u^A$ [or $\mathcal{T}_u^{\neg A}$])
Effects	definedIn(y, $\mathcal{T}_u^{\neg A}$ [or $\mathcal{T}_u^A$]); ¬isGeneral(gci);

EQUIVALCENABS (gci_1, gci_2, x, y)	To move two axioms gci_1 and gci_2 from $\mathcal{T}_g$ to $\mathcal{T}_e$;
Preconditions	isGeneral(gci_1); isGeneral(gci_2); hasDisjunct(gci_1, x); belong(x, **AC**); hasDisjunct(gci_2, y); belong(y, **NA**); ¬definedIn(x, $\mathcal{T}_e$); equals($\neg x$, y); ¬definedIn($\neg y$, $\mathcal{T}_e$);
Effects	definedIn(x, $\mathcal{T}_e$); ¬isGeneral(gci_1); ¬isGeneral(gci_2)

Table 8: Formulation of Operators

As can be seen from Table 8, the restriction of the classical representation formalism has actually been relaxed by allowing expressions that are more general than just sets of literals in the preconditions and effects. More precisely, the operator formulation in this framework is extended to allow conditional expressions (for instance, when...then) and quantified expressions (for example, ∀ and ∃). Together with other extensions, the extended representation formalism can also be used to describe states. In particular, quantified expressions are sometimes convenient for specifying the initial state and the goal.

5.3.3 Goal

A classical planning problem can be expressed by the initial state, the operators, and the goal, of which the first two have been shown in Sections 5.3.1 and 5.3.2 respectively. In this section, we study the goal of our planning problem.

A goal is *not* a state in a planning problem, instead, it is a set of ground literals. In this framework, ideally, a goal should contain no general axioms, all of which should have been absorbed, for any given KB. However, as is noted from the very beginning, such an ambitious goal may be unreachable. That is, there exist KBs whose general axioms can not be completely absorbed by any absorption. In a concrete implementation, an alternative would be to approximate the goals to guarantee its reachability. For instance, the goal can be set in such a way that a small percentage of the total number of axioms can remain in $\mathcal{T}_g$.

5.3.4 Planning Procedure

After concentrating our attention on details of the planner, this section depicts how a classical planner absorbs general axioms. When a KB is fed into the planner, it is preprocessed to determine the initial state. This initial state should contain sufficient knowledge to describe axioms in the KB that might affect absorptions. Starting from this initial state, operators, i.e., absorptions formulated in the planner, will be subsequently applied to states once their preconditions are met. Meanwhile, more states will be created due to successful applications of the operators.

Note that a state transition function is responsible to select states to form a solution. As it is possible that one or more operators are applicable to one general axiom in a state, there will be a number of states available to be selected by this state transition function. Instead of trying every possibility for a solution, state selection heuristics are generally used to resolve these nondeterministic choices. Cost estimations are then introduced to serve as a part of such heuristics. With the help of heuristics, the planner determines its most promising successor for a solution without randomly choosing from multiple possibilities. As is recognized, sometimes a solution that empties the $\mathcal{T}_g$ is not accessible, then a plan with minimized cost totaled along the path is presented as the solution.

Before devising a new planner, we decided to evaluate known general purpose AI planners to analyze their feasibility and scalability in the DL domain.

Evaluation of SGPlan

One of the most competitive planners, SGPlan 5 [Hsu et al., 2006], has been selected as the representative for general classical planners to show if it scales up to large planning problems in the DL domain.

For the scalability experiments, a simple Tbox containing seven axioms represented in PDDL 3.0 [Gerevini and Long, 2005] is constructed, as shown in Figure 16. At the same time, we define three operators: concept absorption (allowing absorptions into both concept names and their negations), binary absorption, and role absorption. Note that binary absorptions introduce dynamics into the planning domain, which amounts to require online planning (Section 2.2.1), hence if binary absorptions are to be defined as established in [Hudek and Weddell, 2006], they are beyond the expressive power of PDDL 3.0 because offline planning applies to the PDDL specification.

TBox	#	Content	Absorptions
$\mathcal{T}_g$	1	$\top \sqsubseteq \neg C \sqcup \neg E \sqcup D$	$\rightsquigarrow$ *Concept Absorption over* $\neg E$
	2	$\top \sqsubseteq \neg B \sqcup \neg C \sqcup E \sqcup \forall R.F$	$\rightsquigarrow$ *Role Absorption over* $\forall R.F$
	3	$\top \sqsubseteq \neg E \sqcup \forall R.F$	$\rightsquigarrow$ *Concept Absorption over* $\neg E$ *or Role Absorption over* $\forall R.F$
$\mathcal{T}_u$	4	$\neg B \sqsubseteq \ldots$	
	5	$\neg C \sqsubseteq \ldots$	
	6	$E \sqsubseteq \ldots$	
$\mathcal{T}_e$	7	$D \equiv \ldots$	

Figure 16: TBox Designed to Test SGPlan 5. We assume that B, C, D, E, F are concept names, and R a role name. $\mathcal{T}_g$ contains general axioms, $\mathcal{T}_u$ is the unfoldable TBox, and $\mathcal{T}_e$ consists of equivalence axioms.

The desirable application of absorptions is listed on the right hand side of Figure 16. Obviously, all three general axioms can be absorbed. Then, all the seven axioms are replicated, i.e., only the concept names are changed correspondingly in each axiom, to increase the size of this planning problem. The experiments showed that a replication of around 50 copies (150 general axioms) costs around 10 seconds for SGPlan 5, but 60 copies (180 general axioms) could lead to a failure of this planner in the sense that SGPlan 5 is unable to return a solution within 15 minutes. We thus argue that even sophisticated classical planners do not scale up well in the DL domain because a Tbox with hundreds or even thousands of general axioms is not uncommon. To avoid the issue on scalability, a dedicated planning approach as described in the following section is considered.

Scalable Planning

We have just demonstrated that efficient classical planners designed in the AI community rarely enjoy scalability for our problem domain. Given that a planner implemented in this framework is usually incomparable to those planners, we can not expect the planner to scale well if it is designed in the conventional manner. Thus, it makes sense to adjust the framework to allow *scalable planning*.

We maintain that a planner that absorbs all axioms should produce a solution within reasonable time to make up for the overhead that would otherwise have been incurred by reasoning on general axioms. An immediate answer could be to reduce the number of states in a state-transition systems. Considering a planner of m operators and a state with n general axioms, a conventional planner, in the worst case where all operators are applicable to every axiom, would normally create $m \times n$ successors of this node. Given the possibility of a large quantity of general axioms in real ontologies, the planner could be designed to behave in such a manner that fewer states need to be generated, slightly different from the above-mentioned conventional planner.

Particularly, scalable planning amounts to apply one operator to a group of general axioms rather than a single one and leads to the creation of fewer new states. Actually, during this process, a number of actions, i.e., instances of this operator, are executed. Normally, every action results in a new state, as in an ordinary planner, however, in this case, only *one* successor state will be introduced by all these actions of this operator. Evidently, this strategy is just an approximation of conventional classical planning, so, the precision might be affected.

Figure 17 depicts a scenario where scalable planning has a coarser granularity than ordinary planning. The generated states are marked by the absorption and the planning applied to it. For instance, the top-left state marked "**CJA** + Scal." means that this state is obtained by using conjunctive absorption and scalable planning.

In scalable planning, axioms have a uniform structure if successfully absorbed, that is, the absorbed axioms will have the same structure, for instance, the LHS consists of the same number of concept names in the above example. Conversely, ordinary planning is able to produce all possible states, where the axioms resulting from absorptions can have different structures, as shown in the state marked "**CJA** + **CA** + Ord.". Despite the unavoidable loss of coarseness (more precisely, the loss of completeness), scalable planning presented in this framework, however, tends to be more feasible in practice compared to the ordinary one.

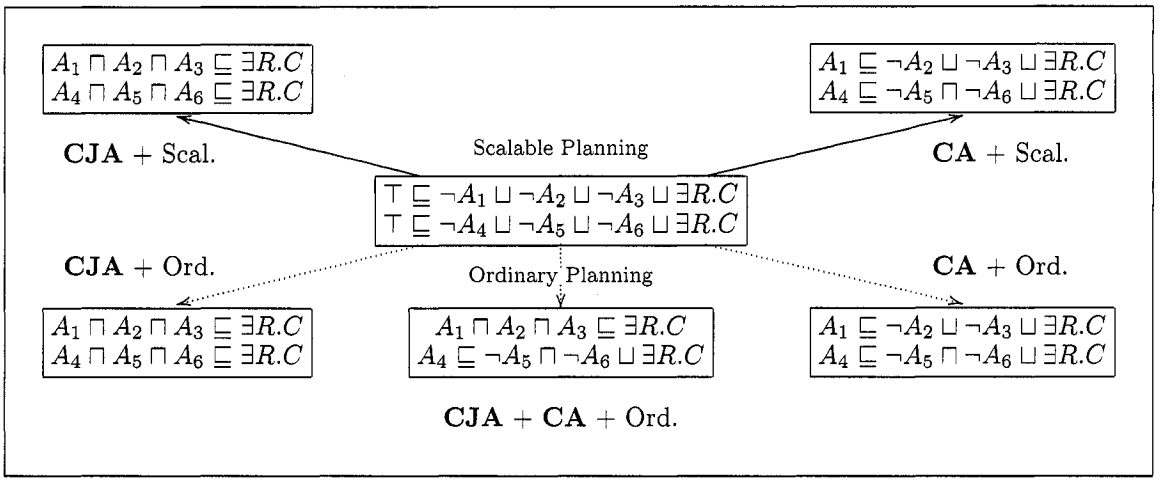

Figure 17: Scalable vs. Ordinary Planning. Each framed box denotes a state, and only representative results are shown. Two operators are considered here: conjunctive absorption and concept absorption.

