Kernel Contraction in $\mathcal{EL}$

by

Zhiwei Liao
B.Sc., Peking University, 2012

Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Fall 2014

APPROVAL

Name: Zhiwei Liao

Degree: Master of Science

Title of Thesis: Kernel Contraction in $\mathcal{EL}$

Examining Committee: Dr. Ramesh Krishnamurti
Chair

Dr. James Delgrande,
Professor, Senior Supervisor

Dr. Eugenia Ternovska,
Associate Professor, Supervisor

Dr. Kay Wiese,
Associate Professor, Internal Examiner

Date Approved: November 13th, 2014

Partial Copyright Licence

Abstract

Belief contraction in description logics in a syntax-independent manner is an important and nontrivial problem for ontology management and DL communities. One approach for belief contractions is to take all minimal sets that imply a formula to be contracted and delete one formula from each est, which is also called kernel contraction. In this thesis we make an investigation of kernel contraction in the description logics $\mathcal{EL}$. We demonstrate that kernel contraction in $\mathcal{EL}$ is rational by proving some of its properties. We present different ways to generating kernel contractions. We introduce some novel approaches such as localization and specificity to decide the preference between different certain kernel contractions by exploiting the structure of an $\mathcal{EL}$ knowledge base and we give algorithms for computing them.

Contents

Approval **ii**

Partial Copyright License **iii**

Abstract **iv**

Contents **v**

1 Introduction **1**

2 Background and Related work **4**

2.1 Description logic 4

2.2 The description logic $\mathcal{EL}$ 6

2.3 Subsumption Problem in $\mathcal{EL}$ 9

2.4 AGM Postulates for Belief Contraction 11

2.5 Definition of Kernel Contraction in $\mathcal{EL}$ 12

3 Kernel Contraction in $\mathcal{EL}$ **15**

3.1 Generating kernels using axiom pinpointing 15

3.2 Algorithms for generating kernel contractions 16

3.3 Properties of kernel contraction in $\mathcal{EL}$ 21

3.4 Localization 24

3.5 Specificity 31

3.6 Connection with Hitting Set Problem and Set Cover Problem 34

3.6.1 Hitting Set Problem 34

3.6.2 Set Cover Problem 35

3.6.3 Connections between Kernel Contraction, Hitting Set Problem and Set Cover Problem 36

4 Conclusion and Future Work **39**

4.1 Thesis Summary 39

4.2 Recommendation for Future Work 40

4.3 Conclusion 41

Bibliography **42**

Chapter 1

Introduction

A long time ago, people believed that the earth's shape was a plane or disk. Later the concept of a spherical earth appeared, and then more and more evidence also support this concept. People replaced the flat earth model in their minds with the spherical earth. In order to do that, they needed to withdraw some of the old beliefs because one cannot believe in both flat earth and spherical earth at the same time.

This is a typical instance of a *belief revision* scenario: a person receives new information that makes one change his/her beliefs. In the principal case where the new information contradicts her initial belief state, this person needs to withdraw some of the old beliefs before he/she can accommodate the new information; he/she also needs to accept the consequences that might result from the interaction of the new information with the (remaining) old beliefs.

Belief revision [9] is a subarea of knowledge representation concerned with describing how an intelligent agent ought to change its beliefs about the world in the face of new and possibly conflicting information, which can be described as the process of rationally adding a certain belief φ to a belief set K.

Apart from belief revision, there is another type of belief change called *belief contraction* (or simply *contraction*), which can be described as the process of rationally removing a certain belief φ from a belief set K. Contraction typically occurs when an agent loses faith in φ and decides to give it up. Simply taking out φ from K however will not suffice since other sentences that are present in K may reproduce φ through logical closure. Consider for example the theory $K = Cn(\{p \to q, p, q\})$($Cn(A)$ represent the deductive closure generated by all the beliefs in A) and assume that we want to contract K by q. Then, not only do

we have to remove q from K, but we also need to give up (at least) one of $p \to q$ or p, for otherwise q will resurface via logical closure.

Various methods exist for constructing basic contraction. Kernel contraction is one of them. The basic idea of kernel contraction [14] is to select the beliefs to be discarded, in other words to base the operator of contraction on a selection among the elements of K that contribute to making it imply φ. A "kernel" is a minimal set of beliefs in a belief set such that you can derive the belief for contraction from that set. For example, if we want to contract $K = Cn(\{p \to q, p, q\})$ by q, the kernels would be $\{p \to q, p\}$ and $\{q\}$. From each of these two sets we can derive q and they are minimal (i.e. if you delete any belief in the set, we cannot derive q from that set any more). There could be a lot of kernels. In order to contract a belief, one would have to remove (at least) one belief from each kernel. So kernel contraction is the process of determining all kernels and removing one belief from each kernel. Clearly there are a lot of ways in general to carry out a kernel contraction and some of them could be better than the others in the sense of rationality. In this thesis, we focus on the belief contraction in $\mathcal{EL}$, which is a specific light-weight description logic.

Description logic (DL) is a family of formal knowledge representation languages. Basically, a DL models concepts, roles and individuals, and their relationships. Description logic is an important and active area in knowledge representation. It is of particular importance in providing a logical formalism for ontology languages like Web Ontology Language and the Semantic Web. The definition of DL will be included in next chapter.

$\mathcal{EL}$ is a light-weight DL using only three concept constructors: top concept ($\top$), conjunction ($\sqcap$), and existential restriction ($\exists r.C$). It is simple but still very important. It was designed to admit subsumption and classification in polynomial time, while still providing sufficient expressive power for life-science ontologies such as SNOMED.

The goal of this thesis is to study kernel contraction in $\mathcal{EL}$. We come up with various ways to select formulas from kernels to generate a kernel contraction, we also analyze and compare them. We exploit the structure of an $\mathcal{EL}$ knowledge base to find better kernel contraction by some new concepts such as localization and specificity. Also we link kernel contraction to the hitting set problem and the set cover problem so that we can apply some results about the hitting set problem and set cover problem into kernel contraction.

The remaining chapters are organized as follows. Chapter 2 introduces some background information and definitions of DL, especially the light-weight DL $\mathcal{EL}$. Basic definition of kernel contraction is also included in this chapter. After that, we discuss some related works

about $\mathcal{EL}$ and kernel contraction, respectively in chapter 2. Chapter 3 is the main part of this thesis. It presents our contribution to kernel contraction in $\mathcal{EL}$. In chapter 3, we introduce the definition of kernel contraction in $\mathcal{EL}$ and the algorithms to actually construct kernel contractions in $\mathcal{EL}$. We also analyze the properties of kernel contraction in $\mathcal{EL}$ and present different algorithms for the contructions. Whereas previous work on kernel contraction considered kernel contraction as a whole, in this part we consider the difference between any specific kernel contractions by presenting some novel concepts such as localization and specificity and algorithms for deciding a better kernel contractions. The connection between kernel contraction, hitting set problem and set cover problem is discussed in Chapper 3. Last but not least, chapter 4 discusses our contributions and limitation of our work, points out the direction for future work, and summarizes this thesis.

Chapter 2

Background and Related work

This chapter includes all the necessary background information and definitions before we present our study on kernel contraction in $\mathcal{EL}$. In this chapter, we first introduce the general concept of description logic and the definition of one of the most widely used description logics: $\mathcal{EL}$. Then we give the definition of kernel contraction in $\mathcal{EL}$. After that, recent progress about both kernel contraction and $\mathcal{EL}$ is discussed.

The chapter is divided into four sections. Section 2.1 is an introduction of general description logic. Section 2.2 introduces the definition of $\mathcal{EL}$ and $\mathcal{EL}^+$, presents some sample knowledge bases of $\mathcal{EL}$ and $\mathcal{EL}^+$. Section 2.3 discusses some related work about subsumption problem in $\mathcal{EL}$. Section 2.4 introduces the AGM postulates for belief contraction. Section 2.5 is the definition of kernel contraction in $\mathcal{EL}$.

2.1 Description logic

Description Logics [5] (DLs) are a well-investigated family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. They are employed in various application domains, such as natural language processing, configuration, and databases, but their most notable success so far is the adoption of the DL-based language OWL as a standard ontology language for the Semantic Web.

A DL models concepts, roles and individuals, and their relationships. In DL, there are three sorts of expressions that act like nouns and noun phrases in English :

- *Atomic concepts* are like basic category nouns, such as

$$Dog, Teenager, GraduateStudent$$

- *Roles* are like relational nouns, such as

$$: Age, : Parent, : AreaOfStudy$$

- *Constants* are like proper nouns, such as

$$JohnSmith, chair128$$

We use these three types of non-logical symbols and some of the logical symbols which are also called *concept constructors* to build the set of *concepts* (The expressivity of a particular DL is determined by which logical symbols are available in it). With concepts and connectives like $=$ and $\sqsubseteq$, we can build the set of *assertions*, which can be true or false. For example, $GraduateStudent \sqsubseteq Student$ says that $GraduateStudent$ is subsumed by $Student$, which is true.

In DL, with the set of concepts fixed, a distinction is drawn between the so-called TBox (terminological box) and the ABox (assertional box). In general, the TBox contains sentences describing concept hierarchies (i.e., relations between concepts) while the ABox contains ground sentences stating where in the hierarchy individuals belong (i.e., relations between constants and concepts).

A TBox can be used to introduce relations about concepts and roles. For example, the TBox

$$GraduateStudent \sqsubseteq Student$$

$$GraduateStudent \sqsubseteq \exists AreaOfStudy.\top$$

tells us that all graduate students are students and graduate students have their area of study.

In the ABox of a DL knowledge base, facts about a specific application situation can be stated, by introducing named individuals and relating them to concepts and roles. For example, the ABox

$$GraduateStudent(John)$$

$$AreaOfStudy(John, Mathematics)$$

tells us that John is a graduate student and his area of study is mathematics.

