論文 / 著書情報
Article / Book Information

題目(和文)	選好に基づく推論のための論理プログラミングによる極小限定の計算
Title(English)	Computing circumscription for preference-based reasoning by logic programming
著者(和文)	若木利子
Author(English)	
出典(和文)	学位:博士(理学), 学位授与機関:東京工業大学, 報告番号:甲第3956号, 授与年月日:1998年12月31日, 学位の種別:課程博士, 審査員:
Citation(English)	Degree:Doctor (Science), Conferring organization: Tokyo Institute of Technology, Report number:甲第3956号, Conferred date:1998/12/31, Degree Type:Course doctor, Examiner:
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Computing Circumscription for Preference-based Reasoning by Logic Programming

選好に基づく推論のための 論理プログラミングによる極小限定の計算

by

Toshiko Wakaki

A dissertation
for the degree of Doctor of Science
submitted to
Tokyo Institute of Technology

December, 1998

Abstract

In everyday life, we are always confronted with the necessity to make a decision under the incomplete information. In such a situation, we often make a decision based on commonsense knowledge, or default rules which are 'almost always' true, with a few exceptions.

So far, it has been noticed that human commonsense reasoning has crucial characteristics such as nonmonotonicity and usage of preferences. Nonmonotonicity cannot be formalized by means of classical logic since classical logic is monotonic. On the other hand, for making an adequate decision, we often encounter the necessity to use the knowledge of preferences which decide which defaults will be selected first in presence of conflicts between defaults. Such knowledge of preferences is classified into two categories such that static preferences and dynamic ones.

The purpose of this thesis is to establish various methods to realize a computer-aided intelligent system which can treat various human commonsense reasoning, especially preference-based reasoning.

First, since circumscription is the powerful formalization of nonmonotonic reasoning which can incorporates static preferences in its logical framework, we aim at computing circumscription for preference-based reasoning. So, in this thesis, we propose two methods of compiling parallel circumscription as well as prioritized circumscription into two classes of logic programs, i.e. a stratified logic program and an extended logic program. Each method is based on the semantic relationship between circumscription and the corresponding transformed logic program.

Secondly, to treat dynamic preferences correctly is recently required in commonsense reasoning especially such as the legal reasoning. In this thesis, we investigate a way to treat dynamic preferences within the framework of circumscription by logic programming.

Thirdly, priorities among the conflict default rules, so-called static preferences, are often required to be found out in order to derive a conclusion of the commonsense reasoning. So, we present a method to find priorities of circumscription policy as a skeptical explanation in abduction. In addition, we propose a method of finding dynamic preferences as hypotheses based on a similar way of abductive reasoning which is particularly required for legal argumentation.

Acknowledgments

First of all, I would like to thank sincerely my supervisor Katsumi Nitta and sub-supervisor Seiichiro Sakurai for invaluable comments and suggestions on this thesis.

I would like to express my deep gratitude to Ken Satoh of Hokkaido University who was a researcher of Fujitsu Laboratories, Ltd.. At the start of this research, his dissertation for the degree of Doctor of Science published in 1992 stimulated my interest very much, and he introduced me to the hot topics of the research field of nonmonotonic reasoning and logic programming. Without his guide, sharp comments and interesting joint researches, I could never have accomplished this thesis.

I am most grateful to Jun-ichi Tanahashi of Fujitsu Laboratories, Ltd. for his indispensable support and suggestions. He gave me a chance and provided the stimulating environments to do the initial stage of this research with Ken Satoh at Fujitsu Laboratories.

I also deeply thank Susumu kunifuji of Japan Advanced Institute of Science and Technology, Hokuriku for his indispensable support and encouragement to continue my research activities at Tokyo Institute of Technology.

Last, but not least, I would like to thank Osamu Imanaka who gave me a vision and an opportunity to start this research as my manager.

Contents

Chapter 1

Introduction

1.1 Motivation

The aim of this thesis is to establish methods to realize a computer-aided intelligent system which can treat various human commonsense reasoning.

So far, it has been noticed that commonsense reasoning has two crucial characteristics such as *nonmonotonicity* and usage of $preferences$ discussed below.

In everyday life, humans are always faced with the necessity to make a decision where only *incomplete* $information$ is available. In such a situation, we often make a decision based on the so-called *commonsense*, or $default\ rules$ which are 'almost always' true, with a few exceptions. The following illustrates the typical human common sense reasoning.

Suppose that in an airport, someone is waiting for his friend who is to arrive there by airplain. He believes that his friend will arrive there just at the arrival time 12:00 of the airplain written in the time table of flight. But if such new information that his friend's airplain is hijacked is announced, he no longer believes that his friend will arrive there at 12:00 and returns home.

This kind of change of decision cannot be formalized by a classical first-order predicate logic. This is because the language of the predicate logic has been created primary for the purpose of formalizing mathematics and it dose not provide any means for talking about what is "generally true"with a few exceptions. In classical logic, once we derive some conclusion, then we no longer retract the conclusion. This feature of classical logic is called *monotonic*. On the other hand, the human commonsense reasoning is *nonmonotonoic* as it is shown in the above example that his first decision should be retracted finally as a rare daily life case.

The first movement of logical foundation of *nonmonotonoic reasoning* is in early 80's, and some important formalisms such as $default\ theory$ [Reiter, 1980], *circumscription* [McCarthy, 1980] and so on, have been proposed in order to explain the human commonsense reasoning based on logic theoretically.

Circumscription is a formalized $rule\ of\ conjecture$ that can be used by a person or program for 'jumping to certain conclusions', and can be used along with the $rules\ of\ inference$ of the first-order logic. It is recognized that *parallel*

circumscription is one of the most powerful formalizations of the *closed world assumption* that a formula (assertion) can be implicitly assumed to be false unless there is no evidence to the contrary. About this assumption, it is well-known as a practical use of nonmonotonic reasoning that the closed world assumption effectively works in the relational database system[Reiter, 1978] and has been adopted in its commercial systems.

A form of circumscription is syntactically represented by a second-order formula whereas its definition is given by a model theory semantically. Therefore circumscription has major advantages over the other nonmonotonic formalizations that it is based on well-developed mathematical theory of the *classical predicate logic*. However, if we try to make use of circumscription in order to make a computer-aided intelligent system such as expert systems, deductive database systems and so on which can handle the human commonsense reasoning, there arises a difficult problem to compute circumscription because a second-order predicate logic is not complete.

On the other hand, in the last decade, the research of logic programming has been explored progressively. In this research field, several syntactically different classes of logic programs have been proposed whose declarative meaning is given as their semantics. Besides, recently, semantic relationships between nonmonotonic reasoning and logic programs have been clarified though in the beginning, both researches of nonmonotonic reasoning and logic programming have been independently pursued. For example, Przymusinski [1987] shows that the declarative semantics of a stratified logic program is equivalent to the semantics of prioritized circumscription without variable predicates and fixed predicates. Gelfond and Lifschitz [1991] show the semantic relationship between an answer set of an extended logic program and an extension of a default theory. On the other hand, they [Gelfond and Lifschitz, 1988b] study the possibility of reducing some special cases of circumscription to logic programming. Their method allows to compute some very restricted class of prioritized circumscription by means of compiling circumscriptive theories into stratified logic programs. Though the method is efficient, its applicable class is too limited to use for practical purpose.

So, in order to realize a computer-aided nonmonotonic reasoning system, we intend to advance further Gelfond and Lifschitz's compiling approach to compute circumscription. Thus the first subject of this thesis is to explore a method of computing a wide class of circumscription by means of compiling circumscriptive theories into logic programs. Regarding this subject, there exists a theoretical problem that we should find the semantic relationship between circumscription and the corresponding transformed logic program.

Another important aspect of commonsense reasoning is that human usually takes into account $preferences$ among default rules when various conflicting commonsense knowledge exists. For example, we encounter the necessity to use preferences between commonsense knowledge as follows [Reiter and Criscuolo, 1981]:

(r1) *Typically, students are unmarried.*

(r2) *Typically, adults are married.*

(f1) *Students are adult.* (f2) *Peter is a student.*

According to commonsense knowledge r1, we consider that Peter may be unmarried since he is a student. On the other hand, according to r2, we consider that Peter may be married since he is adult because a student is adult and he is a student. Thus there occurs conflict between commonsense knowledge r1 and r2. But under such the situation, we generally conclude that Peter is unmarried because r1 is more preferable than r2. In *prioritized circumscription*, preferences are explicitly represented in form of *priorities* to formalize the human commonsense reasoning which needs the preference information. It is based on the mechanism of selecting more preferable models by pre-order in its logical foundation. Priorities incorporated in prioritized circumscription, however, are regarded as *static preferences* because they are specified outside the logical theory to which they apply and cannot be inferred from preferences within the theory. Static preferences are often insufficient in commonsense reasoning such as legal reasoning, etc. In legal domain, for example, there are well-known preferences such as general principles of Lex Posterior, Lex Superior and Lex Supecialis. For instance, a principle of Lex Posterior denotes that the newer law is preferable, that of Lex Supecialis denotes that the more special law is preferable and they give precedence to newer laws and more special laws respectively, each of which plays a role to resolve conflicts between laws. Moreover, in legal domain, sometimes not only meta-preferences between preferences like that Lex Supecialis is preferable than Lex Posterior, but also meta-meta preferences between meta-preferences and so on are also used and reasoned about in order to resolve the conflicts between laws. Then such preferences are called *dynamic preferences* because preferences themselves should be reasoned to infer an object-level conclusion. At present, it is required to extend the formalizations of the conventional nonmonotonic reasoning so that they can treat dynamic preferences correctly in their reasoning system and some approaches have been proposed. Brewka [1996] was the first to integrate dynamic preferences into default logic and logic programming.