5.3.5 Cost Mapping

This section extends the previous discussion on the planning framework to allow cost-based optimal planning. Rather than being satisfied by any arbitrary sequence of actions that leads to the goal set, sometimes it is desirable to optimize some criterion, such as the quality of the generated knowledge base. The distinction between optimal and non-optimal planning is not always a concern, yet it is relevant in practice. To address optimal planning in our problem, we start with the cost accumulation during the plan execution.

Let us assume the main motivation of absorptions is that the absorbed knowledge base should facilitate reasoners. The question of cost consideration becomes to what degree this motivation could be reflected by the cost. It is accepted that reasoning performance is attributed to many factors, and general axioms are said to be one of the main sources of nondeterminism. Due to lack of sufficient knowledge about these factors, the cost mapping presented here only contributes to an estimation of the real cost.

This section focuses on the structure of axioms for an estimation of the real cost, which projects the hardness of reasoning about a KB onto structural analyses. At present, the schema for estimating the difficulty of reasoning over a KB is proposed below. It describes what aspects will be considered in a tableau-based reasoning procedure.

Expansion Rules In a tableau-based reasoner, expansion rules are used to "draw" a tableau, i.e.,

how to expand a concept to detect potential conflicts by taking into account relevant TBox information. Usually, an expansion rule to deal with disjunctions or conjunctively absorbed axioms produces overhead that can not be ignored in practice.

TBox Interactions There can be a number of small unfoldable TBoxes, each of which has its own requirements. Additionally, all the unfoldable TBoxes must maintain unfoldability whenever an absorption adds some axioms to them. For instances, $\mathcal{T}_u^A$ assumes that its axioms cannot have the negations of their LHS defined in $\mathcal{T}_u^{\neg A}$. Similar interactions also exist between $\mathcal{T}_u^{\neg A}$ and $\mathcal{T}_u^{\sqcap}$.

Branching During a tableau expansion, reasoners may face non-deterministic choices where they should try various branches to exhaust all possibilities. The main source of branching is from disjunctions. Consequently, disjunctions occurring in axioms have to be considered in a proper manner.

Unfolding Unfolding, with the help of absorptions, plays a key role in efficient tableau-based reasoning. However, no reasoner expects to be effective if it has to unfold numerous times or it unfolds to add a large amount of concepts to the label of some nodes for one concept satisfiability testing. The underlying question would be how absorptions can heuristically take advantage of unfolding.

A number of factors could be presented other than the aforementioned concerns, but there still exists great difficulty to map these factors appropriately and reasonably onto cost estimations. It remains to be seen if the cost evaluation ameliorates if more factors are mapped.

5.3.6 Cost Metrics

Cost mapping is designed to monitor the hardness of reasoning over some KB, and cost metrics are used to quantitatively represent the hardness. For structures of axioms, the left-hand side (LHS) and the right-hand side (RHS) are separately studied. For absorbed axioms, this section provides a thorough analysis on how to project KB hardness onto cost. Cost estimation on general axioms and other factors are presented at the end of this section.

Cost Analysis on the LHS

The principle of the cost analysis on the LHS, roughly speaking, is to assess the gains and side effects after applying an action to some general axioms. For instance, if an application of concept absorption absorbs a general axiom to B in $\mathcal{T}_u^A$, then one of the gains is that this general axiom is resolved, however one of its negative impacts is that other general axioms lose the chances to be absorbed to $\neg B$ in $\mathcal{T}_u^{\neg A}$ based on the operator formulation. The associated action, which results in some follow-up state, is called an *ingress action* w.r.t. that state.

Sequentially, this section presents the cost metrics for different concepts occurring on the LHS of some absorbed axiom. Naturally, the type of concepts discussed reveals their correspondence to relevant unfoldable TBoxes.

I. Concept A on the LHS General axioms that have been successfully absorbed by the ingress action are called *processed* (general) axioms. The generated axioms then define A in $\mathcal{T}_u^A$. At the same time, some general axioms are *blocked* by this ingress action because otherwise they can be absorbed to axioms of the form $\neg A$ in $\mathcal{T}_u^{\neg A}$. These general axioms are called *blocked* (general) axioms to indicate the negative impact of executing this action.

A partial cost estimation can now be made by counting the difference between the number of processed and blocked axioms. Every time an action is activated and finished in some state, the counting result is recorded. Observe that this counting is not accurate, thus a weight needs to be assigned to this counting to adjust the offset. This counting is suitable for both scalable and ordinary planning.

II. Concept $\neg A$ on the LHS In this case, processed axioms, defined in a similar way as in case I, will have $\neg A$ defined in $\mathcal{T}_u^{\neg A}$, which block two kinds of (blocked) axioms. First, contrary to blocked axioms in case I, general axioms which are able to define A in $\mathcal{T}_u^A$ are blocked. Second, some general axioms that can be conjunctively absorbed may be blocked as well, as can be seen in the preconditions of conjunctive absorption in Table 8. Considering a general axiom that is to define $A_1 \sqcap A_2 \sqcap A_3$ in $\mathcal{T}_u^{\sqcap}$, if $\neg A_1$ and $\neg A_2$ already have their definitions in $\mathcal{T}_u^{\neg A}$, then a definition of $\neg A_3$ in $\mathcal{T}_u^{\neg A}$ obviously keeps this axiom from being amenable to conjunctive absorption. As a result, this general axiom is blocked by this action as well.

Similarly, the cost can be quantified by counting processed and blocked axioms, with a preset weight for cost adjustments.

III. Concept $A_1 \sqcap \ldots \sqcap A_n$ on the LHS Let us assume there are n general axioms GCI_i $(1 \leq i \leq n)$ to define $\neg A_1$ to $\neg A_n$ in $\mathcal{T}_u^{\neg A}$ of some state s. In addition, s contains GCI_k and another general axiom that can have $A_1 \sqcap \ldots \sqcap A_n$ defined in $\mathcal{T}_u^{\sqcap}$. In order to absorb the second axiom into $\mathcal{T}_u^{\sqcap}$, at least one of GCI_i needs to be blocked, and hence becomes a blocked axiom. It is easy to see that the probability to block GCI_k is very low because this would require that all other axioms GCI_i $(i \neq k)$ have no definitions of $\neg A_i$ $(i \neq k)$ in $\mathcal{T}_u^{\neg A}$. Hence, the quantification of cost can be obtained by the counting, a weight, and the probability.

Cost Analysis on the RHS

The RHS of an absorbed axiom can be evaluated by considering disjunctions and unfolding. The former causes branches to increase non-determinism, while the latter may unfold a long chain of concepts. Both can jeopardize the reasoning performance to some degree.

I. The number of disjunctions Disjunctions are one of the main concerns of optimizations. Generally, the fewer disjunctions on the RHS of an axiom, the fewer branches in tableau expansions. The simplest way to compute this cost is counting the number of disjuncts. Prior to that, one has to make sure that axioms are normalized so that no redundant disjunctions are added to the cost.

II. Saturation of concepts Unfolding alone only introduces superclass concepts of the present concept. It may happen that some concept, directly or indirectly, is unfolded to a set of concepts containing itself. Although this situation could be handled by other optimizations in DL reasoners, a solution to prevent such cyclic references is to saturate every concept. In this way, the overhead of cyclic unfolding is pushed to absorptions instead of to tableau expansions.

To terminate the cyclic introduction of some named concept itself, a procedure called **saturation** can be introduced. That is, every named concept keeps track of its named superclass concepts until it encounters itself. The cost on the RHS is then determined after all named concepts have been saturated.

The method to quantify the cost related to unfolding varies in different implementation. For example, for some saturated named concept A, one can calculate its unfolding cost by measuring the size of the set that all the named superclass concepts of A belong to. One can also compute the cost of A by measuring the *levels* of saturation. Considering the following three axioms: $A_1 \sqsubseteq A_2 \sqcap \ldots$, $A_2 \sqsubseteq A_3 \sqcap \ldots$ and $A_3 \sqsubseteq C \sqcap \ldots$, if C is a concept expression that requires no more unfolding, then

A_3 is saturated directly, which has only one level of saturation, whereas A_2 (A_1) has two (three) levels of unfolding in order to be saturated.

Other Metrics

Apart from cost metrics about an absorbed axiom, cost metrics are attached to absorptions themselves. Recall that the axioms resulting from some absorption need to be expanded by tableau rules, each of which might require a different cost. To slightly reflect this idea, every absorption itself can be assigned a weight (intuitively, the priority). For example, compared to concept absorption, conjunctive absorption can be assigned a lower weight to indicate that it generates axioms to be dealt with by a relatively more expensive tableau expansion rule.