There are infinite different DLs. In order to classify and discuss them efficiently, all DLs are divided into different groups of DLs, according roughly to the set of concept constructors allowed in that DL (which is independent to the Tbox and the Abox of that DL). This is because the expressivity of a DL depends on the set of concept constructors. For example, the DL $\mathcal{AL}$, which allows atomic negation, concept conjunction, universal restrictions and limited existential quantification; $\mathcal{FL}$ allows concept conjunction, universal restrictions, limited existential quantification and role restriction; and $\mathcal{EL}$ allows concept conjunction and existential restrictions (of full existential quantification). In this thesis, we will mainly discuss $\mathcal{EL}$, which is a light-weight DL having many important applications.

2.2 The description logic $\mathcal{EL}$

Definition of $\mathcal{EL}$ [14]:

The set of concepts is the least set satisfying:

- $\top$ is a concept.
- Every atomic concept is a concept.
- If C and D are concepts, then $C \sqcap D$ is a concept.
- If C is a concept and r is a role, then $\exists r.C$ is a concept.

The semantics of $\mathcal{EL}$ is defined using the notion of an interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, which consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$. Interpretations of $\mathcal{EL}$ $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ are given by:

$$C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$$

$$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$$

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$$

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$

$$(\exists r.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} | \exists b.(a, b) \in r^{\mathcal{I}} \text{ and } b \in C^{\mathcal{I}}\}$$

A *general concept inclusion* (GCI) has the form where C and D are concepts:

$$C \sqsubseteq D$$

with the semantics:

$$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$$

and a *concept definition* has the form where A is an atomic concept and C is a concept:

$$A \equiv C$$

with the semantics:

$$A^{\mathcal{I}} = C^{\mathcal{I}}$$

The concept definition $A \equiv C$ is equivalent to the two GCIs $A \sqsubseteq C$, $C \sqsubseteq A$. For this reason, in the following we will consider only GCIs. A $\mathcal{EL}$ TBox is any finite set of GCIs.

The syntax and semantics of $\mathcal{EL}$ are summarized in the table below.

Name	Syntax	Semantics
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
top	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in r^{\mathcal{I}} \text{ and } b \in C^{\mathcal{I}}\}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
concept definition	$A \equiv C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$

Table 2.1: Syntax and semantics of $\mathcal{EL}$

We give a sample $\mathcal{EL}$ knowledge base called ANIMAL below.

The ANIMAL ABox :

Elephant(Tom), WhiteWhale(Lily), Lizard(Dave)

The ANIMAL TBox :

$$\text{Mammal} \sqsubseteq \text{Animal}$$
$$\text{Elephant} \sqsubseteq \text{Terrestrial} \sqcap \text{Mammal}$$

$$\text{Whale} \sqsubseteq \text{Aquatic} \sqcap \text{Mammal}$$
$$\text{Elephant} \sqsubseteq \exists\ \text{Color.Grey}$$
$$\text{Elephant} \sqsubseteq \exists\ \text{Feature.Longnose}$$
$$\exists\ \text{Feature.Longnose} \sqsubseteq \text{Animal}$$
$$\text{WhiteWhale} \sqsubseteq \text{Whale} \sqcap \exists\ \text{Color.White}$$
$$\text{Amphibious} \sqsubseteq \text{Animal}$$
$$\text{Lizard} \sqsubseteq \text{Amphibious}$$

The ABox above says that in ANIMAL, Tom is an elephant, Lily is a white whale, and Dave is a lizard. In the TBox we have that mammals are animals, elephants are terrestrial mammals, whales are aquatic mammals, elephants' color is grey, elephants have long noses as features, white whales' color is white, amphibious living things are animals, and lizards are animals.

Here we may also briefly introduce $\mathcal{EL}^+$, an extension of $\mathcal{EL}$ [8]. $\mathcal{EL}^+$ has the same set of concepts as $\mathcal{EL}$. $\mathcal{EL}^+$ TBox is a finite set of *general concept inclusions* (GCIs) and *role inclusions* (RIs), whose syntax is shown in the lower part of the table below.

Notation $*$ is used in RIs to represent the composition of two rules: If $(a, b) \in r_1$ and $(b, c) \in r_2$, then $(a, c) \in r_1 * r_2$.

For example,

$$father * father \sqsubseteq grandfather$$

is a RI which tells us if A is B's father and B is C's father then A is C's grandfather where A, B, C are any three people.

Name	Syntax	Semantics
atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
top	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a, b) \in r^{\mathcal{I}} \text{ and } b \in C^{\mathcal{I}}\}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
RI	$r_1 * \cdots * r_n \sqsubseteq s$	$r_1^{\mathcal{I}} * ... * r_n^{\mathcal{I}} \subseteq s^{\mathcal{I}}$

Table 2.2: Syntax and semantics of $\mathcal{EL}^+$

We give a sample $\mathcal{EL}^+$ called FAMILY below.

The FAMILY ABox :

father(Ray, David), mother(Ray, Joyce), father(David, Alex), mother(Joyce, Grace)

The FAMILY TBox :

$$father * father \sqsubseteq grandfather$$

$$mother * mother \sqsubseteq grandmother$$

$$father \sqsubseteq parent$$

$$mother \sqsubseteq parent$$

The ABox above says that in FAMILY, David is Ray's father, Joyce is Ray's mother, Alex is David's father, and Grace is Joyce's mother. In the TBox we have that one's father's father is one's grandfather, one's mother's mother is one's grandmother, one's father is one's parent, and one's mother is one's parent. And through deductive closure we can deduct something like Alex is Ray's grandfather and Grace is Ray's grandmother.

$\mathcal{EL}$ is simple but still a very important type of DL. It has enough expressivity [5] for many areas while its subsumption problem, which is to decide whether $C \sqsubseteq D$ is true where C and D are two concepts, can still be solved in polynomial time. SNOMED CT [2] is a well-known medical ontology, which comprises 380,000 concepts and is used as a standardized health care terminology in a variety of countries such as the US, Canada, and Australia. From the DL point of view, SNOMED CT is an $\mathcal{EL}$ KB as SNOMED CT contains only three concept constructors: conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$).

2.3 Subsumption Problem in $\mathcal{EL}$

There has been some significant progress about $\mathcal{EL}$ and contraction in recent years. In this subsection we cover all the important related works on them.

DL is used for formal reasoning on the concepts of an application domain, so the inference part is the most important part of a DL. The subsumption problem has always been a major problem in any DL: Given two concepts C and D and a TBox $\mathcal{T}$, is $C \sqsubseteq D$ w.r.t $\mathcal{T}$ true or not? $\mathcal{EL}$ is called a light-weight DL because there exists a polynomial-time subsumption algorithm for $\mathcal{EL}$. This algorithm proceeds in four steps [5]:

1. Normalize the TBox.

   An $\mathcal{EL}$ TBox is *normalized* if and only if it only contains GCIs of the following forms:

   $$A \sqsubseteq B$$

   $$A_1 \sqcap A_2 \sqsubseteq B$$

   $$A \sqsubseteq \exists r.B$$

   $$\exists r.A \sqsubseteq B$$

   Such a normal form can easily be computed in polynomial time and does not increase the size of the TBox more than polynomially. [10]

2. Translate the normalized TBox into a *classification graph*.

   we build the *classification graph* $G_{\mathcal{T}} = (V, V \times V, S, R)$ where V is the set of concepts occurring in the normalized TBox $\mathcal{T}$; S labels nodes with sets of concepts; R labels edges with sets of roles.

   Initially, we set $S(A) := \{A, \top\}$ for all nodes $A \in V$, and $R(A, B) := \emptyset$ for all edges $(A, B) \in V \times V$.

3. Complete the graph using completion rules.

   The labels of nodes and edges are then extended by applying 3 completion rules:

   (R1) If $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$ and $A_1, A_2 \in S(A)$, then add B to $S(A)$.

   (R2) If $A_1 \sqsubseteq \exists r.B \in \mathcal{T}$ and $A_1 \in S(A)$, then add r to $R(A, B)$.

   (R3) If $\exists r.B \sqsubseteq A_1 \in \mathcal{T}$ and $B_1 \in S(B)$, $r \in S(A, B)$, then add A_1 to $S(A)$.

   The extension stops when none of the completion rules can be applied. Rule application will terminate after a polynomial number of steps, and after that, the label sets are supposed to satisfy (see [10] for proofs):

$B \in S(A)$ if and only if $A \sqsubseteq B$.

$r \in S(A, B)$ if and only if $A \sqsubseteq \exists r.B$

4. Read off the subsumption relationships from the classification graph.

   As $B \in S(A)$ if and only if $A \sqsubseteq B$, if we want to know whether $A \sqsubseteq B$ is true, we can just simply read the graph and see if $B \in S(A)$.

With these 4 steps, we can actually classify the given TBox $\mathcal{T}$, i.e., simultaneously compute all subsumption relationships between the concept names occurring in $\mathcal{T}$ in polynomial time.

2.4 AGM Postulates for Belief Contraction

Belief contraction can be described as the process of **rationally** removing from a belief set K a certain belief φ. But what is the definition of "rationally" here? How should a "rational contraction" be? Researchers have proposed several postulates to describe the rationality of a contraction. Among them AGM [14] is the best-known approach. It gives a set of postulates characterizing all rational contraction functions. The aim is to describe belief contraction on the knowledge level. Let's say we have a logically closed knowledge base K (i.e., for any beliefs $\beta \in K$ if and only if $K \vDash \beta$) and want to contract a certain belief φ from K, $K - \varphi$ is the new knowledge sets after the contraction. The AGM postulates for belief contraction, are the following:

$(K-1)$ $K - \varphi$ is deductively closed.

$(K-2)$ $K - \varphi \subseteq K$.

$(K-3)$ If $\varphi \notin K$, then $K - \varphi = K$.

$(K-4)$ If $\nvDash \varphi$, then $\varphi \notin K - \varphi$.

$(K-5)$ If $\varphi \in K$, then $K \subseteq (K - \varphi) + K$.

$(K-6)$ If $\vDash \varphi \equiv \psi$, then $K - \varphi = K - \psi$.