The second subject of this thesis is to propose another new method of reasoning about dynamic preferences. Our approach discussed in the subsequent chapter is to integrate dynamic preferences into the framework of circumscription and logic programming, which makes use of the achievements of the first subject mentioned above.

In commonsense reasoning, sometimes it is not only required to reason about dynamic preferences but also to find priorities (static preferences) as well as dynamic preferences as *hypotheses* automatically. For example, it is historically well-known that the static preferences of the notorious Yale Shooting Problem [Hanks and McDermott, 1987] were found manually in order to explain the human reasoning about the problem. As another example, dynamic preferences as hypotheses are required for disputers to derive his desiring goal in order to win in legal argumentation of the court. Nevertheless, so far, the problem to find such

preferences has neither been recognized significantly nor any method has been proposed to solve the problem. As the technical background about this problem, we address the recent research on *abductive reasoning* a little.

Abduction is one of three distinguished forms of reasoning: Deduction, Induction and Abduction whose notions were firstly identified by Peirce[Peirce, 1932]. Abduction can be regarded as non-monotonic reasoning, because hypotheses which were consistent with a knowledge base may become inconsistent when new knowledge is added. In the last decade, not only the relationships between abduction and various non-monotonic formalisms (e.g. normal default theory, circumscription) have been clarified but also abduction has been studied in terms of logic programming by researchers and various abductive proof procedures have been proposed to calculate an *explanation* as hypotheses for a given *observation*.

Therefore the third subject and objective is to present a method to find out static as well as dynamic preferences as hypotheses (i.e. explanation) to infer a desired goal as an observation by making use of the recent profound results of abductive logic programming as well as the research of computing circumscription by logic programming which is contribution of the first subject.

Lastly, we expect that, the expressive power of the formalism of circumscription as well as the applicability and efficiency w.r.t. our methods to compute circumscription developed for the first subject can be evaluated by applying them to the real world problems taken into account w.r.t. the second and third subjects.

1.2 About the Thesis

This thesis consists of four parts.

1. Survey of existing non-monotonic reasoning system and logic programming.
2. Computing prioritized circumscription by logic programming.
3. Reasoning dynamic preferences in circumscriptive theory.
4. Finding priorities and dynamic preferences as hypothesis.

The first part is Chapter 2 where we survey formalizations of the existing nonmonotonic reasoning system such as default theory, circumscription as well as the syntax and semantics of various classes of logic programs. They are referred in the subsequent chapters. The relationships between different formalizations of nonmonotonic reasoning as well as the relationships between a formalism of nonmonotonic reasoning and a class of logic programs are also shown.

The second part consists of Chapter 3 and Chapter 4. In Chapter 3, we present our first method of compiling prioritized circumscription into a stratified logic program in oder to expand the applicable class of Gelfond and Lifschitz method [Gelfond and Lifschitz, 1988a]. In Chapter 4, we present our second method of

compiling a wide class of circumscription including parallel circumscription as well as prioritized circumscription into extended logic programs. From now on, we sometimes call this second method "Wakaki and Satoh's method".

The third part is Chapter 5. In this chapter, we propose a method to reason dynamic preferences in circumscriptive theory, which are computed by logic programming based on our compiling method shown in Chapter 4.

The fourth part consists of Chapter 6 and Chapter 7. In Chapter 6, we provide a method of finding priorities, i.e. a part of circumscription policy automatically in order to infer a conclusion of the commonsense reasoning as a theorem of circumscriptive theory. In Chapter 7, we provide a method for finding dynamic preferences as hypothesis to derive a desired conclusion of disputers, which are especially required in legal argumentation.

Finally, Chapter 8 gives a summary of contributions of this thesis and discuss future researches about nonmonotonic reasoning.

Almost parts of this thesis are based on the papers which have already been published. Chapter 3 is based on the paper [Wakaki and Satoh, 1995] titled "Computing Prioritized Circumscription by Logic Programming" which was presented at 12th International Conference on Logic Programming (ICLP) in 1995 as well as the Journal paper [Wakaki and Satoh, 1996]. Chapter 4 is based on the paper [Wakaki and Satoh, 1997] titled "Compiling Prioritized Circumscription into Extended Logic Programs," which was presented at Fifteenth International Joint Conference on Artificial Intelligence (IJCAI) in 1997 as well as the Journal paper which will appear in [Wakaki and Satoh, 1999]. Chapter 5 is based on the paper [Wakaki, Satoh and Nitta, 1997] titled "Reasoning about Dynamic Preferences in Circumscriptive Theory by Logic Programming" which originally appeared in International Journal on Advanced Computational Intelligence(JACI), 1997. Chapter 6 is based on the paper [Wakaki et al., 1998] titled "Finding Priorities of Circumscription Policy as a Skeptical Explanation in Abduction" which originally appeared in Journal of IEICE Transactions on Information and Systems, 1998.

1.3 Contributions

The contributions of this thesis are as follows.

1. **We provide new methods of computing circumscription by logic programming.**

   We present two methods to compute circumscription as follows. One is an extension of Gelfond and Lifschitz's method and another is based on our original idea.

   (a) **We provide a method of compiling prioritized circumscription into a stratified logic program**

Gelfond and Lifschitz [Gelfond and Lifschitz, 1988a] studied the possibility of reducing some special class of circumscription to logic programming and showed a way to compile circumscriptive theories for some restricted class of prioritized circumscription into stratified logic programs. But there are such difficulties in their method that a given circumscriptive theory cannot always be compiled successfully due to strong assumptions about the syntax of a circumscriptive theory. So, we extend their method in order to expand the applicable class of their method with keeping its computational efficiency. Our idea is to transform a given circumscription into a logically equivalent one in which such difficulties disappear. In this thesis, we show it can be done by making use of Lifschitz's result that some parallel circumscription can be replaced by an equivalent first-order theory. As a result, some class of prioritized circumscription, which cannot be handled by Gelfond and Lifschitz's method, can be compiled into logic programs by our method.

(b) **We provide a method of compiling a wide class of circumscription into extended logic programs**

The applicable class of our first method mentioned above is still limited for practical use though it is efficient. So, we propose our second method of compiling circumscription into Extended Logic Programs which is widely applicable to a class of parallel circumscription as well as a class of prioritized circumscription. In this thesis, we show theoretically that circumscription whose theory contains both the domain closure axiom and the uniqueness of names axioms and does not contain function symbols can always be compiled into an extended logic program Π, so that, whether a ground literal is provable from circumscription or not, can always be evaluated by deciding whether the literal is true in all answer sets of Π, which can be computed by running Π under the existing logic programming interpreter. In this thesis, we refer to this method as "Wakaki and Satoh's method".

2. **We provide a method to reason dynamic preferences.**

To treat dynamic preferences correctly is inevitably required in the legal reasoning. The traditional formalizations of nonmonotonic reasoning, however, cannot treat dynamic preferences in their logical theory. In this thesis, we present a method which enables us to handle some class of dynamic preferences in the framework of circumscription and to compute consistently its meta-level and object-level reasoning by expressing them in an extended logic program. This is achieved on the basis of policy axioms and priority axioms which permit to describe circumscription policy by axioms and play a role to intervene between the meta-level and object-level reasoning. Not only the information about preferences among rules and meta-rules but also

inference rules from the preferences to priorities are represented by a normal logic program. Thus priorities can be derived from the preferences dynamically, which allows us to compute the object-level circumscriptive theory by logic programming based on Wakaki and Satoh's method. The comparisons between our approach and others are also discussed.

3. **We propose methods of finding priorities as well as dynamic preferences as hypotheses.**

   (a) **We propose a method to find priorities of circumscription policy as a skeptical explanation in abduction**

   Priorities among rules, i.e. static preferences, are often required to be found out in order to derive a conclusion of the commonsense reasoning. In this thesis, first we present the bottom-up and top-down abduction procedures to compute skeptical explanations and secondly show that priorities of circumscription to infer a desired theorem can be abduced as a skeptical explanation in abduction. In our approach, the required priorities can be computed based on the procedure to compute skeptical explanations provided in this thesis as well as Wakaki and Satoh's method of compiling circumscription into extended logic programs. The method, for example, enables us to automatically find the adequate priority w.r.t. the Yale Shooting Problem to express a human natural reasoning in the framework of circumscription.