Occurrences of named concepts (and their negations) are good candidates for cost metrics as well. An absorption process itself is non-deterministic, for example, when a general axiom has more than one (negated) concept name amenable to the concept absorption, concept absorption seems impartial in selecting any of these concepts. An intuition is that some concepts may have more beneficial effects for subsequent absorptions than others. In view of this fact, the frequency of all possible (negated) named concepts occurring in a TBox may be used as a clue to select a more suitable absorption concept. An underlying notion is that such statistics about occurrences is readily applicable to cost metrics. For instance, concept absorption has the option to absorb a general axiom into A or $\neg A$, then occurrences of $\neg A$ and A appearing in all general axioms give the absorption a hint which one can be chosen to minimize the number of general axioms that could have been blocked by the choice. However, it should be noted that absorptions allowing A and $\neg A$ to be considered as two different operators (i.e., **CAP** and **CAN**) in the planning do not need such counting of occurrences because planning tries all situations and always selects the one with the least cost.

Similar observation holds for lazy unfolding. Suppose that an absorption has to randomly absorb a general axiom into either A or B, the number of occurrences of some concept name in unfoldable TBoxes indicate if this concept will bring in lots of additional concepts during tableau expansions. Consequently, if B scarcely appears in any unfoldable TBox, the absorption can absorb general axioms to B to avoid introducing a large amount of concepts to a node in tableau expansions. Otherwise, axioms can be absorbed to A by incurring some cost. Above strategy applies to binary absorption too. Though it is unavailable in classical planning, binary absorption can take advantage of occurrences of concepts to guarantee that the LHS of an absorbed axiom is the most frequently

used concepts.

The overall cost in one state can be calculated easily by a function alike to this:

$$CostInState = \sum_{Op_i \in Operators} WeightOfOp_i \times (|GainsOfOp_i| - |LossOfOp_i|) \qquad (10)$$

5.4 DL Reasoner

So far, our illustration is on the preprocessing level. Figure 12 detaches the planning component from the DL reasoner so that the planning component can be coupled to various DL reasoners as long as the reasoners meet specific requirements imposed by the planning component. Some requirements are imposed only when corresponding operators are formulated in the planning, while other requirements are fundamental ones for a tableau-based DL reasoner. This section specifies what functionality might be needed for a DL reasoner to work in the framework.

Expressiveness As has been argued, some known absorptions are associated with special logic constructors. In this framework, $\mathcal{I}$ and $\mathcal{O}$ are explicitly demanded by inverse role based absorptions and nominal absorptions respectively. The consequence is that a DL reasoner must be able to reason about KBs residing in the expressiveness of $\mathcal{ALCI}$ or $\mathcal{ALCIO}$.

Preprocessing Though the planning component serves as one of the preprocessors, i.e., an absorption preprocessor, of a DL reasoner, additional preprocessing, such as simplification and normalization, are performed to simplify the input KB into a form to facilitate reasoning. Tsarkov et al. [2007] surveyed effective optimizations for this purpose, including taxonomic encodings and synonym replacement, told cycle elimination and so on.

Tableau Rules For any working DL reasoners that deal with KBs in $\mathcal{ALCI}$, the only requirement on tableaux rules is the addition of another rule for conjunctive axioms. Typically, the rule should be triggered when a node containing a set of concept names covers the LHS of some conjunctive axiom in $\mathcal{T}_u^{\sqcap}$. The details are available in Section 4.1.

For inferences, the DL reasoners should at least be able to check concept satisfiability, otherwise it is hard to compare reasoning performance with other reasoners. Although there is no requirements for optimization to be implemented, some basic ones, such as blocking, which guarantees terminations of tableau algorithms, should be considered.

Chapter 6

System Description and Evaluation

This chapter describes an implemented system for planning of axiom absorptions. The system design follows the suggested architecture in Chapter 5, which has two major parts: a planning component and a DL reasoner. In the first phase, an input KB is examined to identify its expressiveness. Then, general axioms are recognized and processed in an appropriate way, i.e., by planning. Immediately after that, a DL reasoner performs reasoning on the modified knowledge base.

To ease the presentation, this system is named PAR. The first part (Section 6.1) of this chapter gives a description of the system implementation, while Section 6.2 analyzes the proposed framework with empirical observations.

6.1 System Description

The implementation of the framework is for conducting experiments about absorptions, hence, it is quite straightforward. An absorption planning system can be implemented in any programming language. This system is implemented in Java because the OWL API [Bechhofer et al., 2003], an influential interface for parsing OWL ontologies, is used in this system as the parser.

6.1.1 The Classical Planner

The implemented planner is a classical state-space planner. Particularly, both ordinary planning and scalable planning as presented in Section 5.3.4 are implemented for comparison. This section describes the decisions and assumptions made in practical aspects.

Expressiveness Identifier

PAR has a module to identify the expressiveness of a given KB, though only $\mathcal{EL}$ and non-$\mathcal{EL}$ are distinguished for an input KB. If the KB is non-$\mathcal{EL}$, the system will then proceed by applying planning using all the available operators, otherwise only conjunctive absorptions will be applied to this KB. This identifier can be disabled as needed to ignore the logic restrictions, thus, all ontologies can be exposed to planning with all operators.

Preprocessing

Upon entering the system, a general axiom is transformed in such a manner that its LHS becomes $\top$. Subsequently, its RHS is decomposed into one or more concepts. The LHS of each axiom is $\top$ and the RHS is a number of disjuncts denoted as sets. For example, a general axiom $\neg A \sqsubseteq B \sqcap \forall R.C$ is eventually rewritten to two axioms $\top \sqsubseteq A \sqcup B$ and $\top \sqsubseteq A \sqcup \forall R.C$, represented as sets $\{A,\ B\}$ and $\{A,\ \forall R.C\}$ respectively. Normalization is simultaneously achieved in the course of representing the axioms.

Operators, States and the Goal

Operators that are formulated in this planner consist of concept absorptions (**CAP** and **CAN**), conjunctive absorption (**CJA**), inverse role based absorption (**IRBA**) and equivalence absorption (**EA**). As stated in the framework, domain and range absorptions (**DRA**) and nominal absorption (**OA**) can be simulated by inverse role based absorption, and the latter (requiring $\mathcal{O}$) can not be dealt with in the implemented DL reasoner, and thus is not considered.

A state in this planner is a set of positive literals, the same as described in the framework. The system imposes no extra conditions on the initial state either. The goal of the planner is designed to allow *one percent of* the total number of general axioms[1] to remain unabsorbed. The goal is regarded appropriate because this system is not expected to be powerful enough to successfully absorb all general axioms in every KB.

Planning Algorithm and Heuristics

Our implementation adopts state space planning, where the process of planning turns out to be a search problem in a graph. The search algorithm implemented in this system is A*, one of the

[1]The number of general axioms might increase after preprocessing. Here, we count the number of axioms before preprocessing in order not to overestimate it.

best-established forward search algorithms. Although it is known that A* is *complete*, this planner adapts scalable planning described in Section 5.3.4 in the framework, which may overlook some combinations of absorptions and possibly result in incomplete plans.

The cost function $f(n) = g(n) + h(n)$ is formulated, where $g(n)$ is to calculate the cost-so-far of a node n, $h(n)$ is the heuristic to estimate the distance from node n to the goal, and $f(n)$ is the total cost for node n. Specifically, $g(n)$ is the cost along the path to n using the cost metrics described in Section 5.3.5, and $h(n)$ is the number of remaining general axioms in the current node n.

The function $f(n)$ mainly computes the cost of applying some absorption to one general axiom. Note that each absorption has been preset to some weight, and the final cost is divided by the weight of the applied absorption. Then, the weight of one absorption is inverse proportional to the final cost, hence, a higher weight of some absorption gives this absorption certain priority in planning.

The heuristic function $h(n)$ is deliberately designed so as not to overestimate the cost of reaching the goal, in other words, $h(n)$ must be *admissible* for A* in order to be *optimal*. In this way, the heuristic avoids finding a suboptimal goal with the A* algorithm. In our implementation, $h(n)$ estimates that the cost to reach the goal is the number of remaining general axioms. However, the actual cost to absorb all these axioms (i.e., to reach the goal) will be greater than this estimation because many other factors will be considered other than the number of remaining general axioms, hence, the actual cost is underestimated by $h(n)$. Consequently, the heuristic function $h(n)$ used in our implementation is admissible, which also guarantees that the algorithm is complete.

Complexity We have seen in Section 2.2.3 that the worst-case computational complexity of classical planning is quite high. In our implementation, the planning operators are given in advance and have both negative preconditions and negative effects, hence the complexity of planning in our problem is at most PSPACE as shown in Table 4. However a lower complexity can generally be obtained in domain-specific planning algorithms. In our case, the number of nodes expanded is exponential in the length of the solution in the worst case. Note that the length of the solution is shortened to a few steps in scalable planning. Furthermore, scalable planning can help to enhance runtime performance by reducing the size of the search space[2].

[2] A* in principle is more problematic with its memory usage, which fortunately is not a problem in scalable planning.

6.1.2 The $\mathcal{ALCI}$ DL Reasoner

We continue by presenting a DL reasoner supporting KBs of the expressiveness $\mathcal{ALCI}$. At this point, preprocessors, such as simplification and normalization, become redundant because the KB will have been preprocessed by the planner.

This reasoner, implemented in a straightforward way, has only limited optimizations built in[3], such as (equality and subset) blocking strategies, to enhance its performance. Additionally, it is only capable of deciding the satisfiability of concepts. As a research prototype, this reasoner can be extensively used for experiments on newly designed techniques, but it is not comparable to existing popular reasoners, such as Racer (Haarslev and Möller [2001b]), FaCT++ (Tsarkov and Horrocks [2006]), and Pellet (Sirin and Parsia [2006]), in terms of reasoning services or efficiency.