$(K-7)$ $(K - \varphi) \cap (K - \psi) \subseteq K - (\varphi \wedge \psi)$

$(K-8)$ If $\varphi \notin K - (\varphi \wedge \psi)$, then $K - (\varphi \wedge \psi) \subseteq K - \varphi$

Any function - : $K_L * L \to K_L$ is called an *AGM contraction function*. We usually split $(K-1)$-$(K-8)$, into two groups: $(K-1)$-$(K-6)$ are called the basic AGM contraction postulates while the other two $(K-7)$ and $(K-8)$ are known as the supplementary AGM contraction postulates.

We will discuss further these postulates and also some other properties of a contraction in later chapters.

2.5 Definition of Kernel Contraction in $\mathcal{EL}$

Here we will introduce the basic definitions for kernel contraction [14] in $\mathcal{EL}$. Our starting point for defining contraction is in terms of kernel sets. As mentioned in previous section, any $\mathcal{EL}$ TBox can be normalized in polynomial time. For this reason, here we just consider the normalized $\mathcal{EL}$ TBox.

Here we follow Hansson's definition of kernel contraction in [14]:

Definition 1 *($\mathcal{EL}$ logical implication)*

X is a set of GCIs and α is a GCI. Then A logically implies α ($X \vDash \alpha$) if and only if there is no model $\mathcal{I}$ in which all members of X are true and α is false. Or, in other words, α can be derived from the GCIs in X.

Definition 2 *($\mathcal{EL}$ kernel set)*

For an $\mathcal{EL}$ TBox $\mathcal{T}$, A is a set of GCIs s.t. $A \subseteq \mathcal{T}$, α is a GCI s.t. $\alpha \in \mathcal{T}$. Then $A \perp \alpha$ is the set of sets such that $X \in A \perp \alpha$ if and only if:

1.$X \subseteq A$

2.$X \vDash \alpha$

3.If $Y \subset X$, then $Y \nvDash \alpha$.

$A \perp \alpha$ is the set of kernels*, and its elements are the α-kernels of A.*

Let β is a belief and $\beta \in \mathcal{T}$, if β in some α-kernels of A, we say β is *relevant* to α in A. Otherwise β is irrelevant to α. With kernel sets defined, now we will choose the set of beliefs to discard so that at least one belief is discarded in each kernel set. We call the function that expresses which to discard an *incision function.*

Definition 3 *($\mathcal{EL}$ incision function)*

For an $\mathcal{EL}$ TBox $\mathcal{T}$, A is a set of GCIs s.t. $A \subseteq \mathcal{T}$, α is a GCI s.t. $\alpha \in \mathcal{T}$. An incision function *σ for A is a function such that for all α:*

1.$\sigma(A \perp \alpha) \subseteq \bigcup(A \perp \alpha)$

2.If $\emptyset \neq X \in A \perp \alpha$, then $X \cap \sigma(A \perp \alpha) \neq \emptyset$

Once we have an incision function σ, we can define a kernel contraction accordingly by simply removing all the beliefs in $\sigma(A \perp \alpha)$ if we want to contract α from A. In the next definition, we use $\setminus$ as set minus, i.e. the elements of $A \setminus B$ are those elements in A but not in B.

Definition 4 *($\mathcal{EL}$ kernel contraction) For an $\mathcal{EL}$ TBox $\mathcal{T}$, A is a set of GCIs s.t. $A \subseteq \mathcal{T}$, α is a GCI s.t. $\alpha \in \mathcal{T}$. Let σ be an incision function for A. The kernel contraction $\approx_\sigma$ for A is defined as follows:*

$$A \approx_\sigma \alpha = A \setminus \sigma(A \perp \alpha)$$

An operator $-$ for A is a kernel contraction if and only if there are some incision functions σ for A such that $A - \alpha = A \approx_\sigma \alpha$ for all beliefs α.

As we can see, there can be many incision functions, thus there can also be many different kernel contractions. One main assumption of a good contraction is that of minimal change: the knowledge before and after the change should be as similar as possible, especially we should try "not giving up too much". The following is one way to give precise definition to the rather vague notion of "not giving up too much".

For example we want to contract belief α from a KB A, and there are totally 3 α-kernels: $\{a, b\}, \{a, c\}, \{a, b, d\}$. Obviously $\sigma_1(A \perp \alpha) = \{a\}$ would lead to the best kernel contraction in this case. We can also have $\sigma_2(A \perp \alpha) = \{b, c, d\}$, but σ_2 is giving up too much because there is redundancy in the give up set $\{b, c, d\}$, i.e., we can choose to believe d without bringing back any kernels ($\{b, c\}$ breaks all the kernels already). By doing that, we will have another contraction: $\sigma_3(A \perp \alpha) = \{b, c\}$, which is absolutely better than σ_2 because σ_3 ensure less change in the original KB than σ_2. More generally, we could say there is redundancy in the contractions like $A \approx_{\sigma_2} \alpha$, and we could call a contraction without any redundancy a *minimal kernel contraction.*

Definition 5 *(Minimal incision function)*

An incision function *σ for A is a minimal incision function if and only if:*

For all $\alpha \in A$ and $P \subsetneqq \sigma(A \perp \alpha)$, there exists $X \in A \perp \alpha$ such that $X \cap P = \emptyset$.

In other words, for all $\alpha \in A$, any proper subset of $\sigma(A \perp \alpha)$ cannot hit all the kernels.

Definition 6 *(Minimal kernel contraction)*

$A \approx_\sigma \alpha$ is a minimal kernel contraction if and only if σ is a minimal incision function.

By this definition, a good kernel contraction should always be a minimal kernel contraction because no one wants to give up beliefs that you don't really need to give up. But still there are many minimal kernel contractions to be considered, such as σ_1 and σ_2 are 2 minimal kernel contractions but still σ_1 is better than σ_2 as $\mid \sigma_1 \mid = 1 < 2 = \mid \sigma_2 \mid$.

Chapter 3

Kernel Contraction in $\mathcal{EL}$

This chapter is the main part of the thesis. It presents our contributions towards kernel contraction in $\mathcal{EL}$. It presents some analysis about kernel contraction in $\mathcal{EL}$ and also several new concepts to determine a good kernel contraction.

The chapter is divided into six sections. Section 3.1 introduces the process of generating the kernels. Section 3.2 introduces some algorithms to generate a kernel contraction based on the kernels. Section 3.3 analyzes some of the properties of kernel contraction in $\mathcal{EL}$. Section 3.4 and section 3.5 present new concepts called localization and specificity, respectively, which are new ways to exploit the structure of an $\mathcal{EL}$ knowledge base, also give some analysis and algorithms to them. Section 3.6 introduces hitting set problem and set cover problem which is closely related to kernel contraction and we link them together so that we can apply some progress to kernel contraction in $\mathcal{EL}$.

3.1 Generating kernels using axiom pinpointing

As we mentioned in the previous chapter above, generating all the α-kernels of a knowledge base A is necessary to build a kernel contraction which contracts α from A. Finding all the α-kernels of A is somewhat similar to answering the question: why does α hold in A? The answers to this question is actually all the α-kernels of A. For example,

$$A = \{\beta, \gamma, \beta \to \alpha, \gamma \to \alpha\}$$

Obviously $\alpha \in Cn(A)$ but why? Because β and $\beta \to \alpha$ both hold in A, or because γ and $\gamma \to \alpha$ both hold in A. Those are exactly the two α-kernels of A in this situation:$\{\beta, \beta \to \alpha\}$

and $\{\gamma, \gamma \to \alpha\}$. Each kernel is a minimal axiom set that helps us to comprehend why a certain consequence holds. For example, consider the $\mathcal{EL}$-Tbox $\mathcal{T}$ consisting of the GCIs

$$\text{ax}_1 : A \sqsubseteq \exists r.A, \text{ax}_2 : A \sqsubseteq Y, \text{ax}_3 : \exists r.Y \sqsubseteq B, \text{ax}_4 : Y \sqsubseteq B.$$

It is easy to see that $A \sqsubseteq B$. Two kernels of $\mathcal{T}$ w.r.t $A \sqsubseteq B$ are $\{\text{ax}_2,\text{ax}_4\}$ and $\{\text{ax}_1,\text{ax}_2,\text{ax}_3\}$.

In fact, finding all the kernels is exactly the goal of axiom pinpointing. Axiom pinpointing is about helping the user to understand the reasons why consequences hold and to remove unwanted consequences. According to the work in [7], we have a pinpointing algorithm for $\mathcal{EL}$ that can calculate all the kernels. Details of this pinpointing algorithm can be found in . The idea of this algorithm is based on the polynomial-time subsumption algorithm for $\mathcal{EL}$. Instead of computing the kernels directly, this algorithm follows the approach introduced in [7] that computes a monotone Boolean formula from which the kernels can be derived by the completion rules for subsumption in $\mathcal{EL}$.

A single kernel can be computed in polynomial time by simply going through all the GCIs. However, in order to compute all kernels, an exponential runtime cannot be avoided in the worst case since there may be exponentially many kernels for a given TBox. Here's an example [7] of linear size Tbox having exponential number of kernels:

$$T_n := \{B_{i-1} \sqsubseteq P_i \sqcap Q_i, P_i \sqsubseteq B_i, Q_i \sqsubseteq B_i | 1 \leq i \leq n\}$$

Fortunately when considering real knowledge base, in most cases, there are only linearly many kernels for a given TBox, so we do not really need to worry about the complexity of pinpointing algorithms in $\mathcal{EL}$. In other words, generating all the kernels according to the KB and the belief we want to contract is practical in application.

3.2 Algorithms for generating kernel contractions

In the last section we introduced about the algorithms for generating kernels in [7]. To decide a kernel contraction, this is not enough. Assume that we have calculated all the α-kernels of A. Still we need a algorithm to decide the incision function, i.e., to choose the set of beliefs to give up, in order to generate a kernel contraction. Here we list several algorithms to obtain kernel contractions.

The methods listed below are all based on the idea to generate a set of beliefs to give up. The part that we need to revise is usually a small part of the whole knowledge base, so

deciding which to give up is faster than deciding which to believe.
In the following algorithms, we generate a kernel contraction that contract α from A, given all the α-kernels of A. Note that *belief* is a list of all the beliefs in belief set A, *kernel* is a list of all the α-kernels of A, and *giveupset* is the set of beliefs that we decide to give up, i.e. the kernel contraction we generate would be $A \setminus giveupset$.