   (b) **We propose a method of finding dynamic preferences as hypotheses for legal argumentation**

   In commonsense reasoning such as legal reasoning, it is inevitably required not only to treat dynamic preferences adequately but also to find out dynamic preferences as hypotheses among rules or meta-rules in order to infer a desired conclusion which give the disputers the advantage in the legal argumentation. In this thesis, we present a method which enables us to to find out such dynamic preferences as hypotheses. We show that, when a desired conclusion is not derived due to the lack of dynamic preferences, such preferences as hypotheses can be found out as a skeptical explanation of a desired conclusion as observation w.r.t. abductive framework proposed in our method. Preferences as hypothesis can be computed by means of abductive logic programming based on the meta-level reasoning to treat dynamic preferences as well as a method of finding priorities mentioned above.

Chapter 2

Nonmonotonic Reasoning and Logic Programming

2.1 Introduction

In this chapter,we review firstly existing frameworks for nonmonotonic reasoning along with abduction, and next, several classes of logic programs as well as relationships between nonmonotonic reasoning and logic programming. Since the purpose of this thesis is not to review a lot of recent researches about nonmonotonic reasoning and logic programming(LPMNR), in this chapter, we only concentrate on such restricted topics about these that will be needed or referred in subsequent chapters.

2.2 Nonmonotonic Reasoning and Abduction

Formalisms for nonmonotonic reasoning were proposed in order to explain the human commonsense reasoning under incomplete information. If only insufficient information is available for decision, human uses commonsense to make a decision. Human commonsense reasoning, however, often surprisingly works well though the results from the reasoning is not always true because commonsense is a collection of expressions of standard cases which hold generally except a few exceptions. We cannot formalize it in a classical logic since classical logic is monotonic as is mentioned in Chapter 1.

This section reviews two major formalisms for nonmonotonic reasoning: $default$ *logic*[Reiter, 1980] and *circumscription*[McCarthy, 1980], [Lifschitz, 1985]. Each of default logic and circumscription extends the classical first-order predicate logic, but in a different way. [Satoh, 1992] divides nonmonotonic reasoning systems into two categories; one is *consistency*-based nonmonotonic reasoning systems and the other is *minimal-model*-based system. Default logic belongs to the former, and circumscription to the latter. Default logic introduces $inference\ rules$, each called $default$, and extends a first order theory by using them meta-theoretically based on the idea of *consistency* such that if we can assume some knowledge without any

inconsistency, then we assume that knowledge. On the other hand, circumscription extends a first order theory with using a *second-order axiom* expressing a kind of *minimality* principle that specifies which models of a given first order theory are most preferable. Though these two formalisms have quite different backgrounds, we will see in a later section (Section 2.2.4) that there is an interesting relationship between them. This relationship is very important for us, because it becomes one of the significant foundations of our method of compiling circumscription into extended logic programs discussed in Chapter 4.

2.2.1 Circumscription

Circumscription was proposed by J. McCarthy [1980] for formalizing non-monotonic aspects of common sense knowledge and reasoning. Soon after, a new form of circumscription called *parallel circumscription* was introduced by Lifschitz [1985]. On the other hand, another new form of circumscription called *prioritized circumscription* was also proposed by Lifschitz [1985] and McCarthy [1986] to solve the multiple extension problem in inheritance system, where different kinds of abnormality should be taken into account and it would lead to conflict if they were minimized (i.e. circumscribed) in parallel.

In this subsection, we briefly review definitions and properties of parallel circumscription as well as prioritized circumscription according to Lifschitz [1985].

Definition 2.1 (*About a second order language*)

A *second order language* is defined, just as a first order language, by sets of n-ary *function constants* and n-ary *predicate constants* ($n \geq 0$). In the second order language, we have also n-ary *function variables* and n-ary *predicate variables* besides *object variables*. *Object variables* and *Object constants* are identified with function variables and function constants of arity 0. Both function and predicate variables can be bounded by quantifiers. A *sentence* is a formula without free variables.

Definition 2.2 (*About semantics of a second order language*)

An interpretation M for a second order language L consists of

(i) a non-empty universe $|M|$,

(ii) functions from $|M|^n$ to $|M|$ representing the function constants and

(iii) subsets of $|M|^n$ representing the predicate constants.

Function variables range over arbitrary functions from $|M|^n$ to $|M|$, and predicate variables range over arbitrary subsets of $|M|^n$. For any constant K, we denote the object (function or set) representing K in M by $M[K]$. A model of a second order formula F is any interpretation M such that F is true in M. A model of a sentence A is any interpretation M such that A is true in M.

Definition 2.3 ($\leq$, $=$, $<$) [Lifschitz, 1985]

If U,V are n-ary predicates, then $U \leq V$ stands for $\forall x(U(x) \supset V(x))$. Thus $U \leq V$ expresses that the extension of U is a subset of the extension of V. If this notation is applied to tuples $U =< U_1, \ldots, U_m >$ and $V =< V_1, \ldots, V_m >$, assuming that they are similar (i.e., that U_i and V_i have the same arity): $U \leq V$ stands for

$$U_1 \leq V_1 \wedge \ldots \wedge U_m \leq V_m.$$

Furthermore, $U = V$ stands for $U \leq V \wedge V \leq U.$, and $U < V$ stands for $U \leq V \wedge \neg(V \leq U)$. If m=1 then $U < V$ expresses that the extension of U is a proper subset of the extension of V.

Now, parallel circumscription is defined as follows.

Definition 2.4 (Parallel Circumscription)[Lifschitz, 1985]
Let $A(P, Z)$ be a sentence, where P is a tuple of predicate constants $P_1, \ldots, P_m$ and Z is a tuple of function and/or predicate constants. P and Z are disjoint each other. The predicates in P are said to be *minimized* and those in Z to be *variables*. Q denotes a tuple of the rest predicates in $A(P, Z)$, called the $fixed$ predicates. Then, the *circumscription of P in A(P,Z) with variable Z* is defined as follows:

$$Circum(A; P; Z) \stackrel{def}{=} A(P, Z) \wedge \neg\exists pz(A(p, z) \wedge p < P).$$

Here p, z are tuples of variables which have the same arity to P, Z respectively. $p < P$ denotes $(p \leq P) \wedge \neg(P \leq p)$ where $p \leq P$ stands for

$$\bigwedge_{i=1}^{m} \forall x(p_i(x) \supset P_i(x)).$$

We write $Circum(A; P)$ when the last argument Z in $Circum(A; P; Z)$ is empty. Intuitively, $Circum(A; P; Z)$ is intended to minimize the number of objects satisfying P, even at the expense of allowing more or different objects to satisfy Z.

The model-theoretic meaning of circumscription is expressed by $\leq^{P;Z}$ which is a pre-order (i.e., a reflexive and transitive relation) on the class of all models of $A(P, Z)$ as follows.

Definition 2.5 [Lifschitz, 1985]
For any two interpretations M_1,M_2, we write $M_1 \leq^{P;Z} M_2$ if

$$(1)|M_1| = |M_2|,$$
$$(2)M_1[K] = M_2[K], \ for\ every\ constant\ K\ not\ in\ P, Z,$$
$$(3)M_1[P_i] \subset M_2[P_i], \ for\ every\ P_i\ in\ P.$$

Furthermore, we also write $M_1 <^{P;Z} M_2$ if $M_1 \leq^{P;Z} M_2$, but not $M_2 \leq^{P;Z} M_1$.

Definition 2.6 (Minimality condition with respect to $\leq^{P;Z}$)
An interpretation M is *minimal* in a class S of interpretations with respect to $\leq^{P;Z}$ if $M \in S$ and there is no interpretation $M' \in S$ such that $M' <^{P;Z} M$.

The models of parallel circumscription is given by the following theorem.

Theorem 2.7 [Lifschitz, 1985]
An interpretation M is a model of $Circum(A; P; Z)$ iff M is minimal in the class of models of A with respect to $\leq^{P;Z}$.

This means that if a formula G is inferred from a second order formula $Circum(A; P; Z)$ (i.e. G is a theorem of $Circum(A; P; Z)$), then G is true in all minimal models of A. However, the converse does not always hold since in general, it is not computable to deduce every formula which is true in all minimal models of A because the second order predicate logic is not complete. Moreover, if there exists no minimal models of A (i.e. no models of $Circum(A; P; Z)$), then $Circum(A; P; Z)$ leads to contradiction.

Lifschitz shows some classes of circumscription called *solitary formula* and *separable formula* [Lifschitz, 1985] each of which can be reduced into a first-order formula as follows.

Definition 2.8 (Solitary formula) [Lifschitz, 1985]
Let P be a tuple of predicate constants $P_1, \ldots, P_m$. $A(P)$ is called a solitary formula if it can be written in the form:

$$N(P) \wedge (Q \leq P),$$

where $N(P)$ contains no positive occurrences of $P_1, \ldots, P_m$, and Q is a tuple of predicates not containing $P_1, \ldots, P_m$. Then the result of parallel circumscription is given by the following formula:

$$Circum(N(P) \wedge (Q \leq P); P) \equiv N(Q) \wedge (Q = P),$$

where $Q = P$ denotes $(Q \leq P) \wedge (P \leq Q)$.