The implementation of tableau expansion rules for $\mathcal{ALCI}$ follows Baader et al. [2003], together with a special rule to expand conjunctive axioms in $\mathcal{T}_u^{\sqcap}$. Because the classical planner asks for conjunctive absorptions, this underlying DL reasoner used the conjunctive expansion rule instead of the binary expansion rule. As reported earlier, conjunctive axioms take up a considerable amount of time in subset testing, and this reasoner implemented the algorithm in Yellin [1992] to minimize this kind of overhead.

6.2 Empirical Studies

This section presents some representative experimental results and analyses. We categorize these experiments according to the type of input KBs, which may be domain specific ontologies or synthetic ontologies used for testing only.

Environment Unless otherwise specified, all the empirical data is collected through the environment described here. The experiments are conducted under Windows on a standard PC: Dual Core (2.40 GHz) Pentium processor and 3 GB of physical memory. To ensure impartial comparisons, we use the latest releases (at the time of writing this thesis) of influential DL reasoners such as RacerPro 1.9.2 Beta, Pellet 1.5.2, and FaCT++ 1.1.11. All reasoners, if ever used in the testing, are allocated physical memory up to 1.2 GB if possible. To keep the data consistent, every testing guarantees at least 5 independent runs of all the systems.

Although we also evaluate other factors of the system performance, most of the data is for runtime

[3]A complete list of optimizations for DL reasoners is available at: http://dl.kr.org/dig/optimisations.html

performance, which is given in seconds in general. Sometimes the system issues a timeout (TO), which may be set to different values from one ontology to another, to terminate the testing. Reasoners might issue an *out of memory* message, denoted by **OOM**, if they cannot process the input ontology after consuming the allocated memory. The "—" indicates that the particular reasoner could not process the input knowledge base.

Note that PAR employs the following absorptions in its planner: concept absorptions (**CA**)[4], inverse role based absorption (**IRBA**), and conjunctive absorption (**CJA**). Domain and range absorptions are actually replaced by **IRBA**; equivalence axioms are usually put into $\mathcal{T}_e$ without further absorptions, however, sometimes they are transformed into inclusions to evaluate the performance of the planner. When the number of general axioms in PAR is used in the experiments, axioms are counted after preprocessing by PAR (Section 6.1.1), thus it may differ from other reasoners. If this number is not available in other reasoners due to some reason, then it is marked by a "*".

6.2.1 DM Ontologies

We have tested a series of ontologies named DM (Ben-David et al. [2007]), which model the domain of bounded model checking. There also exist two digits in the names of these ontologies, for example DM5-10, separated by a dash. The first digit indicates the size of the model in terms of a "cell" that contains 17 state-variables, i.e., DM5-10 has "5 cells" including 85 state-variables. The second digit in DM5-10 such as 10 stands for the bound. The larger the size and the bound, the harder the ontology. These ontologies are listed in an increasing order of the hardness: DM5-5, DM5-10, DM5-15, DM16-5 and DM16-10.

During testing the DM ontologies, we compared the absorption effectiveness and reasoning performance of the planner and arbitrary absorptions. A timeout (TO) of 1 000 seconds was used, as is seen in Table 9. Runtime performance of RacerPro, Pellet and FaCT++ are based solely on concept satisfiability tests, i.e., all other processes, such as loading and preprocessing, are excluded from the time calculation.

As expected, when the planner of PAR is substituted by any other single absorption techniques, the performance deteriorates substantially. For DM ontologies, the number of axioms tends to dominate the time for reasoning. For example, PAR runs fastest when all axioms are absorbed, i.e., the number of axioms becomes 0. Notwithstanding, reasoning performance is not totally determined by the number of general axioms, which can be seen from the experimental results as well. For

[4]Concept absorptions are in fact formulated as {**CAP**, **CAN**}, as depicted in the framework.

	DM5-5	DM5-10	DM5-15	DM16-5	DM16-10
PAR with the planner	0.10/0	0.11/0	0.20/0	1.72/0	2.92/0
PAR with **CA**	14.3/110	16.3/110	16.5/110	TO/352	TO/325
PAR with **CJA**	22.9/170	36.3/170	64.5/170	TO/544	TO/544
PAR with **IRBA**	10.5/130	14.2/129	30.9/129	350.6/394	444.1/409
RacerPro	TO	TO	TO	TO	TO
Pellet	0.25	0.27	0.44	1.53	1.67
FaCT++	0	0	0.02	0.08	0.08

Table 9: Tbox Coherence Check / # Axioms Unabsorbed. The time used for planning range from 0.4 seconds (for DM 5-5) to 3.4 seconds (for DM16-10).

instance, PAR using conjunctive absorption (**CJA**) leaves the same number of axioms unabsorbed (170) for ontologies DM5-5, DM5-10 and DM5-15, but the runtime for reasoning about the three ontologies is quite different.

For DM ontologies, RacerPro seems incapable of checking their concepts satisfiability within a span of 1 000 seconds. Pellet and FaCT++, on the contrary, respond rapidly. Given Table 9, it may be argued that the runtime performance of PAR using the planner suggest that the planning approach is not very competitive. However, the fact that PAR with few optimizations other than a planning system for axiom absorptions is comparable to Pellet (and far better than RacerPro) for DM ontologies already implies the significance of such an absorption planner.

6.2.2 More Realistic Ontologies

Ontologies are nowadays popular in many fields, especially in the biomedical domain. The experiments in this section focus on several ontologies designed for different purposes. Note that only some representative ontologies are selected because the current system implementation only covers ontologies within the expressiveness of $\mathcal{ALCI}$.

Other than to show how effective and efficient planning is, experiments also demonstrate that axiom absorption does not always benefit from classical planning, just like any other optimizations. Such fact does not contradict the purpose of our work in any aspect, as shown in our analyses in this section.

BCS Ontology

Areces et al. [1999] described an $\mathcal{ALC}$ ontology, the basic call services (BCS), as the formal specification of telephone feature interactions. It has been shown that one subscriber of BCS adds an exponential number of new interactions. We observed that BCS 5, indicating five subscribers, cannot be dealt with by almost all existing DL reasoners within a few hours except RacerPro 1.9.2 Beta.

	# General Axioms	Planning Time	Unabsorbed Axiom	Coherence Check Time
I	45	0.156	5	TO
II	2330	39.6	5	TO

Table 10: Experiments with BCS 5. TO = 1 000 seconds, and all times are in seconds. RacerPro: 75.453 seconds, Pellet: TO, and FaCT++: TO.

The data in Table 10 indicates the effectiveness of applying absorption planning to BCS 5. In the first case (I), only general axioms are fed into the planner, while in case II, equivalence axioms are transformed into inclusions, then all the axioms are fed into the planner.

It can be observed that absorptions do not aid in reasoning for BCS 5 because it is still difficult to check the coherence of the KB even if most axioms have been resolved. We conjecture that the remaining 5 axioms greatly contribute to the difficulty of reasoning. These axioms have the form $\top \sqsubseteq A_1 \sqcup \ldots \sqcup A_n$, where every $\neg A_i, 1 \leq i \leq n$ occurs on the LHS of some unfoldable TBoxes. Hence, such axioms are not amenable to absorptions. Table 10 also suggests that the proposed *scalable planning* is feasible and practical. For case II, the planner spends less than 1 minute on more than 2 000 axioms, showing its scalability to a large quantity of axioms (cf. Section 5.3.4).

BFO Ontology

Basic Formal Ontology (BFO) is a neat $\mathcal{ALC}$ ontology first presented in [Grenon et al., 2004]. In the experiments, BFO (Version 1.1) was used with the results shown in Table 11.

	# Axioms	Planning Time	# Unabsorbed	Check Time	$\sum$ Time
CAP	42	—	35	0.160	0.160
I	42	0.047	7	0.094	0.141
II	103	0.266	7	0.078	0.324

Table 11: Experiments with BFO. Times in seconds.

Table 10 shows that if only concept absorptions (**CAP**) are used, most axioms cannot be absorbed, which slightly degrades the runtime performance compared to the cases where planning is used. In the experiments, two different strategies of dealing with equivalence axioms are employed.

In case I, equivalence axioms are directly put into an unfoldable TBox so that none of them needs to be further absorbed. Following the discussions in Section 3.2.7, case II alternatively converts each equivalence axiom to two inclusions, and an additional operator that is used to "merge" inclusions into equivalences are introduced in planning. In the first case, 35 axioms are absorbed by using concept absorptions **CAP**, while both binary absorption (**BA**) and concept absorption (**CAN**) are used in the second case.

In both cases, seven axioms are left unabsorbed, which indicates that the newly introduced operator is actually ineffective for the BFO ontology because it is never encoded in any solution plan but increases the overhead for planning. As a matter of fact, in the first case, the planning is merely as effective as the single absorption **CAP**. Clearly, for ontologies like BFO, where axioms are not complicated, planning does *not* show any advantage over predetermined applications of absorption techniques.

Let us further analyze the reason why such axioms fail absorptions. Though all these axioms are simply of the form $\top \sqsubseteq \neg A_1 \sqcup \neg A_2$, where A_1 and A_2 are named concepts, such named concepts or their negations occur on the LHS of some other unfoldable TBoxes like $\mathcal{T}_u^{\neg A}$ or $\mathcal{T}_e$, which prevents these axioms from being absorbed in order to achieve unfoldability of the TBox. However, it can be observed from Table 11 that the remaining general axioms do not significantly affect reasoning about BFO because this ontology itself is not very challenging.