1. Randomly remove:
For every kernel, randomly choose a belief in it to give up.

function RANDOMLYREMOVE(*kernel*)
  giveupset $\leftarrow \{\}$
  for i from 1 to size(*kernel*) **do**
    randomly choose a belief α from *kernel*[i]
    giveupset $\leftarrow$ *giveupset* $\bigcup\{\alpha\}$
  end for
return *giveupset*
end function

This is the simplest and also the most straightforward algorithm. We can easily see that the time complexity of this algorithm is $O(n)$ (n is the number of kernels), which is the best time complexity we could possibly get. However, a belief may be included in more than one kernel, which means removing a belief will probably break more than one kernel, so removing a belief from *every* kernel will probably result in unnecessary changes.

2. Randomly remove in order of kernels:
List all the kernels in order, and check them one by one. If a kernel is still valid, i.e. we havent removed any beliefs in that kernel, we randomly choose a belief in it to give up; otherwise, skip that kernel and check the next one.

function RANDOMLYREMOVEINORDEROFKERNELS(*kernel*)
  giveupset $\leftarrow \{\}$
  for i from 1 to size(*kernel*) **do**

```
        valid[i] ← true                                  ▷ Set every kernel as valid.
    end for
    for i from 1 to size(kernel) do
        if valid[i] = true then                          ▷ If this kernel is still valid
            randomly choose a belief α from kernel[i]
            giveupset ← giveupset ⋃ {α}      ▷ Add a belief to gievupset to hit this kernel
            for j from i + 1 to size(kernel) do
                if α ∈ kernel[j] then
                    valid[j] ← false                     ▷ Set the hit kernels as invalid
                end if
            end for
        end if
    end for
return giveupset
end function
```

This algorithm gives a tag to every kernel to record their integrities. After removing a belief, refresh all the tags so that we can avoid those unnecessary changes from the first algorithm. The time complexity of this algorithm would be:

i loop: $O(n)$,

j loop: $O(n)$,

Total: $O(n^2)$.

The problem of this algorithm is that it still *randomly* removes beliefs when the kernels are valid. Some beliefs could be more crucial in the process of removing than others because they belong to more kernels.

3. Remove in order of beliefs: List all beliefs in order, and check them one by one. If giving up the current belief can reduce the number of valid kernels, we remove it; otherwise we skip this belief and check the next one.

```
function RandomlyRemoveInOrderOfBeliefs(kernel,belief)
    giveupset ← {}
    for i from 1 to size(belief) do
```

```
        for j from 1 to size(kernel) do
            if giveupset ⋂ kernel[j] = ∅ and belief[i] ∈ kernel[j] then
                giveupset ← giveupset ⋃ {belief[i]}        ▷ If this belief can hit any valid
kernels, add it to giveupset
            end if
        end for
    end for
return giveupset
end function
```

As we mentioned in the last algorithm, some beliefs could be more crucial in the process of removing. So instead of considering kernels, we can also take the actual beliefs into consideration. This algorithm is similar to the last one in the sense of randomly remove beliefs in some order but provides a different point of view which is to go through all the beliefs one by one until we break all the kernels. The time complexity of this algorithm is:

i loop: $O(m)$, (m is the number of beliefs))

j loop: $O(n)$,

Total: $O(mn)$.

4. Greedy algorithm: Remove the belief that appear in the most valid kernels each time until the removal hits all kernel.

```
function GreedyAlgorithm(kernel,belief)
    giveupset ← {}
    for i from 1 to size(kernel) do
        valid[i] ← true                                  ▷ Set every kernel as valid.
    end for
    validkernels ← size(kernel)
    for i from 1 to size(belief) do
        count[i] ← 0
    end for
    for j from 1 to size(kernel) do
        for i from 1 to size(belief) do
```

```
            if belief[i] ∈ kernel[j] then
                count[i] ← count[i] + 1                          ▷ Calculate count
            end if
        end for
    end for                    ▷ Here count[i] represent belief[i] appears in count[i] kernels
    while validkernels > 0 do
        max ← 0
        for i from 1 to size(belief) do
            if count[i] > max then
                max ← count[i]          ▷ Find maxbelief for the largest count[maxbelief]
            end if
        end for
        maxbelief ← 0
        for i from 1 to size(belief) do
            if count[i] == max then
                maxbelief ← i
            end if
        end for
        giveupset ← giveupset ⋃{belief[maxbelief]}          ▷ Add belief[maxbelief] to
giveupset
        for j from 1 to size(kernel) do
            if belief[maxbelief] ∈ kernel[j] and valid[j] == true then
                valid[j] ← false
                for i from 1 to size(belief) do
                    if belief[i] ∈ kernel[j] then
                        count[i] ← count[i] − 1                  ▷ Adjust count
                    end if
                end for
                validkernels ← validkernels − 1                  ▷ Adjust validkernels
            end if
        end for
    end while
return giveupset
```

end function

This algorithm is greedy in the sense that every time we remove a belief, we remove the belief that is a member of the largest number of the most remaining kernels. For example, if

$$\mathrm{k}_1 : \{a, b, c\}, \mathrm{k}_2 : \{b, c, d\}, \mathrm{k}_3 : \{c, d, e\}, \mathrm{k}_4 : \{a, c, e\}$$

are all the kernels, we will remove c according to the greedy algorithm because c appears in all 4 kernels. Obviously, the greedy algorithm does not guarantee the minimal change kernel contraction, but still it performs very well. The time complexity of this greedy algorithm is:

while loop: $O(n)$,

j loop: $O(n)$,

i loop: $O(m)$,

Total: $O(m * n^2)$,

which means we can generate a comparatively small change kernel contraction in polynomial time.

In fact, according to the work about set cover problem and the relation between kernel contraction and set cover problem which we will discuss in later sections, the greedy algorithm is essentially the best-possible polynomial time approximation algorithm for general kernel contraction, under plausible complexity assumptions.

3.3 Properties of kernel contraction in $\mathcal{EL}$

In this section we will analyze the properties of kernel contraction, especially kernel contraction in $\mathcal{EL}$. As we mentioned in the beginning, the AGM postulates are the best-known approach to characterize rational contraction functions. We will first discuss the AGM properties then some other properties of kernel contraction in $\mathcal{EL}$. Here we assume that we want to contract α from a belief set A. As defined in the definition of kernel contraction in $\mathcal{EL}$, $A - \alpha$ is the new belief set after the kernel contraction and $Cn(A - \alpha)$ is the knowledge base accordingly.

We may briefly recall all the AGM postulates according to our notations:

$(K-1)$ $Cn(A-\alpha)$ is deductively closed.
$(K-2)$ $Cn(A-\alpha) \subseteq Cn(A)$.
$(K-3)$ If $\alpha \notin Cn(A)$, then $Cn(A-\alpha) = Cn(A)$.
$(K-4)$ If $\nvDash \alpha$, then $\alpha \notin Cn(A-\alpha)$.
$(K-5)$ If $\alpha \in Cn(A)$, then $Cn(A) \subseteq Cn((A-\alpha)+\alpha)$.
$(K-6)$ If $\vDash \alpha \equiv \beta$, then $Cn(A-\alpha) = Cn(A-\beta)$.
$(K-7)$ $Cn(A-\alpha) \cap Cn(A-\beta) \subseteq Cn(A-\alpha \wedge \beta)$
$(K-8)$ If $\alpha \notin Cn(A-\alpha \wedge \beta)$, then $Cn(A-\alpha \wedge \beta) \subseteq Cn(A-\alpha)$

$(K-7), (K-8)$ postulates are not applicable here because conjunction between GCIs is not allowed in $\mathcal{EL}$. So next we will check $(K-1)$ to $(K-6)$ postulates of kernel contraction in $\mathcal{EL}$.

Theorem 1 *Kernel contraction in $\mathcal{EL}$ satisfies $(K-1)$, $(K-2)$, $(K-3)$, $(K-4)$, $(K-6)$ AGM postulates but not $(K-5)$.*

Proof

$(K-1)$ postulate says that $Cn(A-\alpha)$ should be deductively closed, which is included in the definition of Cn.

$(K-2)$ postulate, also known as the inclusion postulate, says that $A-\alpha \subseteq A$. This is natural since we do not want to add anything new with our contraction. Obviously our kernel contraction in $\mathcal{EL}$ satisfyies this postulate because no new beliefs are added into the belief set in our kernel contractions.

$(K-3)$ postulate, also known as the vacuity postulate, says that if $\alpha \notin Cn(A)$, then $A-\alpha = A$. This can be interpreted as we should not do anything when required to take away something that is not there. If $\alpha \notin Cn(A)$, i.e. $A \nvDash \alpha$, then for any subset X of A, we have $X \nvDash \alpha$, which means there are no α-kernels of A. So we don't need to give up any beliefs, i.e. $A-\alpha = A$. Thus our kernel contraction in $\mathcal{EL}$ also satisfies the vacuity postulate.

$(K-1), (K-2), (K-3)$ postulates are the most important postulates that all rational contraction should satisfy because they are related to the major goals of belief contraction itself.

$(K-4)$ postulate, also known as the failure postulate, says that if $\alpha \notin Cn(\emptyset)$, then $A-\alpha = A$. Intuitively we should not follow an invalid request that asks us to give up a

tautology. This is similar to $(K-3)$ postulate. We should not do anything when required to take away a tautology. If $\alpha \notin Cn(\emptyset)$, i.e. $\emptyset \vDash \alpha$, then for any subset X $(\neq \emptyset)$ of A, X is not a α-kernels of A since $\emptyset$ is a proper subset of X and $\emptyset \vDash \alpha$, which means there are no α-kernels of A. So again $A - \alpha = A$ and our kernel contraction in $\mathcal{EL}$ satisfies the failure postulate.