In some applications of circumscription, different kinds of abnormal (minimized) predicates are required to assign different priorities (i.e. static preferences) for minimization. To support this requirement, another circumscription called *prioritized circumscription* is proposed [Lifschitz, 1985] in which a tuple P of minimized predicates is broken into disjoint parts $P^1, \ldots, P^k$, and members of P^i are assigned a higher priority than the members of P^j for $i < j$ as follows:

Definition 2.9 (Prioritized circumscription) [Lifschitz, 1985]
Let $A(P^1, P^2, \ldots, P^k, Z)$ be a sentence, where $P^1, P^2, \ldots, P^k, Z$ are disjoint tuples of predicate constants. Then, *prioritized circumscription w.r.t $P^1, P^2, \ldots, P^k$ with variable Z* is defined as follows:

$$\begin{aligned} &Circum(A; P^1 > P^2 > \ldots > P^k; Z) \stackrel{def}{=} \\ &\quad A(P^1, P^2, \cdots, P^k, Z) \wedge \neg\exists p^1 \exists p^2 \ldots \exists p^k \exists z \\ &\qquad (A(p^1, p^2, \ldots, p^k, z) \wedge \langle p^1, p^2, \ldots, p^k\rangle \prec \langle P^1, P^2, \ldots, P^k\rangle), \end{aligned}$$

where $\langle p^1, p^2, \ldots, p^k\rangle \prec \langle P^1, P^2, \ldots, P^k\rangle$ is an abbreviation of the following formula:

$$\bigwedge_{i=1}^{k}((\bigwedge_{j=1}^{i-1}(p^j = P^j)) \supset (p^i \leq P^i)) \wedge \neg \bigwedge_{i=1}^{k}((\bigwedge_{j=1}^{i-1}(P^j = p^j)) \supset (P^i \leq p^i)),$$

where $p^j = P^j$ stands for $(p^j \leq P^j) \wedge (P^j \leq p^j)$.

Theorem 2.10 [Lifschitz, 1985] *Prioritized circumscription is reducible to the conjunction of k parallel circumscriptions as follows:*

$$Circum(A; P^1 > \ldots > P^k; Z) \equiv \bigwedge_{i=1}^{k} Circum(A; P^i; P^{i+1}, \ldots, P^k, Z)$$

This means that prioritized circumscription is reduced to parallel circumscription when $k = 1$.

There is an interesting property that it is possible to transform syntactically any parallel circumscription as well as prioritized circumscription using fixed predicates into an equivalent one which does not use fixed predicates [de Kleer and Konolige, 1989]. When defining the concept of equivalence in the following theorem, the requisite notion is that of *conservative extension*. Let L be the language of a second-order sentence S; we say that a sentence S' is a conservative extension of S just in case every sentence of L is true in all models of S exactly when it is true in all models of S'.

Theorem 2.11 [de Kleer and Konolige, 1989, Corollary 2] *Let Q be a tuple of predicates $q_1, \ldots, q_n$ of A not contained in P or Z, and Q' be a tuple of similar predicates $q'_1, \ldots, q'_n$. Then*

$$Circum(A \wedge \bigwedge_{i=1}^{n} \forall x(q_i(x) \equiv \neg q'_i(x));\ P, Q, Q';\ Z)$$

is a conservative extension of $Circum(A; P; Z)$.

In [Lifschitz, 1987a], Lifschitz defines a *circumscriptive theory* as a theory which represents a formalism of circumscription (e.g. [McCarthy, 1980], [Lifschitz, 1985]) in a sense that such a theory is completely determined by its axioms, without any additional metamathematical specifications. For each particular application of commonsense reasoning, the use of circumscription requires to describe the objects of reasoning by an axiom set A as well as to select a *circumscription policy*, that specifies which predicates are circumscribed (i.e. minimized), which predicates and functions are allowed to vary, and what priorities between the minimized predicates are established.

Lifschitz showed that the circumscriptive policy can be described by the policy axioms. According to [Lifschitz, 1987a], $\mathcal{P}$ and $\mathcal{F}$ stand for sets of predicates symbols and function symbols in the language of A respectively. Then a policy axiom

is defined as an additional propositional constants $V[p:c]$ for each predicate constant $p \in \mathcal{P}$ and each constant $c \in \mathcal{P} \cup \mathcal{F}$. $V[p:c]$ means that c is allowed to vary in the process of minimizing p. In his paper, the minimality condition for p in P is given pointwisely and the axiom $V[p:p]$ means that p is a minimized predicate. However, it is possible to establish relative priorities between the tasks of minimizing such "conflicting" predicates. In the formalism, assigning a higher priority to $p_1 \in \mathcal{P}$ than to $p_2 \in \mathcal{P}$ is expressed by the policy axiom $V[p_1 : p_2]$. Introducing such policy axioms expressing priorities make the theory stronger monotonically.

In [Gelfond and Lifschitz, 1988a], Gelfond and Lifschitz express a circumscriptive theory by (A, C), where A is a sentence and C is a circumscription policy which gives the *minimality conditions* w.r.t. models of A and specifies the syntactic forms of circumscription relative to A. For example, the syntactic difference between definitions of parallel circumscription and prioritized circumscription occurs due to the difference between their circumscription policies.

2.2.2 Default Logic

Default Logic was proposed by Reiter [1980] which extends a first order logic by introducing new inference rules called $default$. This is one of the most intuitive and natural logics for nonmonotonic reasoning.

Definition 2.12 [Reiter, 1980] A $default$ is an inference rule of the form as follows:

$$\frac{\alpha(\mathbf{x}) : \beta_1(\mathbf{x}), \ldots, \beta_m(\mathbf{x})}{w(\mathbf{x})},$$

where $\mathbf{x}$ be a tuple of free variables and $\alpha(\mathbf{x}), \beta_1(\mathbf{x}), \ldots, \beta_m(\mathbf{x}), w(\mathbf{x})$ be first-order formulas each of which contains $\mathbf{x}$ as free variables. The intended meaning of the default is: for a specific $\mathbf{x}$, "if $\alpha(\mathbf{x})$ can be shown and $\neg\beta_1(\mathbf{x}), \ldots, \neg\beta_m(\mathbf{x})$ cannot be shown, then derive $w(\mathbf{x})$". $\alpha(\mathbf{x})$ is called the *prerequisite*, $\beta_1(\mathbf{x}), \ldots, \beta_m(\mathbf{x})$ the $justifications$, and $w(\mathbf{x})$ the *consequence* of the default. A $default$ without free variables is called a *closed default*. A default is called a *normal default* if it contains only one justification that is equivalent to the consequent as follows:

$$\frac{\alpha(\mathbf{x}) : w(\mathbf{x})}{w(\mathbf{x})}.$$

Definition 2.13 [Reiter, 1980] A *default theory* is a pair (D, W) where D is a set of $defaults$ and W is a set of first-order formulas. If D is a set of closed defaults, then (D, W) is called a *closed default theory*. If D is a set of normal defaults, then (D, W) is called a *normal default theory*.

For a closed default theory (D, W), a fixed pont operator Γ is defined and extensions of the default theory are defined as its fixed points as follows:

Definition 2.14 [Reiter 1980, Definition 1] Let (D, W) be a closed default theory. For any set E of sentences (closed formulas), $\Gamma(E)$ is the smallest set of sentences such that:

1. $W \subseteq \Gamma(E)$,
2. $Th(\Gamma(E)) = \Gamma(E)$, where $Th(X)$ is the deductive closure of X, i.e. a set of all logical consequences of X.
3. For any default in D and any tuple $\mathbf{t}$ of ground terms, if $\alpha(\mathbf{t}) \in \Gamma(E)$ and $\neg\beta_1(\mathbf{t}), \ldots, \neg\beta_m(\mathbf{t}) \notin E$, then $w(\mathbf{t}) \in \Gamma(E)$.

Then a fixed point E of the operator Γ such that $\Gamma(E) = E$ is called an *extension* of (D, W).

Since in the above definition, an extension is defined as a fixed point of the operator Γ, it cannot be obtained constructively, and thus we must solve the fixed point equation to know an extension.

On the other hand, a semantics of default logic was finally found out by Etherington [Etherington, 1987b] in seven years later of the publication of Reiter's paper in the Artificial Intelligence Journal. Though its details are omitted in this thesis, we show a procedure [Satoh, 1992] of producing an extension based on a semantics of default logic [Etherington, 1987b, 1988] in the following. For this purpose, an alternative definition of extensions [Satoh, 1992] is presented as follows.

Definition 2.15 [Satoh, 1992] Let (D, W) be a closed default theory. We define $CON(D)$ as a set of consequences of each default in D. And let E and E' be a set of formulas. We define $UD(D, E', E)$ as a set of usable defaults in D with respect to E' and E as follows:

$$\left\{ \frac{\alpha : \beta_1, \ldots, \beta_m}{w} \in D | \alpha \in E' \; and \; for \; every \; \beta_i, \; \neg\beta_i \notin E \right\}.$$

Definition 2.16 [Satoh, 1992] An extension of (D, W) is E if E is equivalent to the smallest set of formulas E' such that $E' = Th(CON(UD(D, E', E)) \cup W)$ where $Th(X)$ is a set of tautological consequences of X.