GALEN Ontology

The GALEN [Rector and Horrocks, 1997] ontology is considered a benchmark that is difficult for sophisticated tableau-based[5] reasoners. We show in Table 12 the experimental results of the GALEN ontology[6]. The original full GALEN ontology is quite difficult and cannot be classified by existing DL reasoners. However, it does not contain any general axiom, thus absorptions are not needed at all. An old version of full GALEN, named GALEN_Original, is also used for DL reasoners. A simplified version of GALEN_Original by removing some axioms has been frequently tested by DL reasoners as well, which is named GALEN_Simp in Table 12.

[5]Some non-tableau-based reasoners, however, sometimes use GALEN as the benchmark to show their advantages over traditional tableau-based systems as well.

[6]Available at http://www.opengalen.org

KB	DL	Size	Plan	# Axioms	Time
Full GALEN	$\mathcal{ALEHIF}$	23 136 Classes 950 Object Properties OWL File: 20.1 MB	—	Total: 0	0
GALEN_Original	$\mathcal{ALEHIF}$	2 748 Classes 413 Object Properties OWL File: 1.58 MB	**IRBA**: 111 **CAP**: 244	Total: 363 Remain: 8	2.09
GALEN_Simp	$\mathcal{ALEHIF}$	2 748 Classes 413 Object Properties OWL File: 1.49 MB	**IRBA**: 105 **CAP**: 244	Total: 357 Remain: 8	2.11

Table 12: The GALEN Ontology. The column "Plan" shows the solution, and the column "Time" indicates runtime for planning in seconds.

Table 12 shows that all three versions of the GALEN ontology have the expressiveness of $\mathcal{ALEHIF}$. On the one hand, PAR can apply absorptions freely to these KBs, it cannot reason about them because it only supports KBs with expressiveness $\mathcal{ALCI}$ or less. On the other hand, it has been shown that PAR is able to absorb most general axioms within 3 seconds, and the remaining 8 axioms are the same for both GALEN_Simp and GALEN_Original. Through further analyses, we found that these 8 axioms actually could be absorbed by breaking certain equivalence axioms, similar to what has been implemented in Racer. However, in our implemented system PAR, this is not achievable due to the offline requirement of classical planning.

Other Ontologies

We have seen previously the BFO ontology and the GALEN ontology, both of which are from the biomedical domain. In this section, we show our experiences with some other biomedical ontologies obtained from two major on-line ontology repositories: the Open Biomedical Ontologies[7] (OBO) and the Protégé Ontology Library[8].

Nearly all the ontologies (over 50 ontologies) in the above two repositories have been tested. Since the expressiveness of a large amount of these ontologies is outside the scope of $\mathcal{ALCI}$, we only provide a qualitative analysis based on their axioms. These ontologies can be categorized into the following three groups:

1. Ontologies do not have any general axioms. Most ontologies fall in this category, hence there

[7]Available at http://www.berkeleybop.org/ontologies
[8]Available at http://protegewiki.stanford.edu/index.php/Protege_Ontology_Library

is no need to use any absorptions on them. Typical ontologies include "Biological Process", "Human Disease", "National Cancer Institute Thesaurus (NCI)", "Foundational Model of Anatomy (subset) (FMA_Lite)" and so on.

2. Ontologies have several but trivial general axioms. Ontologies in this group can be trivially absorbed by one absorption alone (in most cases concept absorption **CAP** suffices), that is, the planner of PAR always uses only one operator. For such ontologies, we are unable to gain performance w.r.t. axiom absorption. "Gene Regulation", "Sequence", "Biochemical Reaction" and some more ontologies are of this type.

3. Ontologies have some general axioms, but the axioms do not affect the reasoning significantly. Certain ontologies contain general axioms that can not be absorbed by any existing absorptions. For instance, the "Molecule-Complex-2.0" ontology (with complex concept definitions in version 2.0) has two axioms of the form $\top \sqsubseteq \neg A \sqcup \neg B$, where both A and B are defined in an unfoldable TBox $\mathcal{T}_u^A$, hence such axioms can not be absorbed by absorptions available in PAR[9]. Our observation is that the existence of these unabsorbed axioms hardly affects the difficulty of the tableau-based reasoning.

To sum up, we find that compared to some predetermined application of absorptions, planning of axiom absorptions does not show great advantages for absorptions on ontologies from OBO and the Protégé Ontology Library. The reasons, as have been discussed, are closely related to the ontologies themselves: either they have only trivial (or even not any) general axioms or their general axioms will not significantly affect concept satisfiability tests.

6.2.3 Synthetic Ontologies

The planner does not significantly improve reasoning for ontologies where general axioms do not matter significantly in the reasoning, for example, axioms are simple or of a small quantity. For the experimental analysis, it is reasonable to synthesize ontologies featuring relatively complicated general axioms. Here, a series of knowledge bases, all of which base themselves on the same paradigm (cf. Figure 18) with only distinct replications of that paradigm, i.e., only the role names and concept names in each axiom are changed correspondingly in each replication.

The pattern for constructing axioms can be called the *Absorb or Death* test case. Intuitively, every

[9]These axioms *may* be absorbed by the enhanced concept absorption (**ECA**), which, however, is only available in an online planner.

I. $\top \sqsubseteq \neg A \sqcup \neg B \sqcup \neg C \sqcup \exists R.\exists R^{-}.D$

II. $\top \sqsubseteq A \sqcup \exists R.A \sqcup \exists R.\exists R.\forall R.C \sqcup \exists R.D \sqcup \exists R.E \sqcup \exists R.F$

III. $\top \sqsubseteq B \sqcup \exists R.B \sqcup \exists R.\exists R.\forall R.A \sqcup \exists R.D \sqcup \exists R.E \sqcup \exists R.F$

IV. $\top \sqsubseteq C \sqcup \forall R.\neg D \sqcup \exists R.\exists R.C$

Figure 18: A Pattern for General Axioms

axiom can be absorbed via concept absorptions without considering the whole KB. Unfortunately, the restriction that concept absorptions cannot absorb into a concept name and its negation at the same time applies to this small KB. Hence, some of these axioms cannot be absorbed by concept absorptions. If some axiom is not absorbed, it makes reasoning much harder due to its high degree of non-determinism. Thus, this example is given the name "absorb or death". A number of KBs can be synthesized by replicating N copies of this schema with a name $SampleN$. For instance, KB Sample5 has 5 replications of the pattern totaling to 20 axioms. Note that concept names will be automatically replaced by fresh ones for each replication.

In Table 13, the runtime performance really reflects the absorption effectiveness of all reasoners because for concept satisfiability tests the reasoners find these KBs difficult only if some axioms cannot be absorbed (that is, the "death" of the reasoners).

The planner can completely absorb axioms of the patterns shown in Figure 19. As long as the replications follow this pattern, all the axioms can undoubtedly be absorbed by PAR using the planner. In the phase of concept satisfiability tests, lazy unfolding rarely occurs so that PAR achieves the result effortless.

I. $A \sqcap B \sqcap C \sqsubseteq \exists R.\exists R^{-}.D$

II. $\neg A \sqsubseteq \exists R.A \sqcup \exists R.\exists R.\forall R.C \sqcup \exists R.D \sqcup \exists R.E \sqcup \exists R.F$

III. $\neg B \sqsubseteq \exists R.B \sqcup \exists R.\exists R.\forall R.A \sqcup \exists R.D \sqcup \exists R.E \sqcup \exists R.F$

IV. $D \sqsubseteq \forall R^{-}.(C \sqcup \exists R.\exists R.C)$

Figure 19: Planning Absorption on the Pattern (cf. Figure 18)

KB Name	RacerPro	Pellet	FaCT++	PAR with the planner
Sample1	0.031/1	0.016/*	0/*	0/0
Sample2	0.063/2	**OOM**	0/*	0/0
Sample3	1.875/3	—	0/*	0.015/0
Sample4	154.2/4	—	0/*	0/0
Sample5	**OOM**	—	0/*	0.016/0
Sample6	—	—	0/*	0/0
Sample10	—	—	0.016/*	0/0
Sample30	—	—	0/*	0.016/0
Sample50	—	—	0.016/*	0.016/0
Sample100	—	—	0.016/*	0.094/0
Sample250	—	—	0.109/*	0.25/0

Table 13: Concepts Satisfiability Test Time / # Axioms Unabsorbed. The planning time does not exceed 10 seconds for any sample KB.

From Table 13, it can be seen that both Pellet and RacerPro are incapable to deal with these general axioms. However, RacerPro outperforms Pellet due to its absorptions, which resolve almost all axioms except those of the type IV. On the contrary, Pellet can only handle the basic pattern, i.e., 4 axioms. Noticeably, the reasoning performance of FaCT++ remains nearly unaffected even when the size of the ontologies grows gradually. We conjecture that Pellet possibly has a weak form of absorption preprocessing, and FaCT++, in contrast to Pellet, may have its own strategy such as ordering to organize different kinds of absorptions.

6.2.4 Experiments on Some Components

The framework adapts the classical planning to bypass the scalability issue. This section then tests in practice if such adaptation is more efficient compared to ordinary planning. Cost mapping is essential in planning since it projects approximately the real cost (the hardness for reasoning) of a KB onto manageable cost. Experiments regarding the cost mapping are also reported in this section.