$(K-5)$ postulate, also known as the recovery postulate is a long-discussed postulate as it is sometimes unreasonable. It says that if $\alpha \in Cn(A)$, then $Cn(A) \subseteq Cn((A-\alpha)+\alpha)$ (Here $+\alpha$ means simply add α into the belief set, which means contracting and then expanding by α will give us back (at least) the initial theory A. In fact, if $(K-5)$ postulate hold, we get back precisely A because of $(K-2)$ postulate. Our kernel contraction in $\mathcal{EL}$ actually does not satisfy this postulate. For example, $A = \{X \sqsubseteq Y, Y \sqsubseteq Z\}$ and $\alpha = X \sqsubseteq Z$, then $A - \alpha$ is either $\{X \sqsubseteq Y\}$ or $\{Y \sqsubseteq Z\}$, so

$$Cn(A) = Cn\{X \sqsubseteq Y, Y \sqsubseteq Z\} = \{X \sqsubseteq Y, Y \sqsubseteq Z, X \sqsubseteq Z\}$$

$$\notin Cn((A-\alpha)+\alpha) = \{X \sqsubseteq Y, X \sqsubseteq Z\} or \{X \sqsubseteq Z, Y \sqsubseteq Z\}$$

$(K-6)$ postulate , also known as extensionality postulate, says that if $\vDash \alpha \leftrightarrows \beta$, then $A - \alpha = A - \beta$, which means contraction by logically equivalent sentences produces the same result. If α and β are logically equivalent, then they should have the same kernel sets of A according to the definition of $\mathcal{EL}$ kernel sets. So $A - \alpha$ and $A - \beta$ should produce the same result, i.e. kernel contraction in $\mathcal{EL}$ satisfies this postulate.

Beside the AGM postulates, there are some other properties worth discussing.

Relevance postulate: If $\beta \in A$ and $\beta \notin A-\alpha$, then there is a set A' such that $A-\alpha \subseteq A' \subseteq A$ and that $\alpha \notin Cn(A')$ but $\alpha \in Cn(A' \bigcup \{\beta\})$.

This postulate says that we should not give up beliefs for no reason, in other words, if we give up β in our contraction, β should be relevant to α in some way. Intuitively kernel contraction in $\mathcal{EL}$ should satisfy the vacuity postulate because the beliefs we give up belong to at least one of the α-kernels of A. However kernel contraction can be "not giving up too much" something discussed last section, for example $A = \{V \sqsubseteq W, W \sqsubseteq X, W \sqsubseteq Y, X \sqsubseteq Z, Y \sqsubseteq Z\}$, $\alpha = V \sqsubseteq Z$, $A - \alpha = \{W \sqsubseteq Y, X \sqsubseteq Z, Y \sqsubseteq Z\}$ and $\beta = W \sqsubseteq X$ does not satisfy relevance postulate just because giving up β is not necessary with $X \sqsubseteq Z$ given up. In the

other hand, if we only consider all *minimal kernel contractions*, we can easily see that they actually satisfy this relevance postulate by its definition.

3.4 Localization

Items of knowledge are not independent, they may be related. If one day we find out that a piece of knowledge about mammoths is wrong, then we may tend to suspect the reliability of other knowledge about mammoth since that mistake could be interpreted as a misunderstanding about this creature. More generally, the beliefs are more likely to be wrong if they share some common concepts or roles with a wrong belief because they are directly related to that wrong belief. By this intuition, when we select some beliefs to be discarded to form a kernel contraction, we may want those beliefs to be localized on the classification graph in order to get a good kernel contraction. In other words, it would be a better kernel contraction if the elements of the incision function σ are *localized*.

Here is an example:

$$\text{OrnithorhynchusAnatinus} \sqsubseteq \text{LayEggs}$$
$$\text{LayEggs} \sqsubseteq \text{NotAMammal}$$
$$\text{Ornithorhynchus Anatinus} \sqsubseteq \exists \text{ Spurs.Poisonous}$$
$$\exists \text{ Spurs.Poisonous} \sqsubseteq \text{NotAMammal}$$

We want to contract:

$$\text{OrnithorhynchusAnatinus} \sqsubseteq \text{NotAMammal}$$

We can have two reasonable localized kernel contractions:

- (1).Discard OrnithorhynchusAnatinus $\sqsubseteq$ LayEggs and OrnithorhynchusAnatinus $\sqsubseteq \exists$ purs.Poisonous, which means we are not familiar with this creature and the information we have about it are wrong.
- (2).Discard LayEggs $\sqsubseteq$ NotAMammal and $\exists$ Spurs.Poisonous $\sqsubseteq$ NotAMammal, which means there is exceptions of those properties of mammal.

It does not make sense when the elements of σ are isolated, such as discarding OrnithorhynchusAnatinus $\sqsubseteq$ LayEggs and $\exists$ Spurs.Poisonous $\sqsubseteq$ NotAMammal.

Base on the idea of localization, we first define adjacent GCIs, i.e. the GCIs that are close to each other.

Definition 7 *(Adjacent GCIs in $\mathcal{EL}$) For an $\mathcal{EL}$ TBox $\mathcal{T}$, α, β are two GCIs s.t. $\alpha, \beta \in \mathcal{T}$, we say α and β are adjacent GCIs if and only if there is at least one common concept or role in both α and β.*

For example,

$$\text{Hand}_P \sqsubseteq \exists \text{ part.Hand}_E, \text{ Hand}_E \sqsubseteq \text{Hand}_S$$

are adjacent GCIs, while

$$\text{Hand}_P \sqsubseteq \exists \text{ part.Hand}_E, \text{ Finger}_P \sqsubseteq \text{Finger}_S$$

are not.

With the definition of adjacent GCIs, we can draw an undirected graph by using GCIs as nodes and we draw an edge between two GCIs if they are adjacent GCIs and define connected GCIs based on this graph.

Definition 8 *(Connected GCIs in $\mathcal{EL}$) For an $\mathcal{EL}$ TBox $\mathcal{T}$, α, β are two GCIs s.t. $\alpha, \beta \in \mathcal{T}$, we say α and β are connected if and only if this undirected graph contains a path from α to β. Applying the definitions in graph theory, α and β are from the same cluster if and only if α and β are connected.*

Definition 9 *(Clusters) For an $\mathcal{EL}$ TBox $\mathcal{T}$, α, β are two GCIs s.t. $\alpha, \beta \in \mathcal{T}$, α and β are in the same clusters if and only if there exist a path $\alpha\gamma_1\gamma_2...\gamma_n\beta$ where $\gamma_i \in \mathcal{T} (i = 1, 2, ..., n)$ such that α and γ_1 , γ_i and $\gamma_{i+1} (i = 1, 2, ..., n-1)$, γ_n and β are connected.*

Now we can quantify how localized a contraction is by the number of clusters on this undirected graph.

Definition 10 *(The clustering number of a incision function) Let σ be an incision function for A. Using the definition of* connected GCIs *above, the clustering number C of $\sigma(A \perp \alpha)$ is defined as follows:*

$$C(\sigma(A \perp \alpha)) = \textit{the number of clusters of the GCIs in } \sigma(A \perp \alpha)$$

With the definitions above, we get a new criterion to determine a better kernel contraction:

We want to contract α from A and σ_1, σ_2 are two incision functions. If

$$C(\sigma_1(A \perp \alpha)) < C(\sigma_2(A \perp \alpha))$$

then it is likely that $A \approx_{\sigma_1} \alpha$ is better than $A \approx_{\sigma_2} \alpha$.

In order to calculate the clustering number of a incision function, we can apply the algorithm following:

function ClusteringNumbers($\sigma(A \perp \alpha)$)
  $ClusteringNum \leftarrow 0$
  for i from 1 to size($\sigma(A \perp \alpha)$) **do**
    $mark[i] \leftarrow i$
  end for
  for i from 1 to size($\sigma(A \perp \alpha)$) **do**
    $cluster[i] \leftarrow 0$
  end for
  for i from 1 to size($\sigma(A \perp \alpha)$) **do**
    for j from $i+1$ to size($\sigma(A \perp \alpha)$) **do**
      if $\sigma(A \perp \alpha)[i]$ and $\sigma(A \perp \alpha)[j]$ **then**
        $mark[j] \leftarrow mark[i]$
      end if
    end for
  end for
  for i from 1 to size($\sigma(A \perp \alpha)$) **do**
    $cluster[mark[i]] \leftarrow 1$
  end for
  for i from 1 to size($\sigma(A \perp \alpha)$) **do**
    $ClusteringNum \leftarrow ClusteringNum + cluster[i]$
  end for
return $ClusteringNum$
end function

In the algorithm above we treat $\sigma(A \perp \alpha)$ as a list of beliefs in the incision function. Also we use the array $mark$ to record the clusters: α and β are in the same cluster if and only

if they have the same *mark* value. The time complexity of this algorithm is $O(p)$, (p is the number of beliefs in the incision function, which should be comparatively small)

Here is an example in $\mathcal{EL}$ [8]:

$$\text{AmpOfFinger} \equiv \text{Amp} \sqcap \exists \text{ site.Finger}_S$$
$$\text{AmpOfHand} \equiv \text{Amp} \sqcap \exists \text{ site.Hand}_S$$
$$\text{InjToFinger} \equiv \text{Inj} \sqcap \exists \text{ site.Finger}_S$$
$$\text{InjToHand} \equiv \text{Inj} \sqcap \exists \text{ site.Hand}_S$$
$$\text{Finger}_E \sqsubseteq \text{Finger}_S$$
$$\text{Finger}_P \sqsubseteq \text{Finger}_S \sqcap \exists \text{ part.Finger}_E$$
$$\text{Hand}_E \sqsubseteq \text{Hand}_S$$
$$\text{Hand}_P \sqsubseteq \text{Hand}_S \sqcap \exists \text{ part.Hand}_E$$
$$\text{ULimb}_E \sqsubseteq \text{ULimb}_S$$
$$\text{ULimb}_P \sqsubseteq \text{ULimb}_S \sqcap \exists \text{ part.ULimb}_E$$
$$\text{Finger}_S \sqsubseteq \text{Hand}_P$$
$$\text{Hand}_S \sqsubseteq \text{ULimb}_P$$

It is not hard to see that, w.r.t. this ontology,

$$\text{Finger}_P \sqsubseteq \exists \text{part.ULimb}_S$$

We want contract $\text{Finger}_P \sqsubseteq \exists \text{part.ULimb}_S$ from this KB. There are 3 kernels of this belief:

Kernel 1.

$$\text{Finger}_P \sqsubseteq \text{Finger}_S$$
$$\text{Finger}_S \sqsubseteq \text{Hand}_P$$
$$\text{Hand}_P \sqsubseteq \text{Hand}_S$$
$$\text{Hand}_S \sqsubseteq \text{ULimb}_P$$
$$\text{ULimb}_P \sqsubseteq \exists \text{ part.ULimb}_E$$
$$\text{ULimb}_E \sqsubseteq \text{ULimb}_S$$

Kernel 2.

$$\text{Finger}_P \sqsubseteq \text{Finger}_S$$
$$\text{Finger}_S \sqsubseteq \text{Hand}_P$$
$$\text{Hand}_P \sqsubseteq \exists \text{ part.Hand}_E$$
$$\text{Hand}_E \sqsubseteq \text{Hand}_S$$

$$\text{Hand}_S \sqsubseteq \text{ULimb}_P$$
$$\text{ULimb}_P \sqsubseteq \text{ULimb}_S$$

Kernel 3.

$$\text{Finger}_P \sqsubseteq \exists \text{ part.Finger}_E$$
$$\text{Finger}_E \sqsubseteq \text{Finger}_S$$
$$\text{Finger}_S \sqsubseteq \text{Hand}_P$$
$$\text{Hand}_P \sqsubseteq \text{Hand}_S$$
$$\text{Hand}_S \sqsubseteq \text{ULimb}_P$$
$$\text{ULimb}_P \sqsubseteq \text{ULimb}_S$$

These 3 kernels do not have a common GCI, so we need to give up at least 2 GCIs to do a kernel contraction and there are many ways to do so. For example:

σ_1: give up $\text{Finger}_P \sqsubseteq \text{Finger}_S$ and $\text{Finger}_E \sqsubseteq \text{Finger}_S$.

σ_2: give up $\text{Finger}_P \sqsubseteq \text{Finger}_S$ and $\text{Hand}_S \sqsubseteq \text{ULimb}_P$.

σ_3: give up $\text{Hand}_P \sqsubseteq \exists$ part.Hand_E and $\text{ULimb}_P \sqsubseteq \text{ULimb}_S$.

etc.

According to localization, σ_1 is the best among because $C(\sigma_1) = 1 < 2 = C(\sigma_2) = N(\sigma_3)$. Here is another example in $\mathcal{EL}^+$ [4]:

$$\text{Endocardium} \sqsubseteq \text{Tissue} \sqcap \exists \text{ cont-in.HeartWall} \sqcap \exists \text{ cont-in.HeartValve}$$
$$\text{HeartWall} \sqsubseteq \text{BodyWall} \sqcap \exists \text{ part-of.Heart}$$
$$\text{HeartValve} \sqsubseteq \text{BodyValve} \sqcap \exists \text{ part-of.Heart}$$
$$\text{Endocarditis} \sqsubseteq \text{Inflammation} \sqcap \exists \text{ has-loc.Endocardium}$$
$$\text{Inflammation} \sqsubseteq \text{Disease} \sqcap \exists \text{ act-on.Tissue}$$
$$\text{Heartdisease} \sqcap \exists \text{ has-loc.HeartValve} \sqsubseteq \text{CriticalDisease}$$
$$\text{Heartdisease} = \text{Disease} \sqcap \exists \text{ has-loc.Heart}$$
$$\text{part-of} * \text{part-of} \sqsubseteq \text{part-of}$$
$$\text{part-of} \sqsubseteq \text{cont-in}$$
$$\text{has-loc} * \text{cont-in} \sqsubseteq \text{has-loc}$$

It is not hard to see that, w.r.t. this ontology, endocarditis is classified as a heart disease, i.e.,

$$\text{Endocarditis} \sqsubseteq \text{Heartdisease}$$

To see this, note that Endocarditis implies Inflammation and thus Disease,which yields the first conjunct in the definition of Heartdisease. Moreover, Endocarditis implies ∃ has-loc.Endocardium which can be seen to imply ∃has-loc.∃cont-in.∃part-of:Heart. In the presence of the last two RIs, this implies ∃has-loc:Heart, which is second conjunct in the definition of Heartdisease.

If somehow we find that endocarditis is not a heart disease, then we may want to contract

$$\text{Endocarditis} \sqsubseteq \text{Heartdisease}$$

from this knowledge base. It is not hard to see that there are two kernels:

$$\text{Endocarditis} \sqsubseteq \text{Inflammation}$$
$$\text{Inflammation} \sqsubseteq \text{Disease}$$
$$\text{Endocarditis} \sqsubseteq \exists \text{ has-loc.Endocardium}$$
$$\text{Endocardium} \sqsubseteq \exists \text{ cont-in.HeartWall}$$
$$\text{HeartWall} \sqsubseteq \exists \text{ part-of.Heart}$$
$$\text{part-of} \sqsubseteq \text{cont-in}$$
$$\text{has-loc} * \text{cont-in} \sqsubseteq \text{has-loc}$$

and

$$\text{Endocarditis} \sqsubseteq \text{Inflammation}$$
$$\text{Inflammation} \sqsubseteq \text{Disease}$$
$$\text{Endocarditis} \sqsubseteq \exists \text{ has-loc.Endocardium}$$
$$\text{Endocardium} \sqsubseteq \exists \text{ cont-in.HeartValve}$$
$$\text{HeartValve} \sqsubseteq \exists \text{ part-of.Heart}$$
$$\text{part-of} \sqsubseteq \text{cont-in}$$
$$\text{has-loc} * \text{cont-in} \sqsubseteq \text{has-loc}$$

Then we can easily see that the clustering number of a good contraction here should be 1, which is the minimum. In fact, this happens in many of the knowledge bases. We may call it one-cluster contraction. Based on ensuring minimum removal, a good contraction should be one-cluster contraction.

Here we present the algorithms for generating a kernel contraction in $\mathcal{EL}$ with a small clustering number.

```
function AvoidIncreasingClusteringNumber(kernel,belief)
    giveupset ← {}
    for i from 1 to size(kernel) do
        valid[i] ← true
    end for
    for i from 1 to size(kernel) do
        if valid[i] = true then
            for each α ∈ kernel[i] do
                if α and giveupset are connected & valid[i] = true then
                    giveupset ← giveupset ⋃ {α}
                    for j from 1 to size(kernel) do
                        if α ∈ kernel[j] then
                            valid[j] ← false
                        end if
                    end for
                end if
            end for
            if valid[i] = true then
                Randomly choose a belief α from kernel[i]
                giveupset ← giveupset ⋃ {α}
                for j from 1 to size(kernel) do
                    if α ∈ kernel[j] then
                        valid[i] ← false
                    end if
                end for
            end if
        end if
    end for
return giveupset
end function
```

This algorithm is based on the function *RandomlyRemoveInOrderOfKernels* we defined before. However, instead of randomly removing a belief from a valid kernel, we tend to

remove a belief which is connected to *giveupset* so that the clustering number will not increase. The time complexity of this algorithm is the same as the time complexity of function *RandomlyRemoveInOrderOfKernels*, which is $O(n^2)$. The problem of this algorithm is that it considers the kernels one by one but not as a whole which may lead to inefficiency.

function GREEDYPLUSCLUSTERINGNUMBER(*kernel*,*belief*)
GreedyAlgorithm(*kernel*,*belief*) ▷ Here we need to change GreedyAlgorithm a little bit. After we find the beliefs those hitting the most kernels, we give up the one that is connected to *giveupset*. (If not found, give up a random belief among them)
end function

This algorithm is a combination of the greedy algorithm we defined before and the concept of localization. In this algorithm, we first ensure every time the belief we remove "hit" the most remaining kernels by the greedy algorithm, then we find a belief among them so that removing this belief will not increase the clustering number. The time complexity of this algorithm is the same as the time complexity of function *GreedyAlgorithm*, which is $O(m * n^2)$.

3.5 Specificity

When we want to contract a belief, usually we find it more reasonable to discard something specific instead of giving up something general. For example, in a medical system, we tend to discard some properties of a specific disease instead of giving up the general category of illness. If the concepts/roles in a GCI are all general concepts/roles, then very likely this GCI is true. How can we know whether a concept/role is general or not? We quantify the generality of a concept/role by the number of the GCIs that relate to this concept/role ($d(v)$).

$$d(v) = \text{degree of vertex } v = \text{number of edges incident with } v$$

Here a GCI is related to a concept/role if this concept/role is in this GCI.

Definition 11 *(degree of a concept/role) For an $\mathcal{EL}$ TBox $\mathcal{T}$ and a belief $\alpha \in \mathcal{T}$, v is a concept/role that appears $\mathcal{T}$. We say that α is related to v if v appears in α. We define $d(v)$, the degree of v, as the number of GCIs that are related to v.*

Definition 12 *(Specificity of a GCI) For an $\mathcal{EL}$ TBox $\mathcal{T}$ and a belief $\alpha \in \mathcal{T}$, we define the* specificity *of α $S(\alpha)$ as $S(\alpha) = average_{v \in \alpha}(d(v))$ where v is a concept/role.*

Theoretically, if $S(\alpha) > S(\beta)$ where $\alpha, \beta \in \mathcal{T}$, then from the definitions we know that on average the concepts/roles in α should appear in more GCIs than the ones in β do, which means the concepts/roles in α should be more general than the ones in β, in other words, α should be more reliable than β in $\mathcal{T}$. If $S(\alpha)$ is comparatively large, it means that all the concepts and roles in α are reliable, so very likely α is a true statement and our contraction should not remove it during the process of kernel contraction. By considering specificity of all the GCIs, we can assign true value to some of the GCIs with large specificity before we start to do the contraction, which may make the contraction work much easier.