<<**A non-deterministic procedure of computing extensions**>>
[Etherington, 1988; Satoh, 1992]

Step 0: $E_0 = Th(W), \quad i := 0.$

Step 1: **select** $\delta_i \in UD(D, E_i, E_i)$ such that the consequence of the default δ_i is not in E_i. If there is not such δ_i, **output** E_i else **go to** Step 2.

Step 2: $E_{i+1} = Th(E_i \cup \{\omega_i\})$ where ω_i is the consequence of the default δ_i. If there is some $\delta_n (n = 0, \ldots, i)$ such that $\delta_n \notin UD(D, E_{i+1}, E_{i+1})$ then **fail** else $i = i + 1$ and *go to* Step 1.

There are default theories with multiple or, even, no extensions. However, there is an interesting property w.r.t. a normal default theory that it always has at least one extension [Reiter, 1980, Theorem 3.1].

2.2.3 Abduction in Logic

Abduction is one of the three fundamental reasoning classified by the philosopher Peirce [1932] while the others are *deduction* and *induction*. In terms of logic, deduction derives a result C from a general rule $A \supset C$ and a case A based on an inference rule called *modus ponens* where A and C are first-order formulas. On the other hand, abduction derives a case A from a general rule $A \supset C$ and a result C, where A is probationally adopted hypothesis as explanation for the observed fact C according to known rule $A \supset C$. This means that, although we cannot conclude that A is true, we can say that A is a sufficient *hypothesis* that *explains* the observation C based on the general rule $A \supset C$.

As a logical framework for abductive reasoning, *Theorist* was recently proposed by Poole, et al.[1987], which is a simple hypothetical reasoning system based on a first-order theorem prover that distinguishes hypotheses from facts.

Definition 2.17 (Theorist)[Poole, 1988] *

A first order language is given. Let Σ be a set of closed formulae in the language, which is treated as facts, and Γ be a set of formulae, called hypotheses. Given a closed formula G, a set Δ of ground instances of elements of Γ is an *explanation* of G with respect to (Σ, Γ) if

1. $\Sigma \cup \Delta \models G$, and

2. $\Sigma \cup \Delta$ is consistent.

An explanation Δ of G is *minimal* if no proper subset Δ' of Δ is an explanation of G.

Hereafter, we call (Σ, Γ) given by Definition2.17 the $Theorist\ framework$ according to [Inoue, 1992b].

2.2.4 Relating Circumscription, Default Logic and Abduction

Etherington [1987a, 1988] has shown that there exists an important relationship between circumscription and default logic under the conditions of the *domain-closure assumption* as well as the *unique-name assumption*.

* This definition owes to [Inoue, 1992b]. Poole originally gave a slightly different version of the definition in [Poole, 1988].

The domain closure assumption assumes that the only objects in the domain (universe) are the ones that can be named using the object and function constants in the language. Thus this can be written by using the *domain-closure axiom*(DCA)[Reiter, 1984] as follows: $\forall x(x = t_1 \vee x = t_2 \vee \ldots)$,
where the t_i's $(i = 1, 2, \ldots)$ are all ground terms. Note that if there exists a function symbol, then the DCA cannot be expressed any more by a first order formula since there are infinite terms.

On the other hand, the unique-names assumption assumes that any object in the domain (universe) cannot be named (i.e. represented) by more or equal two names (i.e. ground terms). Thus this is expressed by the conjunction of all the *uniqueness* of *names axioms*(UNA) as follows:

$$\bigwedge_{i,j} t_i \neq t_j \qquad (where\ i, j = 1, 2, \ldots, i \neq j) \quad .$$

In this thesis, all ground terms occuring in DCA and UNA are supposed to be finite. Supposing DCA and UNA, Etherington showed the relationship between a normal default theory and circumscription without fixed predicates as follows:

Theorem 2.18 [Etherington, 1987a][†] *Assume that T is a theory including DCA and UNA. Let P be a tuple of predicates, and Z the tuple of all predicates disjoint with P in the language. Then, those formulas true in every default extension of the default theory:*

$$\left(\left\{ \frac{: \neg p(x)}{\neg p(x)} \middle| \, p \in P \right\}, T \right)$$

are precisely those theorems of $Circum(T; P; Z)$.

Theorem 2.18 is very important for us, because our main theorems in chapter 4 are based on this theorem.

On the other hand, Poole [1988] shows an interesting relationship between a normal default theory and the *Theorist* framework for abductive reasoning by defining *Theorist extension* as follows:

Definition 2.19 Suppose that the *Theorist framework* (Σ, Γ) is given where Σ is a set of facts, and Γ a set of hypotheses. *Theorist extension* of (Σ, Γ) is the set of logical consequences of $\Sigma \cup \mathbf{M}$ where $\mathbf{M}$ is a maximal (with respect to set inclusion) set of ground instances of elements of Γ such that $\Sigma \cup \mathbf{M}$ is consistent.

Theorem 2.20 [Poole, 1988, Theorem 4.1] *For the given Theorist framework (Σ, Γ), we define a default theory (D_Γ, Σ), where*

$$D_\Gamma = \left\{ \frac{: w(x)}{w(x)} \middle| \, w(x) \in \Gamma \right\}.$$

Then it holds that a set S of formulas is a Theorist extension of (Σ, Γ) if and only if S is an extension of a normal default theory (D_Γ, Σ).

[†] The description of this theorem owes to [Inoue, 1992b]. Etherington originally showed this theorem by using slightly different notation in [Etherington, 1987a].

Poole also shows the following theorem.

Theorem 2.21 [Poole, 1988, Theorem 2.6] *Let G be a formula. There is an explanation of G with respect to the Theorist framework (Σ, Γ) if and only if there is a Theorist extension of (Σ, Γ) in which G holds.*

As Inoue[Inoue, 1992b] points out, the next theorem is obtained by combining the above two theorems.

Theorem 2.22 [Inoue, 1992b] *There is an explanation of G with respect to the Theorist framework (Σ, Γ) if and only if there is an extension of the normal default theory (D_Γ, Σ) in which G holds.*

Thus from these theorems, we regard abduction as one of nonmonotonic reasoning. Besides, though there are interesting works such as [Ginsberg, 1989], [Inoue and Heft, 1990] on evaluating circumscription using abduction, we do not review about them in detail here since our methods presented in the subsequent chapters of this thesis do not concern about such issues.

2.3 Logic Programming

2.3.1 Classes of Logic programs

We show classes of logic programs as follows [Inoue and Sakama, 1994].

A *general extended disjunctive program* (GEDP) is a set of rules of the form:

$$L_1 \mid \ldots \mid L_k \mid not L_{k+1} \mid \ldots \mid not L_\ell \leftarrow L_{\ell+1}, \ldots, L_m, \ not L_{m+1}, \ldots, \ not L_n$$

where L_i's are literals and $n \geq m \geq \ell \geq k \geq 0$. *not* is a *negation as failure* operator. The disjunction in the left of $\leftarrow$ is called the *head* and the conjunction in the right of $\leftarrow$ is called the *body* of the rule. A rule with the empty head is an *integrity constant*. A rule with variables stands for the set of its ground instances.

A GEDP is called an extended disjunctive program (EDP) when it contains no positive *not*, i.e., $k = \ell$ in each rule. A GEDP is called a general disjunctive program (GDP) if every L_i is an atom, i.e., positive literals without classical negation $\neg$. An EDP is called (i) an *extended logic program* (ELP) if for each rule, $k = \ell = 1$; and (ii) a *normal disjunctive program* (NDP) if every L_i is an atom. A NDP is called (i) a *normal logic program* (NLP) if for each rule, $k = \ell = 1$; and (ii) a *positive disjunctive program* (PDP) if it contains no *not*, i.e., for each rule $m = n$. A NLP is called a *stratified logic program* if its form syntactically satisfies the condition of a *stratification* (see Section 2.3.4).

2.3.2 Extended Logic Programs and Default Logic

Extended logic programs (ELP) have the expressive power to represent classical negation ($\neg$) along with negation as failure (*not*) which enables us to represent

incomplete information [Gelfond and Lifschitz, 1991]. The following definitions give syntax and semantics of an extended logic program with integrity constraints, which are slightly extended by including integrity constraints by Inoue [1994] from the original definitions of Gelfond and Lifschitz [1991].

Definition 2.23 [Gelfond and Lifschitz, 1991; Inoue, 1994]
A extended logic program is a set of rules of the form:

$$L \leftarrow L_1, \ldots, L_m, notL_{m+1}, \ldots, notL_n, \tag{2.1}$$

or of the form:

$$\leftarrow L_1, \ldots, L_m, notL_{m+1}, \ldots, notL_n, \tag{2.2}$$

where $n \geq m \geq 0$, and L and L_i's are literals. Each rule of the form (2.2) is called an *integrity constraint*.

The declarative meaning of an extended logic program is given by the *answer sets semantics* as follows [Gelfond and Lifschitz, 1991].