Experiments on Scalable Planning

In Section 5.3.4, the idea of scalable planning was presented to allow for efficient planning. Until this point, we have not yet witnessed the necessity of scalable planning. This section then presents

a few findings regarding our planning approaches, as described in Figure 20.

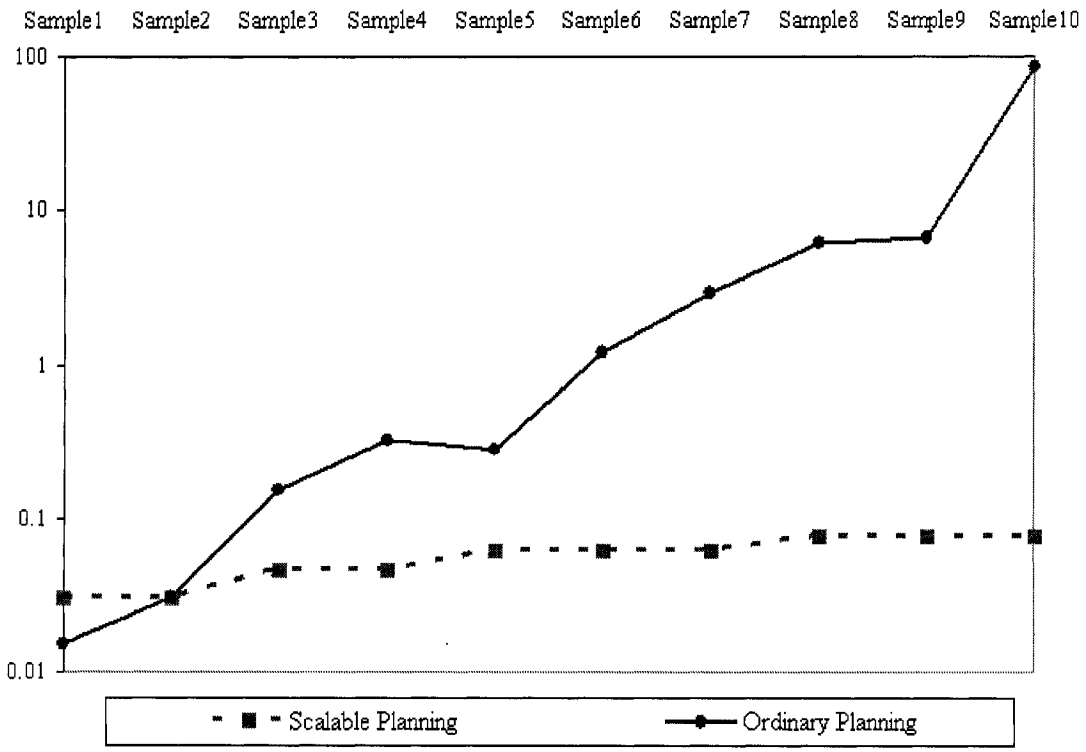

Figure 20: LogTime Performance of Two Planning Approaches on Synthetic Ontologies. The X axis represents runtime performance, and the Y axis shows the KB name.

Figure 20 represents times of planning using a logarithmic scale to reduce the large range of values: from less than 30 millisecond to more than 80 seconds. Note that for both planning approaches, all axioms can be completely resolved.

Obviously, ordinary planning outperforms scalable planning only in the first two test cases, where less than 10 axioms exist. When the difficulty of the DM ontologies increases, the runtime performance of scalable planning remains nearly stable, while the runtime performance of ordinary planning grows significantly. On average, all other things being equal, scalable planning improves by several orders of magnitude compared to ordinary planning.

An observable conclusion from experiments is that ordinary planning creates far more states than scalable planning. Especially for the DM ontologies, our experiments on DM5-5, the easiest one in the series of DM ontologies, demonstrated that ordinary planning absorbs only one tenth of all axioms, then it runs out of memory leading to a failure with more than 56 000 states created for this planning problem. For the same ontology, scalable planning takes only a few seconds to

completely absorb all axioms by building a state-transition system of less than 30 states. Again, though our test cases show no evidence that scalable planning will be less effective than ordinary planning, ontologies that can prove this point do exist. For instance, we can easily synthesize an ontology discussed in Section 5.3.4 to show that scalable planning may be incomplete in the sense that it might overlook some situations of absorptions.

Experiments on Cost Mapping

As we have seen in Section 5.3.5, cost mapping of the framework is only proposed as an estimation of the real cost, however, an intuition is that a more realistic projection from real cost onto estimated cost could lead to better solutions. The PAR system strictly follows the architecture of the proposed framework, hence the cost estimation is based on several factors: the weight of a particular absorption (Section 6.1.1), weights assigned to disjunctions and so on. Accordingly, our following experiments first evaluate the cost assignments to absorptions in our implementation, then other factors are considered as well.

Evaluating Absorptions Absorptions could be assigned different weights for cost accumulation because each of them leads to different behaviors of unfolding. In PAR, every absorption has a weight $\omega \in [0, 3]$. By default, all of them have an equal weight of 1.5. The reasons for such settings can be found in the experiments. A setting that enables a weight of 0 for some absorption might still produce a solution containing that absorption. In this case, the absorption is found indispensable in order to fully absorb the TBox, however, the solution has infinite cost along the path. Table 14 shows whether planning is affected when the weight varies in different cases.

It is impossible to evaluate all settings, as a result, only seven representative settings are given in Table 14. The data can be explained as follows. In setting 1, every absorption is assigned the same weight 1.5 (the default case), which leads to a total cost of 344.27. After planning, all 270 general axioms are absorbed, among which 16 axioms are absorbed by **CAP**, 100 axioms by **CJA** and the remaining 154 axioms by **IRBA**. Further, the DL reasoner spends 0.080 seconds to test the concepts satisfiability of a TBox like this.

Nevertheless, Table 14 describes exclusively DM5-5 ontology, for some ontology, weights may not affect planning if, for example, there exists only one way to completely absorb that ontology because the planner will eventually select that solution in order to achieve the goal. The synthetic ontologies described in Figure 18 exactly fall into this category. By observation, a few conclusions

No.	Weights ω Assigned to Absorptions				Cost	# Absorbed Axioms Distribution				Time
	CAP	**CAN**	**CJA**	**IRBA**		**CAP**	**CAN**	**CJA**	**IRBA**	
1	1.5	1.5	1.5	1.5	344.27	16	0	100	154	0.080
2	1.0	1.0	1.2	1.5	391.04	16	0	100	154	0.094
3	1.5	1.0	2.0	1.0	404.77	16	0	100	154	0.094
4	1.0	2.5	1.0	1.0	415.60	0	190	0	80	0.910
5	0.5	1.0	0.5	2.0	528.31	5	85	40	140	0.984
6	1.0	1.0	3.0	0.5	644.67	0	138	100	32	1.969
7	2.0	1.0	1.5	2.5	262.76	16	0	100	154	0.078

Table 14: Absorptions with Varying Weights on DM5-5 (with 270 general axioms). Cost refers to the final numerical value along the path of the solution; times (in seconds) in the last column is for concept satisfiability tests.

can be drawn regarding the weight assignment to absorptions.

- Minor differences in assigning weights to different absorptions will not significantly affect planning, as in settings {1, 2, 3}. The costs of these three setting are close to each other, so is the reasoning runtime performance. By contrast, planning is very likely to be affected if one absorption has a weight several times (>2) the weight of another one, as can be seen in settings {4, 5, 6}.

- The default setting, i.e., all absorptions have the same weight, is *probably* the safest practice for all ontologies, but may not result in the best setting w.r.t. a specific ontology. For DM5-5, setting 7 is slightly better w.r.t. runtime performance than setting 1. However, the best setting depends on the input, thus it is hard to determine the best setting prior to any empirical studies over the input.

- **CAN** (negative concept absorption) tends to degrade runtime performance if assigned a higher priority. Settings {4, 5, 6}, which degrade runtime performance by one order of magnitude compared to settings {1, 2, 3, 7}, have a large number of axioms produced by **CAN**. However, we argue that this does *not* imply that **CAN** should always be given a lower weight than **CAP** due to their polarity, since such choices really depend on how frequently concept names or negated concept names will be unfolded during reasoning.

  The observations do not indicate that **CJA** is a relatively "expensive" absorption due to the subset testing of conjunctive axioms. The reason is that conjunctive axioms are rarely triggered during reasoning for DM ontologies (see Section 4.1.2), which in turn proves that conjunctive

absorption is favored in absorbing DM ontologies.

- Cost mapping seems to be reasonable in our settings. As already discussed, the chosen cost mapping is aimed to project the hardness of absorbed TBoxes onto path cost in order to guide (possibly optimal) planning. The total cost (solution cost) is proportional to the runtime performance of reasoning (hardness of TBoxes), hence we conclude that the cost implementation of PAR seems to be effective.

Evaluating Weights of Disjunctions It should be noted that every implementation of the framework has its own unique consideration of evaluation factors. For example, in PAR, factors that contribute to the total cost includes the number of disjunctions in absorbed axioms, the choice of (negated) concept names in absorptions, and so on. A discussion of all factors is trivial and unnecessary, instead, this section selects the dominant factor, disjunctions, for evaluation. It is clear that disjunctions (thus non-determinism) is the motivation for axiom absorption, thus this selection is plausible. Figure 21 depicts the evaluation of DM5-5.