For instance, in this knowledge base below:

$$\text{Endocardium} \sqsubseteq \text{Tissue} \sqcap \exists \text{ cont-in.HeartWall} \sqcap \exists \text{ cont-in.HeartValve}$$
$$\text{HeartWall} \sqsubseteq \text{BodyWall} \sqcap \exists \text{ part-of.Heart}$$
$$\text{HeartValve} \sqsubseteq \text{BodyValve} \sqcap \exists \text{ part-of.Heart}$$
$$\text{Endocarditis} \sqsubseteq \text{Inflammation} \sqcap \exists \text{ has-loc.Endocardium}$$
$$\text{Inflammation} \sqsubseteq \text{Disease} \sqcap \exists \text{ act-on.Tissue}$$
$$\text{Heartdisease} \sqcap \exists \text{ has-loc.HeartValve} \sqsubseteq \text{CriticalDisease}$$
$$\text{Heartdisease} = \text{Disease} \sqcap \exists \text{ has-loc.Heart}$$
$$\text{part-of} * \text{part-of} \sqsubseteq \text{part-of}$$
$$\text{part-of} \sqsubseteq \text{cont-in}$$
$$\text{has-loc} * \text{cont-in} \sqsubseteq \text{has-loc}$$

Heartdisease = Disease $\sqcap$ $\exists$ has-loc. Heart should be trustworthy because d(Heartdisease), d(Disease), d(has-loc), d(Heart) are comparatively large.

Here we present the algorithms to generate kernel contractions including specificity as a factor.

function SPECIFICITYDIVIDING($kernel$,$belief$)
  $newBelief \leftarrow \{\}$
  $\mu \leftarrow 50\%$
  for i from 1 to size($belief$) **do**

$F[i] \leftarrow average_{v \in belief[i]}(d(v))$
end for
for i from 1 to size($belief$) **do**
if $F[i]$ is in the μ larger part among F **then**
$trust[i] \leftarrow true$
end if
end for
for i from 1 to size($belief$) **do**
if $trust[i] == false$ **then**
$newBelief = newBelief + belief[i]$
end if
end for
GreedyAlgorithm($kernel$,$newBelief$)
end function

The algorithm above is a combination of the greedy algorithm and specificity. We sort the GCIs with their specificity. Then we can set a threshold μ to separate them into two groups. The GCIs in the groups with larger specificity will be assigned *true* automatically so that we will not give up any of them until we have to. This algorithm looks longer than the original greedy algorithm but it is actually simpler because we can ignore some of the GCIs with large specificity. It is easy to see that the complexity of this algorithm is $O(m * n^2)$, same as the original greedy algorithm.

function SpecificityScaling($kernel$,$belief$)
for i from 1 to size($belief$) **do**
$F[i] \leftarrow average_{v \in belief[i]}(d(v)$
end for
GreedyAlgorithm($kernel$,$belief$) ▷ Here we need to change GreedyAlgorithm a little bit by replacing $count[i] > max$ with $count[i] * F[i] > max$ and $count[i] = max$ with $count[i] * F[i] > max$ in the conditions.
end function

The algorithm above presents another possible way to use specificity. Instead of using specificity to divide all GCIs into groups, we use specificity as a "weight" in the part that deciding the most crucial belief each turn by simply replacing $count[i]$ by $count[i] * F[i]$. It is easy to see that the complexity of this algorithm is $O(m * n^2)$, same as the original greedy algorithm.

Specificity of a GCI can be used to determine those reliable GCIs in a large knowledge base. But it may not work so well in small knowledge base as the generality of a concept/role may not be represented correctly in small knowledge base.

3.6 Connection with Hitting Set Problem and Set Cover Problem

In this section we look into two kinds of problems: the hitting set problem and the set cover problem. We will see that the process to select the beliefs to give up in kernel contraction is actually equivalent to the process to generate a hitting set or a set covering, which means we can apply some of the results in the hitting set problem and the set cover problem to kernel contraction.

3.6.1 Hitting Set Problem

Let us first consider the *hitting set problem* (HSP):

We are given a set H and a set C of subsets $C_1, C_2, ..., C_m \subseteq H$. We are looking for a subset $S \subseteq H$ such that

- $S \cap C_i \neq \emptyset, i = 1, 2, .., m.$

Such a set S is called a *hitting set* of C, because it "hits" all subsets C_i. If no elements can be removed from S without violating the hitting set property, then we say that S is minimal. Let M be the collection of the minimal hitting sets. Determining a minimal cardinality element of M is called the *optimal hitting set problem* (or simply *hitting set problem*). We can also call this element an *optimal hitting set* of C

The hitting set problem is equivalent to many known problems such as the set covering problem. And it is not difficult to prove that finding the optimal hitting set is NP-hard. As of now, no feasible algorithm for finding the optimal hitting set has been discovered (actually there may be none because it is NP-hard), but still people have made progress on finding a good hitting set (with relative small cardinality) using different algorithm, likes approximation algorithms.

There is also an expansion of the *hitting set problem*, the *weighted hitting set problem*:

We are given a set H and a set C of subsets $C_1, C_2, ..., C_m \subseteq H$. The elements of H have positive weights $w_a, a \in H$. We are looking for a subset $S \subseteq H$ such that

- $S \cap C_i \neq \emptyset, i = 1, 2, .., m$.
- $\sum_{a \in S} w_a$ is minimal.

Such a set S is called the *weighted-minimal hitting set* or *optimal hitting set*.

Based on the normal hitting set problem, the weighted hitting set problem adds different weights to different elements in H. It is exactly the same as the normal hitting set problem when all the elements in H are equally weighted, i.e. $w_a = w_b, a, b \in H$. And finding the weighted-minimal hitting set is also NP-hard. Similar progress has been made on finding a good weighted hitting set (with relatively small total weight) using approximation algorithms.

3.6.2 Set Cover Problem

The *set cover problem* (SCP) is a another classical question in combinatorics, computer science and complexity theory.

We are given a set H and a set C of subsets $C_1, C_2, ..., C_m \subseteq H$. We are looking for a subfamily $S \subseteq C$ such that

- $\bigcup_{A \in S} A = H$

Such a set S is called a *set covering* of H, because its union covers the whole set H. If no elements can be removed from S without violating the set covering property, then we

say that S is minimal. Let M be the collection of the minimal set coverings. Determining a minimal cardinality element of M is called the *optimal set cover problem* (or simply *set cover problem*). We can also call this element an *optimal set covering* of H.

It is not difficult to see that the set cover problem is equivalent to the hitting set problem by considering the set C in the set cover problem as the set H in the hitting set problem and the set H in the set cover problem as the set C in the hitting set problem reversely. Hence, the set cover problem is also NP-hard.

3.6.3 Connections between Kernel Contraction, Hitting Set Problem and Set Cover Problem

We discussed the definitions of the hitting set problem and the set cover problem. Now we can explain the connections between kernel contraction, the hitting set problem and the set cover problem.

Let say we want to contract α from A. $A \perp \alpha$ is the kernel set, whose elements are the α-kernels of A. And $\sigma(A \perp \alpha)$ is an incision function, i.e. it is the set of beliefs to be given up. By definition, if $\emptyset \neq X \in A \perp \alpha$, then $X \cap \sigma(A \perp \alpha) \neq \emptyset$, which means $\sigma(A \perp \alpha)$ "hits" all the α-kernels. We can consider "finding a set of beliefs to be given up" as "finding a hitting set of all the kernels". In this way, "finding a set of beliefs to be given up with minimal cardinality" is exactly the same thing as "finding an optimal hitting set of all the kernels", which means the process to select the beliefs to give up in kernel contraction is equivalent to the process to generating a hitting set. As we mentioned last section that the set cover problem is equivalent to the hitting set problem, we can apply everything about the hitting set problem and the set cover problem to kernel contraction.

As finding an optimal hitting set is NP-hard, we know that finding a kernel contraction with minimal change is also NP-hard. However, finding a kernel contraction with minimal change can be solved in polynomial time under low-frequency systems: If each belief occurs in at most f kernels, the solution can be found in polynomial time that approximates the optimum to within a factor of f using LP relaxation [23]. Also, it has been shown that the greedy algorithm (at each stage, choose the set that contains the largest number of uncovered elements) is essentially the best-possible polynomial time approximation algorithm for the set cover problem, under plausible complexity assumptions, which means the greedy algorithm we presented in the previous section is the best-possible polynomial time algorithm we can get.

What about the weighted hitting set problem? Are there any corresponding concepts in kernel contraction? In the weighted hitting set problem we give every element a weight to distinguish them. In kernel contraction, the elements are different, they are different beliefs in description logic. If we relate these two to each other, we could give every belief a weight to represent the reliability of that belief. In this way, the sum of weights of beliefs to be given up in a kernel contraction can be interpreted as the unreliability of this kernel contraction. With this measurement, "finding the best kernel contraction" can be interpreted as "finding the set of beliefs to be given up with minimal cardinality", which is also the same thing as "finding a weighted-minimal hitting set of all the kernels".

Let's say for belief $\alpha \in A$, $W(\alpha)$ is the weight of this belief. Function W should satisfy:

- $\alpha, \beta \in A$, $W(\alpha) > W(\beta)$ if and only if α is more reliable than β.

However, this property is still unclear because different people might have different opinions about the reliability of a certain belief. We need some clear properties to describe the weight of a belief. Now we may need to analyze the structure and the concepts of a belief.

When we want to contract a belief, usually we find it more reasonable to discard something specific instead of giving up something general. For example in a medical system, we tend to discard some properties of a specific disease instead of giving up the general category of illness. If the concepts/roles in a GCI are all general concepts/roles, then very likely this GCI is true.

How can we know whether a concept/role is general or not? This is actually something we discussed in the previous section "specificity". Following what we have done in "specificity", one possible way would be to quantify the generality of a concept/role by the number of the GCIs that related to this concept/role($d(v)$). Here a GCI is related to a concept/role if this concept/role is in this GCI. If $d(u) > d(v)$, probably u is more general than v. Another possible way would be to analyse the hierarchy of the knowledge base. The concept at the higher level should be more general and the lower level should be more specific.

Here are some of my suggestions of what a function W could be:

- $w_1(\alpha) = average_{v \in \alpha}(d(v))$

$w_1(\alpha)$ takes vertex degrees of all the concepts/roles appear in the α into account averagely using arithmetic mean. This actually comes from the definition of specificity in the

last section. It treats all the concepts/roles equally and it is very easy to be applied.