Definition 2.24 [Gelfond and Lifschitz, 1991; Inoue, 1994]
The answer sets of an extended logic program are defined in the following two steps.

1. Let Π be a set of ground rules of an extended logic program not containing *not* and *Lit* be the set of ground literals in the language. The answer set, $\alpha(\Pi)$, of Π is the smallest subset S of *Lit* satisfying the conditions:
   (a) For any rule $L \leftarrow L_1, \ldots, L_m$ in Π, if $L_1, \ldots, L_m \in S$, then $L \in S$;
   (b) For any integrity constraint $\leftarrow L_1, \ldots, L_m$ in Π, if $L_1, \ldots, L_m \in S$, then $S = Lit$;
   (c) if S contains a pair of complementary literals, then $S = Lit$.
2. Let Π be any extended logic program without variables. For any set $S \subseteq Lit$, let Π^S be the set of rules without *not* obtained from Π by deleting
   (a) every rule containing a formula $notL$ in its body with $L \in S$, and
   (b) every formula $notL$ in the bodies of the remaining rules.

   Then, S is an answer set of Π if S is the answer set of Π^S, that is, $S = \alpha(\Pi^S)$.

The notion of *consistency* is important for extended logic programs. Extended logic programs are classified as follows.

Definition 2.25 [Inoue, 1992b] Let Π be an extended logic program.
(1) Π is *consistent* if it has an answer set which is not *Lit*.
(2) Π is *contradictory* if it has an inconsistent answer set which is *Lit*.
(3) Π is *incoherent* if its has no answer set.

Notice that no extended logic program is both consistent and contradictory, and a contradictory program has the unique answer set *Lit*.

Gelfond and Lifschits [1991] show the relation between the answer sets of an extended logic program and the extensions of the corresponding Reiter's default theory[1980]. Every rule in an extended logic program Π of the form

$$L \leftarrow L_1, \ldots, L_m, notL_{m+1}, \ldots, notL_n,$$

can be identified with the default

$$\frac{L_1 \wedge \ldots \wedge L_m : \overline{L_{m+1}}, \ldots, \overline{L_n}}{L}$$

where $\overline{L}$ is the literal complementary to L: that is, if A is an atom, $\overline{A} = \neg A$, $\overline{\neg A} = A$.

Thus every extended logic program Π is identified in this way with the corresponding default theory (Π,ϕ). Gelfond and Lifschitz show that there is a 1-1 correspondence between the answer sets of Π and the extensions of a default theory (Π,ϕ) as follows.

Theorem 2.26 [Gelfond and Lifschitz, 1991, Proposition 3]
For any extended logic program Π,

(i) if S is an answer set of Π, then its deductive closure $Th(S)$ is an extension of a default theory (Π,ϕ).

(ii) Every extension E of a default theory (Π,ϕ) is the deductive closure of exactly one answer set $E \cap Lit$ of Π, i.e. $E = Th(E \cap Lit)$,

where $Th(S)$ denotes the deductive closure, i.e., a set of logical consequences of S.

2.3.3 Normal Logic Programs

An extended logic program is reduced to a normal logic program when L and L_i's in (2.1) and (2.2) are atoms, and its declarative meaning, i.e. answer set semantics, is reduced to stable model semantics [Gelfond and Lifschitz, 1988b]. In the following, we show the definition of a stable model for a normal logic program with integrity constraints by slightly modifying the primary definition given by Gelfond and Lifschitz [1988b].

Definition 2.27 A normal logic program T with integrity constraints is a set of rules of the form:

$$A \leftarrow A_1, \ldots, A_m, notA_{m+1}, \ldots, notA_n, \tag{2.3}$$

or of the form:

$$\bot \leftarrow A_1, \ldots, A_m, notA_{m+1}, \ldots, notA_n, \tag{2.4}$$

where $n \geq m \geq 0$, and A and A_i's are atoms. Each rule of the form (2.4) is called an *integrity constraint*. The following form (2.5) which has the empty head is often used instead of (2.4) as an integrity constraint.

$$\leftarrow A_1, \ldots, A_m, notA_{m+1}, \ldots, notA_n, \tag{2.5}$$

For a given normal logic program T with integrity constraints, a stable model which satisfies all integrity constraints is defined as follows.

Definition 2.28 (Stable model semantics) [Gelfond and Lifschitz, 1988b]
Let T be a normal logic program and Π_T be a set of ground rules obtained by replacing all variables in each rule in T by every element of its Herbrand universe. Let M be a set of ground atoms from Π_T and Π_T^M be the following (possibly infinite) program.

$$\Pi_T^M = \{A \leftarrow A_1, \ldots, A_m | A \leftarrow A_1, \ldots, A_m, notA_{m+1}, \ldots, notA_n \in \Pi_T, \\ \text{and } A_i \notin M \text{ for each } i = m+1, \ldots, n.\}$$

Let $min(\Pi_T^M)$ be the least model of Π_T^M. A *stable model* for a normal logic program with integrity constraints T is M iff $M = min(\Pi_T^M)$ and $\bot \notin M$. We say that T is *consistent* if there exists a stable model for T.

Theorem 2.29 *If T is a consistent normal logic program with integrity constraints, then the answer set of T are identical to the stable models of T. On the other hand, when T has no stable model, T may have the answer set Lit if no other answer set satisfies the integrity constraints in T.*

2.3.4 Stratified Logic Programs and Circumscription

A class of stratified logic programs was proposed by [Apt,et. 1987] and [Van Gelder, 1988]. This is a subclass of normal logic programs which syntactically satisfies the condition of *stratification* and its declarative meaning is given by the iterated fixed point semantics[Apt,et. 1987] or equivalently perfect model semantics [Przymusinski, 1987].

Definition 2.30 (Stratified Logic Programs)‡ [Przymusinski, 1987]
A normal logic program Π (without integrity constraints) is called a *stratified logic program* if it is possible to decompose the predicates of the program Π into k disjoint parts

$$P_1; \ldots; P_k \tag{2.6}$$

called *strata*, so that for every rule of the form where A_i, B_j, C are atoms:

$$C \leftarrow A_1, \ldots, A_m, notB_{m+1}, \ldots, notB_n,$$

if the predicate of C belongs to some stratum, say P^ℓ $(1 \leq \ell \leq k)$, we have that:

(a) all predicates of A_i belong to $P_1, \ldots, P_\ell$;

(b) all predicates of B_j belong to $P_1, \ldots, P_{\ell-1}$.

‡ The syntax and semantics of a stratified logic program with integrity constraints is presented in Section 7.5

Any particular decomposition $P^1; \ldots; P^k$ of the predicates of Π satisfying the above conditions is called a *stratification* of Π. For $i < j$, we call that a stratum P^i is lower than a stratum P^j.

A stratified logic program has a unique stable model which is equivalent to an iterated fixed point model defined as follows.

Definition 2.31 (An iterated fixed point model [Apt,et. 1987])
Let Π be a stratified logic program which has a *stratification* of Π such that $P^1 ; \ldots ; P^k$. Let Π_i stand for the subset of Π which consists of all definitions of predicates in P^i where the definition of a predicate symbol P denotes the subset of Π consisting of all rules that contain P in the head. Then corresponding to $P^1 ; \ldots ; P^k$, i.e. a *stratification* of Π, Π is decomposed into k disjoint sets of rules as follows.

$$\Pi_1; \ldots; \Pi_k$$

For every $i = 1, \ldots, k$, let T_i be the operator on sets of ground atoms defined as follows: for every set M of ground atoms and every ground atom H, $H \in T_i(M)$ iff $H \in M$ or, for some rule $H_1 \leftarrow L_1, \ldots, L_n$ in Π_i (where L_f $(1 \leq f \leq n)$ is an atom or a negated atom of the form $notB$) and some substitution θ of ground terms for variables, $L_1\theta, \ldots, L_n\theta$ are true in M and $H = H_1\theta$.

The sets $M_0, \ldots, M_k$ of ground atoms are defined by the equations:

$$M_0 = \phi,$$
$$M_i = \bigcup_{j \geq 0} T_i^j(M_{i-1}) \qquad (i = 1, \ldots, k).$$

(T^j is T applied j times.) Each M_i becomes a fixed point of each operator T_i $(1 \leq i \leq k)$. Apt, Blair, and Walker show [1987] that M_k is a canonical model of Π called the *iterated fixed point model* and it does not depend on the choice of stratification (2.6). M_k is shown to be a unique *perfect model* if Π is a stratified logic program [Przymusinski, 1987].

Theorem 2.32 [Gelfond and Lifschitz, 1988b, Corollary 2]
If Π is a stratified logic program, then its unique stable model is identical to its iterated fixed point model.

Przymusinski found a semantic relationship between a stratified logic program and prioritized circumscription without fixed predicates as well as variable predicates as follows.