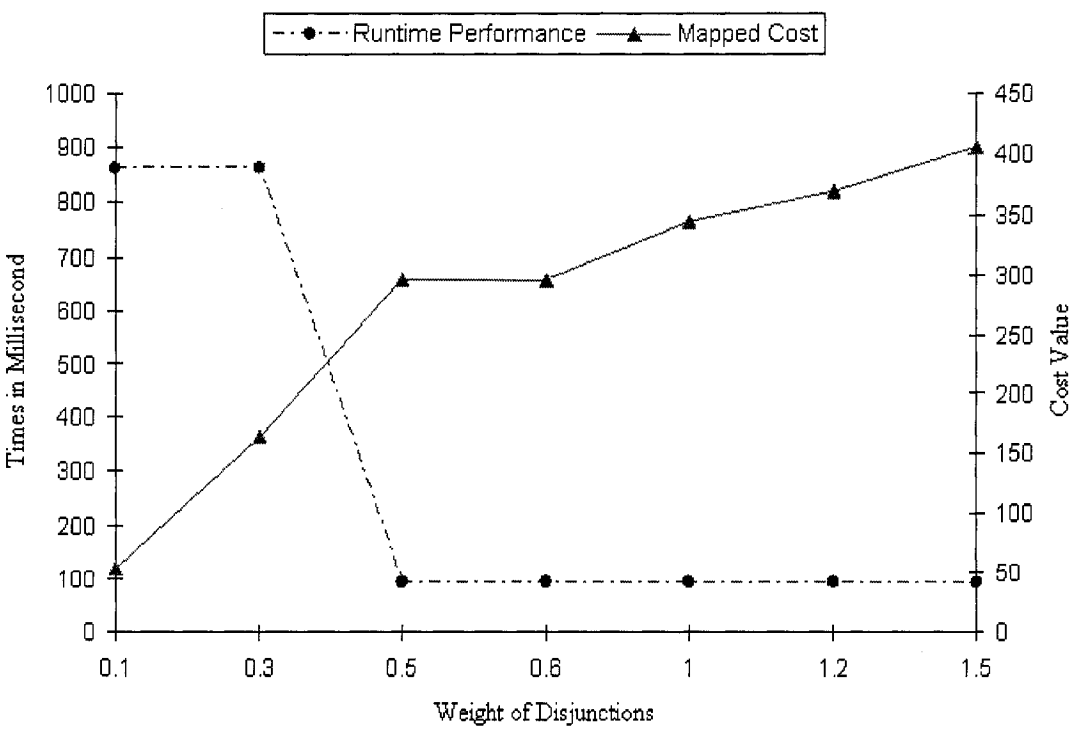

Figure 21: Evaluation of Weights of Disjunctions.

Note that in Figure 21 the dotted line (the hardness of the TBox) represents the runtime performance of reasoning over DM5-5 , while the straight line (the estimated cost for planning) reflects the implemented costs in PAR. There are two Y axes. The left Y axis indicates the runtime performance of DM5-5 in milliseconds, while the right one represents the virtual cost value produced in planning. The X axis shows the different weights assigned to disjunctions in different experiments.

The selection of weights can be simplified as the following statement: *weights should help to map the estimated cost as closely onto the actual cost as possible.* Here, the actual cost is considered as the hardness of absorbed TBox because a better absorption is expected to result in a simpler TBox for reasoning. Further observations from Figure 21 reveal that the actual cost tends to saturate (remains stable) when the weight assigned to disjunctions are greater than 0.5, while mapped cost increases slightly after its significant increase from 0.1 to 0.5. Bearing the selection criteria in mind, it is easy to select value "0.5" as the weight. However, it does no harm to select some values greater than 0.5 as long as the gap between the actual cost and the mapped cost is not large. In PAR, the default weight of disjunctions is set to "1".

Chapter 7

Conclusions and Future Work

7.1 Conclusions and Discussions

Description logic reasoners adopt a wide variety of optimization techniques to accommodate themselves to emerging large and complex ontologies such as FMA and GALEN. It is not hard to find ontologies with a considerable amount of complex axioms that are very likely, once fed into present reasoners, to degrade the runtime performance of the reasoners by several orders of magnitude. For one thing, most DL reasoners use absorptions in such a fixed or predetermined fashion that this may lead to an incomplete absorptions of all axioms. For another thing, if all axioms are fully absorbed, there is no explanation toward why this particular absorbed TBox is obtained instead of other possibilities that could have been obtained due to the intrinsic *nondeterminism* of absorptions themselves.

One might argue that a novel absorption could be our solution, however, as we conjecture, some axioms may be *inherently unabsorbable*, i.e., they cannot be absorbed by any (at least already known) absorptions. Taking above comments into consideration, it would be interesting to make absorptions *deterministic* so that DL reasoners show a preference toward some absorbed TBox which they *consider* to be easier for reasoning services. This potential idea propels us to be realistic to take advantage of existing absorptions rather than finding the silver bullet.

This thesis then showed the first attempt to build a framework incorporating adapted general AI planning to guide absorption techniques. The proposed framework mainly defines how planning and heuristics are to be designed in order to provide a solution containing the types of absorptions with a preferred sequence to apply. The solution is repeatable for the same input and settings because

nondeterminism is thus removed due to the heuristics.

We also implemented the framework as a research prototype for evaluation. Extensive experiments have been carried out to show how the underlying ideas work on ontologies of different characteristics such as the domain modeled by ontology, the size of the ontology, the number of general axioms and so on. On the one hand, it can be seen that some ontologies such as BCS 5 and BFO are not completely absorbable, which makes sense because the framework is totally based on some existing absorptions. On the other hand, experimental results imply that planning of axiom absorption tends to lead to a more effective absorbed TBox compared to usual application of absorptions, at the cost of some overhead in planning itself.

FaCT++, a very efficient and highly optimized DL reasoner, seems to have a quite efficient absorption module because it is the only existing reasoner that can survive the so-called "*Absorb or Death*" ontologies (cf. Section 6.2.3). An explanation toward this phenomenon is that FaCT++ has its own strategy (with a lot of hard-coded heuristics) to organize its implemented absorption techniques, which shares the identical idea with planning, i.e., arrange absorptions in such a heuristic way that the generated TBox meets certain preset goals.

Racer also implements a very effective absorption preprocessor, which utilizes nearly all known absorptions. Racer seems to have no mechanism to arrange absorptions systematically though it uses certain heuristics that might occasionally absorb axioms better than our system PAR. For instance, Racer is capable of absorbing all general axioms of the simplified GALEN ontology (cf. Section 6.2.2), while PAR leaves seven general axioms unabsorbed. For this ontology, Racer transforms equivalence axioms into inclusion axioms only when there are still general axioms left after all other absorptions have been tried. In this case, only equivalence axioms whose LHS has occurred in the remaining general axioms will be transformed. The transformation of equivalence axioms on demand is a feature not achievable in PAR because of the offline restriction.

The experimental results, though not always satisfactory in terms of performance gain, demonstrate that the framework we have proposed in this thesis is a good start point for future refinement of optimizations about DL reasoning. This framework can also be viewed as a generalization/categorization of currently known absorption techniques, which thoroughly studies the underlying ideas of different kinds of absorptions. At the same time, this framework also reveals its drawbacks in both theoretic and practical aspects, as can be seen from the empirical studies.

- Classical planning restricts the types of absorption that can be used. For example, enhanced

concept absorptions and enhanced conjunctive absorptions are excluded from this framework. It is the offline requirement that limits the use of these absorptions, hence it is worth considering to ease this restriction by extending the framework.

- Experiments witness no significant improvement when planning is applied to KBs with a small number of uncomplicated axioms. For these ontologies, the overhead of applying planning may neutralize the insignificant achievement the system obtains in absorptions.

The first problem mentioned above can be extended in the future research, as discussed in the next section; for the second one, we argue that a more sophisticated implementation of the framework on existing DL reasoners can improve reasoning performance for current and for emerging ontologies that have difficult general axioms.

7.2 Future Work

There exist a number of ways to extend or improve the framework presented in the thesis, however, most of them require further investigation before they can be directly applied to the framework. Some of the interesting extensions and/or improvement strategies are named in this section as future work.

7.2.1 Optimizing the DL Reasoner

The experiments demonstrate that for some ontologies our system PAR, which even lacks sophisticated optimizations, is comparable to existing highly optimized DL reasoners in some aspects, but it would be more interesting to examine the results if the DL reasoner in PAR were highly optimized. Given the planner as an processor of axioms, an optimized reasoner could be more practical for future ontologies. Another option can be to incorporate such a preprocessor into present DL reasoners, particularly the open source ones like Pellet and FaCT++. Armed with a more practical absorption module, the reasoners can expect to deal with general axioms in a more effective way for some KBs.

As the planner, which can serve as a typical preprocessor, is independent from specific DL reasoners, it can be implemented as a plug-in to Protégé[1]. Note that developers are expected to use

[1]Developing Protégé plug-ins: http://protege.stanford.edu/doc/dev.html

the Protégé-OWL API and Java. After implementing such a plug-in, DL reasoners supported by Protégé can use this plug-in when needed.

7.2.2 Improving Cost Mapping

Cost metrics presented in the framework may be sufficient for a prototype system like our implementation PAR to examine the effectiveness of the framework, but experiments, as can be seen in Section 6.2.4, suggest that a better setting of weights of all factors can lead to a more realistic cost mapping, thus the precision of mapping needs to be enhanced as well.