- $w_2(\alpha) = 1/(2^{min_{v\in\alpha}(level(v))+max_{v\in\alpha}(level(v))})$

Beside the degrees of concepts/roles, the level of a vertex in the hierarchy tree can also be a factor. $w_2(\alpha)$ consider the highest and lowest levels in α exponentially. Generally, the lower level a concept/role has, the more general this concept/role should be.

Chapter 4

Conclusion and Future Work

This chapter summarizes the thesis, discusses its findings and contributions, points out the limitations of the current work, and outlines directions for future research. A number of properties of kernel contraction in $\mathcal{EL}$ have been demonstrated and the preference between different kernel contractions has been discussed. However, still many extensions of this research deserve further consideration.

The chapter is divided into three sections. Section 4.1 is a summary of the thesis. Section 4.2 presents a discussion of the contribution and limitations of the current work and discusses the future work, and finally Section 4.3 brings the thesis to a conclusion.

4.1 Thesis Summary

The work described in this thesis has been concerned with the description logic $\mathcal{EL}$ and kernel contraction in it.

Chapter 1 and 2 introduced the background of this thesis. Description logic, especially the description logic $\mathcal{EL}$ is introduced. On the other hand, the basic ideas of contraction and kernel contraction are also introduced. After that, we combine $\mathcal{EL}$ with kernel contraction and give the definition to kernel contraction in $\mathcal{EL}$. Last but not least we discuss some related works in both areas.

In chapter 3, which was the main part of this thesis, we presented our contributions toward kernel contraction in $\mathcal{EL}$.

First we started to analyze kernel contraction in $\mathcal{EL}$ in two ways. One was to examine the rationality of kernel contraction in $\mathcal{EL}$, which was discussed in sections 3.1-3.3. The

other was to find ways to measure the rationality a specific kernel contraction, which was discussed in sections 3.4-3.6.

We presented algorithms to actually generate a kernel contraction. A kernel contraction generating algorithm included two parts: an algorithm to find all the kernels and an algorithm to form the incision function based on all the kernels. We could generate kernels using axiom pinpointing, then we apply one of the algorithms in section 3.3 to form the set of beliefs to give up, which determine a kernel contraction. We also checked if kernel contraction in $\mathcal{EL}$ satisfied AGM postulates and also some other worth discussing properties like relevance postulate. We proved the satisfaction and gave counter examples for the dissatisfaction one by one. We proved that kernel contraction in $\mathcal{EL}$ satisfied 5 out of 6 basic AGM postulates ($(K-1), (K-2), (K-3), (K-4), (K-6)$ postulates). As a result, we can say that kernel contraction in $\mathcal{EL}$ is a rational contraction.

Overall, kernel contraction in $\mathcal{EL}$ is a good way to contract a belief. However, in terms of rationality, there are still big differences between different kernel contractions which come from different incision functions. And we cannot calculate all of them because there can be exponential many different kernel contractions when we want to contract a certain belief from a certain knowledge base. So it would be helpful if we could find an algorithm which ensures the kernel contraction is good enough, or at least find some quantitative criteria to measure a certain kernel contraction. We proposed some new concepts like *localization* and *specificity*. Both *localization* and *specificity* are about quantifying the rationality of a certain kernel contraction. Localization was based on the idea that errors should be close to each other and specificity came from the idea that general concepts/roles are more reliable than specific concepts/roles. We gave some functions to describe them according to their properties. The kernel contraction generation is equivalent to *hitting set problem* and *set cover problem*. In the last section of the contribution part, we introduced some interesting progress in the hitting set problem and set cover problem and then tried to apply them to kernel contraction.

4.2 Recommendation for Future Work

While this thesis has analysed kernel contraction in $\mathcal{EL}$ in many aspects, many opportunities for extending the scope of this thesis remain. This section presents some of these directions.

One such direction would be to investigate the concept of *localization* further. As we

mentioned in in section 3.5, the mistakes in a knowledge base tend to be closed to each other. *Localization* could be applied not only on kernel contraction in $\mathcal{EL}$ but also general belief contraction. In this thesis, we used clustering numbers to describe localization in which adjacent GCIs are defined as GCIs with at least one common concept. This clustering number can describe localization to some extent, but it is limited in some cases especially when the clustering numbers of most of the kernel contractions are 1. Probably this is not the best option. Perhaps by finding a better measurement of localization, such as something related to the distance on the classification graph, we could obtain more reliable results.

There is also a similar direction for improving which is about *specificity* in section 3.6. Specificity is based on the idea that general concepts/roles are more reliable than specific concepts/roles. In this thesis, we use $d(v)$ (degree of vertex) to measure how specific a concept/role is. We measure the specificity of a GCI by the average of all the degrees of concept/role in this GCI. There are of course other possible ways to do it, such as to calculate vertex degrees of the concepts/roles in a different way, or to use something related to hierarchy to describe the concepts/roles instead of using vertex degrees.

Another possibility would be to develop weighted kernel contraction further. We applied weighted hitting set problem to kernel contraction and it became weighted kernel contraction. This is definitely an interesting topic. Sometimes we do not want to treat all different GCIs equally as some of them may be well-known truths and some may be not. In this thesis, we introduced the concept "weighted kernel contraction" and gave some ideas about what the weights can be. Weighted kernel contraction itself is not yet fully developed.

4.3 Conclusion

Kernel contraction in $\mathcal{EL}$ is one kind of kernel contraction developed on the lightweight description logic $\mathcal{EL}$. With the analysis and the algorithm given in this thesis, we can see that kernel contraction in $\mathcal{EL}$ is practical and also rational overall. Meanwhile, this thesis has presented some possible ways like localization and specificity to decide the preference between any specific kernel contractions.

Bibliography

[1] Franz Baader. *The description logic handbook: theory, implementation, and applications*. Cambridge: Cambridge University Press, 2003.

[2] Franz Baader. Whats new in description logics. *Informatik-Spektrum*, 34(5):434–442, 2011.

[3] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *International Joint Conference on Artificial Intelligence*, volume 19, page 364. Lawrence Erlbaum Associates Ltd, 2005.

[4] Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice. In *Proceedings of the 2005 International Workshop on Methods for Modalities (M4M-05)*, page 5, 2005.

[5] Franz Baader, Carsten Lutz, and Anni-Yasmin Turhan. Small is again beautiful in description logics. *KI-Künstliche Intelligenz*, 24(1):25–33, 2010.

[6] Franz Baader and Rafael Peñaloza. Axiom pinpointing in general tableaux. In *Automated Reasoning with Analytic Tableaux and Related Methods*, pages 11–27. Springer, 2007.

[7] Franz Baader, Rafael Penaloza, and Boontawee Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}$. In *KI 2007: Advances in Artificial Intelligence: 30th Annual German Conference on AI, KI 2007, Osnabrück, Germany, September 10-13, 2007, Proceedings*, volume 4667, page 52. Springer, 2007.

[8] Franz Baader and Boontawee Suntisrivaraporn. Debugging snomed ct using axiom pinpointing in the description logic $\mathcal{EL}^+$. In *Proceedings of the International Conference on Representing and Sharing Knowledge Using SNOMED (KR-MED 2008), Phoenix, Arizona*. Citeseer, 2008.

[9] Richard Booth, Thomas Meyer, and Ivan José Varzinczak. First steps in $\mathcal{EL}$ contraction. *ARCOE-09*, page 16, 2009.

[10] Sebastian Brandt. Polynomial time reasoning in a description logic with existential restrictions, gci axioms, and-what else? In *ECAI*, volume 16, page 298, 2004.

[11] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, and Riccardo Rosati. Ontologies and databases: The dl-lite approach. In *Reasoning Web. Semantic Technologies for Information Systems*, pages 255–356. Springer, 2009.

[12] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The dl-lite family. *Journal of Automated Reasoning*, 39(3):385–429, 2007.

[13] Peter Gärdenfors. *Belief revision*, volume 29. Cambridge University Press, 2003.

[14] Sven Ove Hansson. *A Textbook of Belief Dynamics*. Applied Logic Series. Kluwer Academic Publishers, 1999.

[15] Carsten Lutz and Frank Wolterinst. Conservative extensions in the lightweight description logic $\mathcal{EL}$. In *Automated Deduction–CADE-21*, pages 84–99. Springer, 2007.

[16] Thomas Meyer, Kevin Lee, Richard Booth, and Jeff Z Pan. Finding maximally satisfiable terminologies for the description logic alc. In *Proceedings of the National Cconference on Artificial Intelligence*, volume 21, page 269. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[17] Rafael Peñaloza and Barış Sertkaya. On the complexity of axiom pinpointing in the $\mathcal{EL}$ family of description logics. In *Twelfth International Conference on the Principles of Knowledge Representation and Reasoning*, 2010.

[18] Guilin Qi and Jianfeng Du. Model-based revision operators for terminologies in description logics. In *Proc. of IJCAI*, volume 2009, pages 891–897, 2009.

[19] M Ribeiro, Renata Wassermann, Grigoris Antoniou, Giorgos Flouris, and Jeff Pan. Belief contraction in web-ontology languages. In *Proceedings of the 3rd International Workshop on Ontology Dynamics, IWOD*, 2009.

[20] Roberto Sebastiani and Michele Vescovi. Axiom pinpointing in lightweight description logics via horn-sat encoding and conflict analysis. In *Automated Deduction–CADE-22*, pages 84–99. Springer, 2009.

[21] Boontawee Suntisrivaraporn. The cel manual.

[22] Frank van Harmelen, Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter. *Handbook of Knowledge Representation*. Elsevier Science, San Diego, USA, 2007.

[23] Vijay V Vazirani. *Approximation algorithms*. springer, 2001.

[24] Zhe Wang, Kewen Wang, and Rodney Topor. Revision of dl-lite knowledge bases. In *Proceedings of the Description Logics Workshop*, 2009.

[25] Zhe Wang, Kewen Wang, and Rodney W Topor. A new approach to knowledge base revision in dl-lite. In *Proc. of AAAI*, volume 10, pages 369–374, 2010.

[26] Zhe Wang, Kewen Wang, and RW Topor. Revising general knowledge bases in description logics. In *Proc. of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2010.