Theorem 2.33 [Przymusinski, 1988]
Let Π be a stratified logic program without functions, and $P^1; \ldots; P^k$ be a stratification of Π. Suppose Π' stands for $Circum(\widetilde{\forall}\Pi; P^1 > \ldots > P^k)$ where $\widetilde{\forall}$ denotes

universal closure. Then, according to Przymusinski [Przymusinski, 1988]*, for a sentence F in the language of* Π*,*

$$M_\Pi \models F \qquad \textit{iff} \qquad U \wedge \Pi' \models F$$

where M_Π is a canonical model of Π *given by the iterated fixed point semantics* [Apt,et. 1987]*(or equivalently a perfect model*[Przymusinski, 1987]*), and U stands for the Uniqueness of names axioms.*

This means that $M_\Pi \models F$ can be computed from the answer of a sound and complete procedure w.r.t the iterated fixed point semantics (or equivalently the perfect model semantics) of a stratified logic program Π, then so can $U \wedge \Pi' \models F$.

2.3.5 Abductive Logic Programming

Recently, abductive reasoning is related with logic programming, and abductive frameworks defined by means of logic programs have been proposed. Eshghi and Kowalski [1989] are the first to consider abduction based on stable model semantics for the class of normal logic programs. In [Eshghi and Kowalski, 1989], they give an abductive interpretation of negation as failure and provide a top-down abduction procedure which is given as a proof theory for abduction. Dung [1991] shows that Eshghi and Kowalski's procedure is sound w.r.t. preferential semantics proposed by him [Dung, 1991]. Eshghi and Kowalski's approach is followed by Kakas and Mancarella [1990] in which a *generalized stable model* is provided as the semantics for abduction. On the other hand, Satoh and Iwayama present a bottom-up method to compute abduction [Satoh and Iwayama, 1991] as well as a top-down abduction procedure [Satoh and Iwayama, 1992], both of which are sound w.r.t. stable model semantics and are based on generalized stable models.

In the following, we show the definitions of an abductive framework, generalized stable models and explanations in Kakas and Mancarella [1990], but for notational conveniences, we show the slightly modified versions of them according to Satoh and Iwayama [1991,1992].

Definition 2.34 [Kakas and Mancarella, 1990]
An abductive framework is a pair $\langle T, A\rangle$ where A is a set of predicate symbols, called *abducible predicates* and T is a set of rules of a normal logic program, each of whose head is not in A. A set of all ground atoms for predicates in A is called *abducibles*.

Definition 2.35 [Kakas and Mancarella, 1990]
Let $\langle T, A\rangle$ be an abductive framework and Δ be a subset of abducibles generated from A. A generalized stable model $M(\Delta)$ is a stable model of $T \cup \{H \leftarrow | H \in \Delta\}$. We say that $\langle T, A\rangle$ is *consistent* if there exists a generalized stable model $M(\Delta)$ for some Δ.

Definition 2.36 [Kakas and Mancarella, 1990]
Let $\langle T, A\rangle$ be an abductive framework, Q be an atom called *observation* and Δ be a subset of abducibles generated from A. Q has an *explanation* Δ w.r.t $\langle T, A\rangle$ if and only if there exists a generalized stable model $M(\Delta)$ such that $Q \in M(\Delta)$.

Notice that this definition of an explanation corresponds to Theorist's explanation in Definition 2.17 though the former is based on a stable model for a normal logic program and the latter is based on a first-order predicate logic.

In [Satoh and Iwayama, 1991], Satoh and Iwayama present a method to compute explanations of an observation w.r.t. an abductive framework only in propositional programs. Here we show their theorem in slightly extended form for non-propositional programs as follows.

Theorem 2.37 [Satoh and Iwayama, 1991]
Let $\langle T, A\rangle$ be an abductive framework, Q be an observation and Δ be a subset of abducibles generated from A. We define $\Gamma(\Delta)$ as a set of rules as follows:

$$\Gamma(\Delta) \stackrel{def}{=} \{p'(t) \leftarrow not\ p(t),\ p(t) \leftarrow not\ p'(t) |\ p(t) \in \Delta\ \}$$

where each p' is a newly introduced predicate symbol corresponding to each abducible predicate p in A. Then Q has an explanation Δ w.r.t. $\langle T, A\rangle$ if and only if there is a stable model M for $T \cup \Gamma(A) \cup \{\leftarrow not\ Q\}$ such that $\Delta = M \cap A$.

Chapter 3

Compiling Circumscription into Stratified Programs

3.1 Introduction

Circumscription was proposed by J. McCarthy [1980] for formalizing non-monotonic aspects of common sense knowledge and reasoning. Soon after, Lifschitz [1985] and McCarthy [1986] introduced a new form of circumscription called *prioritized circumscription* to solve the multiple extension problem in inheritance system, where different kinds of abnormality should be taken into account and it would lead to conflict if they were minimized (i.e. circumscribed) in parallel.

There are many examples which reveal the expressive power of prioritized circumscription. For example, *soft constraints*[Satoh, 1990] in the area of scheduling, design and planning, which provide preferences over solutions, can be represented syntactically by prioritized circumscription and the most preferable solution can be obtained as a most preferable model since the models are ordered by the given priorities between predicates corresponding with preference over solutions.

With respect to computation of prioritized circumscription, there are two approaches; one is a compiling method such as Gelfond and Lifschitz's method [Gelfond and Lifschitz, 1988a], and the other is an interpretive one such as Baker and Ginsberg's theorem prover for prioritized circumscription [Baker and Ginsberg, 1989] as well as Przymusinski's algorithm based on MILO-resolution[Przymusinski, 1989].

Gelfond and Lifschitz explored the approach to use of logic programming for computing prioritized circumscription based on the relationship between the semantics of prioritized circumscription and the semantics of the stratified logic program [Lifschitz, 1987a; Przymusinski, 1987]. Their method consists of two processes: compilation of a given axiom sets into a logic program by using two operations, and query evaluation from the response of logic programming interpreter.

At the beginning of our research, we tried to apply Gelfond and Lifschitz's compiling method to some practical problems since their method is as efficient as the logic program interpretation, but we found such difficulties that their method does not guarantee that a given axiom set can be always compiled successfully

into a stratified logic program whose stratification syntactically satisfies the circumscriptive policy given by a user. For example:

1. There is a case that a clause with more than two positive literals is generated in the process of compilation, and it cannot be decided which positive literal should be placed in the head of the rule even making use of the priorities of the predicates given by the circumscriptive policy. This case occurs, for example, if positive literals are in the same stratum.
2. Even if all clauses are transformed into the form of rules, but if their syntax does not correspond to the stratification specified by the circumscriptive policy, their method also cannot be applied to this case.

Then, the aim of the method presented in this chapter is to extend their method in order to overcome such difficulties. We propose to transform a given circumscription into a logically equivalent one in which such difficulties disappear. We show it can be done by making use of the Lifschitz's result [Lifschitz, 1985] that some class of parallel circumscription whose axiom set is the form of a *solitary formula* can be replaced by an equivalent first order formula.

As a result, some class of circumscriptive theories, which cannot be handled by Gelfond and Lifschitz's method, can also be compiled into logic programs and query evaluation can be done in a similar way.

This chapter is organized as follows. In Section 3.2, we review Gelfond and Lifschitz's method and explain the limitation of their method by using an example. In Section 3.3, we present theorems and corollaries which give the theoretical foundations of our method and define the additional three operations. In Section 3.4, we examine examples which cannot be handled by their method and explain how circumscriptive theories of these examples can be translated into logic programs by using our theorems. In Section 3.5, we discuss about related works. The proofs of two main theorems (Theorem 3.7 and Theorem 3.12) are deferred to Section 3.6.

3.2 Overview of Gelfond and Lifschitz's method

Making use of Theorem 2.33, Gelfond and Lifschitz [1988a] tried to evaluate a query W with respect to the following circumscription:

$$Circum(\widetilde{\forall} A \wedge U; P^1 > \ldots > P^k; Z),$$

by means of

- compiling a set of clauses A into a stratified program Π;
- and evaluating a query W from the response of the logic programming interpreter of Π.

Here it is supposed that W is a ground atom whose predicate symbol occurs in A, no function symbols are included in A and U is the conjunction of the uniqueness of names axioms. $\widetilde{\forall}$ denotes universal closure. The circumscriptive policy is given by such priorities as $P^1 > \ldots > P^k$ where $P^1, \ldots, P^k$ are disjoint lists of the minimized predicates occurring in A. Z is the list of all varied predicates $Z_1, \ldots, Z_\ell$ and it is also assumed that every clause in A contains at most one literal whose predicate symbol belongs to Z.

In their method, compilation consists of two operations, which transform A into a set of clauses:$Replace(A) \cup Resolve(A)$. By the operation of $Replace$, $Replace(A)$ is obtained in such a way that each $\neg Z_i (1 \leq i \leq \ell)$ in A is replaced by a new predicate symbol $\overline{Z_i}$. Then $\overline{Z}$ denotes the lists of new predicates $\overline{Z_1}, \ldots, \overline{Z_\ell}$. By the operation of $Resolve$, $Resolve(A)$ is obtained by resolving each pair of clauses from A upon an atom whose predicate symbol belongs to Z.