Our experiments, through trial and error, define a reasonable setting for some ontologies, however, this setting of cost mapping may not be optimal at all for other ontologies. This naturally inspires us to consider improving such settings for all ontologies through learning. Given a dataset of ontologies with their settings of cost mapping, the system can be trained to find a setting impartial to any specific ontologies. In the near future, we expect to see more ontologies that are amenable to the axiom planning system in order to maintain a training dataset in moderate size.

7.2.3 Online Planning

We have seen that the framework is based on the classical planning, which is restricted to offline planning. The consequence is that binary absorption, the enhanced concept absorption and the enhanced conjunctive absorption cannot be directly employed as operators in the planner. While binary absorption is replaced by conjunctive absorption in the classical planner, it should be noted that for some KBs binary absorption may be preferred, as discussed in Section 4.1.2. The enhanced concept absorption is actually resolution based, and the same methodology is used for extending conjunctive absorptions.

If all the absorptions need to be considered in the planner, online planning is usually required. Though a planner may not have to worry about all the details of the actual *dynamics*, at least, it needs to check online whether a solution plan remains valid and, if needed, to revise it or re-plan. The demanding requirement of online planning could also be indirectly handled, which we have initially attempted in our work. The lessons we learned is that online planning is also an advanced research topic in planning, thus the trade-off between the cost and the gains of applying online planning must be balanced for a practical system.

Bibliography

Carlos Areces, Wiet Bouma, and Maarten de Rijke. Feature interaction as a satisfiability problem. In *Proceedings of International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems 1999*, pages 339–346, Los Alamitos, California, October 1999. IEEE Computer Society.

Franz Baader, Bernhard Hollunder, Bernhard Nebel, Hans-Jürgen Profitlich, and Enrico Franconi. An emperical analysis of optimization techniques for terminological representation systems or: "making KRIS get a move on". *Applied Intelligence*, 4(2):109–132, 1994.

Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*, 2003. Cambridge University Press. ISBN 0-521-78176-0.

Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proceedings of International Joint Conference on Artificial Intelligence 2005*, pages 364–369. Professional Book Center, 2005.

Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In Ulrich Furbach and Natarajan Shankar, editors, *Proceedings of International Joint Conference on Automated Reasoning 2006*, volume 4130 of *LNAI*, pages 287–291. Springer-Verlag, 2006.

Sean Bechhofer, Raphael Volz, and Phillip W. Lord. Cooking the semantic web with the OWL API. In *Proceedings of International Semantic Web Conference 2003*, pages 659–675, 2003.

Shoham Ben-David, Richard J. Trefler, and Grant Weddell. Bounded model checking with description logic reasoning. In *Proceedings of International Conference on Automated Reasoning with Analytic Tableaux and Related Methods 2007*, pages 60–72. Springer, July 3-6 2007.

Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5): 34–43, May 2001.

Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research (JAIR)*, 17:229–264, 2002.

Mike Dean and Guus Schreiber, editors. *OWL Web Ontology Language Reference*, February 2004. Latest version available at http://www.w3.org/TR/owl-ref/.

Yu Ding and Volker Haarslev. Towards efficient reasoning for description logics with inverse roles. In *Proceedings of International Workshop on Description Logics 2005*, Edinburgh, Scotland, UK, July 26-28 2005.

Yu Ding, Volker Haarslev, and Jiewen Wu. A new mapping from $\mathcal{ALCI}$ to $\mathcal{ALC}$. In *Proceedings of International Workshop on Description Logics 2007*, pages 53–64, Italy, 8-10 June 2007.

Richard Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3/4):189–208, 1971.

Alfonso Gerevini and Derek Long. Plan constraints and preferences in PDDL3. Technical report, Department of Electronics for Automation, University of Brescia, Italy, August 2005.

Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice.* Morgan Kaufmann, 1st edition, 2004.

Pierre Grenon, Barry Smith, and Louis Goldberg. Biodynamic ontology: Applying BFO in the biomedical domain. In *Ontologies in Medicine - Proceedings of the Workshop on Medical Ontologies*, pages 20–38, 2004.

Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, June 1993. ISSN 1042-8143.

Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics*, 3(2):158–182, 2005.

Volker Haarslev and Ralf Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proceedings of International Joint Conference on Artificial Intelligence 2001*, pages 161–168, 2001a.

Volker Haarslev and Ralf Möller. Racer system description. In *Proceedings of International Joint Conference on Automated Reasoning 2001*, pages 701–706, London, UK, 2001b. Springer-Verlag.

Ian Horrocks. Implementation and optimization techniques. In *Baader et al. [2003]*, pages 306–346, 2003.

Ian Horrocks and Ulrike Sattler. A Tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 448–453, 2005.

Ian Horrocks and Stephan Tobies. Reasoning with axioms: Theory and practice. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning 2000*, pages 285–296, San Francisco, CA, 2000. Morgan Kaufmann Publishers.

Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In Harald Ganzinger, David McAllester, and Andrei Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR 1999)*, pages 161–180. Springer-Verlag, 1999.

Chih-Wei Hsu, Benjamin W. Wah, Ruoyun Huang, and Yixin Chen. New features in SGPlan for handling soft constraints and goal preferences in PDDL3.0. In *Fifth International Planning Competition, International Conference on Automated Planning and Scheduling (ICAPS 2006)*, pages 54–58, June 2006.

Alexander K. Hudek and Grant Weddell. Binary absorption in tableaux-based reasoning for description logics. In *Proceedings of International Workshop on Description Logics 2006*, Lake District, UK, 30 May-1 June 2006.

David C. Marr. Early processing of visual information. *Royal Society of London Philosophical Transactions Series B*, 275:483–519, October 1976.

Steven Minton, John L. Bresina, and Mark Drummond. Total-order and partial-order planning: A comparative analysis. *Journal of Artificial Intelligence Research*, 2:227–262, 1994.

Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: engineering an efficient SAT solver. In *Proceedings of the 38th conference on Design automation (DAC 2001)*, pages 530–535, New York, NY, USA, 2001. ACM Press. ISBN 1581132972.

Boris Motik, Rob Shearer, and Ian Horrocks. Optimized reasoning in description logics using hypertableaux. In Frank Pfenning, editor, *Proceedings of the 21st Conference on Automated Deduction (CADE 2007)*, volume 4603 of *LNAI*, pages 67–83, Bremen, Germany, July 2007. Springer.

Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43(2): 235–249, 1990.

Alan L. Rector and Ian Horrocks. Experience building a large, re-usable medical ontology using a description logic with transitivity and concept inclusions. In *Proceedings of Workshop on Ontological Engineering, AAAI Spring Symposium*, pages 100–107, Stanford CA, January 1997.

Cornelius Rosse and José L. V. Mejino. A reference ontology for biomedical informatics: the foundational model of anatomy. *Journal of Biomedical Informatics*, 36(6):478–500, 2003. ISSN 1532-0464.

Andrea Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994.

Klaus Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of International Joint Conference on Artificial Intelligence 1991*, pages 466–471, 1991.

Manfred Schmidt-Schauss and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

Evren Sirin and Bijan Parsia. Pellet system description. In Bijan Parsia, Ulrike Sattler, and David Toman, editors, *Proceedings of International Workshop on Description Logics 2006*, Lake District, UK, 30 May-1 June 2006.

Evren Sirin, Bernardo Cuenca Grau, and Bijan Parsia. From wine to water: Optimizing description logic reasoning for nominals. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning 2006*, pages 90–99. AAAI Press, 2006.

Dmitry Tsarkov and Ian Horrocks. Efficient reasoning with range and domain constraints. In *Proceedings of International Workshop on Description Logics 2004*, British Columbia, Canada, June 6-8 2004.

Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: System description. In *Proceedings of International Joint Conference on Automated Reasoning 2006*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006.

Dmitry Tsarkov, Ian Horrocks, and Peter F. Patel-Schneider. Optimizing terminological reasoning for expressive description logics. *Journal of Automated Reasoning*, 39(3):277–316, 2007. ISSN 0168-7433.

Jiewen Wu and Volker Haarslev. Planning of axiom absorptions. In *Proceedings of International Workshop on Description Logics 2008*, Dresden, Germany, May 13-16 2008.

Daniel M. Yellin. Algorithms for subset testing and finding maximal sets. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms 1992*, pages 386–392, Philadelphia, PA, USA, 1992. Society for Industrial and Applied Mathematics.

Ming Zuo. High performance absorption algorithms for terminological reasoning in description logics. Master's thesis, Concordia University, Montréal, Québec, Canada, September 2006.

Ming Zuo and Volker Haarslev. High performance absorption algorithms for terminological reasoning. In *Proceedings of International Workshop on Description Logics 2006*, Lake District, UK, 30 May-1 June 2006.

List of Abbreviations

ABox	Assertion Box
BA	Binary Absorption
CA	Concept Absorption
CAN	Negative Concept Absorption
CAP	Positive Concept Absorption
CJA	Conjunctive Absorption
CNF	Conjunctive Normal Form
DL	Description Logic
DNF	Disjunctive Normal Form
DRA	Domain and Range Absorptions
EA	Equivalence Absorption
ECA	Enhanced Concept Absorption
FOL	First Order Logic
GCI	General Concept Inclusion
IRBA	Inverse Role Based Absorption
KB	Knowledge Base
KR	Knowledge Representation
LHS	Left-hand Side
NNF	Negation Normal Form
OA	Nominal Absorption
OWL	Web Ontology Language
RHS	Right-hand Side
TBox	Terminology Box
UNA	Unique Name Assumption