After compilation, if $Replace(A) \cup Resolve(A)$ satisfies the syntactical condition of a stratified program whose stratification should be

$$P^1; \ldots; P^k; Z, \overline{Z}$$

then A is successfully compiled into a stratified logic program Π, logically equivalent to $Replace(A) \cup Resolve(A)$, and the query evaluation w.r.t above circumscription can be computed from the response of the logic programming interpreter of Π.

But the following example shows a typical limitation of their method.

Example 3.1

Let's consider the following common sense reasoning:

$$U: \quad Tweety \neq Joe, \tag{1}$$

$$A: \quad x = Tweety \vee x = Joe, \tag{2}$$

$$bird(Tweety) \vee bird(Joe), \tag{3}$$

$$\neg fly(Joe), \tag{4}$$

$$bird(x) \wedge \neg ab(x) \supset fly(x). \tag{5}$$

Suppose that $Circum(\widetilde{\forall} A \wedge U; ab > bird; fly)$ is given in this case.

Now $Resolve(A) \cup Replace(A)$ are obtained as the following set of clauses π. Hereafter we use T and J instead of $Tweety$ and Joe.

$$\pi: \quad bird(T) \vee bird(J), \tag{3}$$

$$\overline{fly}(J), \tag{4'}$$

$$bird(x) \wedge \neg ab(x) \supset fly(x), \tag{5}$$

$$bird(J) \supset ab(J). \tag{6}$$

It should be noted that π is not a stratified logic program because

- the axiom (3) cannot be written as a rule of a logic program since it is undecidable which literal of $bird(T)$ and $bird(J)$ should be placed in the head;

- the axiom (6) has the form like a rule, but the priority of predicate *bird* is higher or equal to the priority of the predicate *ab* and this violates the requirement of stratification, i.e. $ab > bird$, which is given by the circumscriptive policy.

3.3 Extensions of Gelfond and Lifschitz's Method

We propose to transform a given prioritized circumscription into a logically equivalent one in which such difficulties mentioned in the previous section disappear. For such transformations, we make use of formulas of equivalence, which are obtained from the Lifschitz's result shown by Definition 2.8. Theorems and corollaries in this section give theoretical foundations for the transformations. Corresponding to the corollaries, we define three operations. The proofs of Theorem 3.7 and Theorem 3.12 are contained in section 3.6.

Lemma 3.2 *For any formula B, if $Circum(A; P^1 > \ldots > P^k; Z) \models B$, then*
$$Circum(A; P^1 > \ldots > P^k; Z) \equiv Circum(A \wedge B; P^1 > \ldots > P^k; Z).$$

Theorem 3.3
$$Circum(A; P^1 > \ldots > P^k; Z) \equiv Circum(A \wedge Circum(A; P^i); P^1 > \ldots > P^k; Z).$$

Proof:

For any tuple of predicates X, it follows that
$$Circum(A; P^i; X) \vdash Circum(A; P^i), \qquad (1 \leq i \leq k).$$
According to Lifschitz[Lifschitz, 1985]*,*
$$\begin{aligned} &Circum(A; P^1 > \ldots > P^k; Z) \\ &\equiv Circum(A; P^1; P^2, \ldots, P^k, Z) \wedge \\ &\quad Circum(A; P^2; P^3, \ldots, P^k, Z) \wedge \ldots \wedge Circum(A; P^k; Z), \end{aligned}$$
then $Circum(A; P^1 > \ldots > P^k; Z) \vdash Circum(A; P^i), \quad (1 \leq i \leq k).$

Using Lemma 3.2, for any i $(1 \leq i \leq k)$,
$$Circum(A; P^1 > \ldots > P^k; Z) \equiv Circum(A \wedge Circum(A; P^i); P^1 > \ldots > P^k; Z).$$
□

Remark With respect to $Circum(A; P^i)$, it can easily be computed by the syntactical transformation of the formula A if A is a solitary formula (see Definition 2.8).

Corollary 3.4 *If A is a solitary formula such as $N(P^i) \wedge (Q \leq P^i)$, then $Circum(A; P^i)$ is written as $N(Q) \wedge (Q = P^i)$. So let $Add(A)$ be a sub-formula of $Q = P^i$ whose form is $\bigwedge_j \forall x(Q_j(x) \equiv P_j(x))$ where Q_j and P_j are predicate constants. Then it follows that*
$$Circum(A; P^1 > \ldots > P^k; Z) \equiv Circum(A \wedge Add(A); P^1 > \ldots > P^k; Z).$$

Definition 3.5 *(Operation of Add)*
Let $Add(A)$ be a formula defined in Corollary 3.4. We define Add as such an operation that transforms A into $A \wedge Add(A)$, so that we can transform the left-hand circumscription into the right-hand one w.r.t the equivalence relation shown in Corollary 3.4. .

Corollary 3.6 *Let A be a set of clauses, then*
$Circum(\widetilde{\forall}A; P^1 > \ldots > P^k; Z) \equiv Circum(\widetilde{\forall}(A \wedge Resolve(A)); P^1 > \ldots > P^k; Z)$.

Theorem 3.7 *Let P_i be a predicate constant for any i $(1 \leq i \leq k)$. Then for $1 \leq l < m \leq k$,*

$$\begin{aligned} &Circum(\forall x(P_m(x) \equiv P_l(x)) \wedge \\ &\quad A(P_1 \ldots P_l \ldots P_m \ldots P_k, Z); P_1 > \ldots > P_l > \ldots > P_{m-1} > P_m > \ldots > P_k; Z) \\ &\equiv Circum(A(P_1 \ldots P_l \ldots P_l \ldots P_k, Z); P_1 > \ldots > P_l > \ldots > P_{m-1} > P_{m+1} > \ldots \\ &\qquad \ldots > P_k; Z) \wedge \forall x(P_m(x) \equiv P_l(x)), \end{aligned}$$

where $A(P_1 \ldots P_l \ldots P_l \ldots P_k, Z)$ is such a formula that every P_m occurring in $A(P_1 \ldots P_l \ldots P_m \ldots P_k, Z)$ is replaced by P_l.

Remarks Theorem 3.7 can be generalized into Corollary 3.8.

Corollary 3.8 *Let P_i be a predicate constant for any i $(1 \leq i \leq k)$, and P^j be a tuple of predicate constants, i.e. $\left[P_{j_1}, \ldots, P_{j_{n_j}}\right]$ for any j $(1 \leq j \leq k)$. Then for $1 \leq l < m \leq k$,*

$$\begin{aligned} &Circum(\forall x(P_{m_s}(x) \equiv P_{l_t}(x)) \wedge \\ &\quad A(P^1 \ldots P^l \ldots P^m \ldots P^k, Z); P^1 > \ldots > P^l > \ldots > P^{m-1} > P^m > \ldots > P^k;\ Z) \\ &\equiv Circum(A(P^1 \ldots P^l \ldots P^m_{rep} \ldots P^k, Z); P^1 > \ldots > P^l > \ldots > P^{m-1} > P^m_{drop} > \ldots \\ &\qquad \ldots > P^k; Z) \wedge \forall x(P_{m_s}(x) \equiv P_{l_t}(x)), \end{aligned}$$

where $P^l, P^m, P^m_{rep}, P^m_{drop}$ are especially defined by using P_{m_s} and P_{l_t} as follows:

$$\begin{aligned} P^l &\stackrel{def}{=} \left[P_{l_1}, \ldots, P_{l_{t-1}}, P_{l_t}, P_{l_{t+1}}, \ldots, P_{l_{n_l}}\right], \\ P^m &\stackrel{def}{=} \left[P_{m_1}, \ldots, P_{m_{s-1}}, P_{m_s}, P_{m_{s+1}}, \ldots, P_{m_{n_m}}\right], \\ P^m_{rep} &\stackrel{def}{=} \left[P_{m_1}, \ldots, P_{m_{s-1}}, P_{l_t}, P_{m_{s+1}}, \ldots, P_{m_{n_m}}\right], \\ P^m_{drop} &\stackrel{def}{=} \left[P_{m_1}, \ldots, P_{m_{s-1}}, P_{m_{s+1}}, \ldots, P_{m_{n_m}}\right]. \end{aligned}$$

Definition 3.9 *(Operation of Equiv)*
W.r.t the logical equivalence shown in both Theorem 3.7 and Corollary 3.8, we define *Equiv* as such an operation that transforms the left-hand circumscription into the right-hand one.

Corollary 3.10 *Suppose that* $\Pi \vdash \gamma$, *then*

$$Circum(\Pi \wedge \gamma; P^1 > \ldots > P^k) \equiv Circum(\Pi; P^1 > \ldots > P^k),$$

where $P^1, \ldots, P^k$ *are tuples of predicates occurring in* Π.

Definition 3.11 *(Operation of Delete)*

W.r.t. the logical equivalence in Corollary 3.10, we define *Delete* as such an operation that transforms the axiom set $\Pi \wedge \gamma$ into Π in order to transform the left-hand circumscription into the right-hand one.

In the following theorem, we use the same terminology and suppose the same assumptions addressed in section 3. This is the extended one of Gelfond and Lifschitz's main theorem[Gelfond and Lifschitz, 1988a].

Theorem 3.12 *Suppose the same assumption addressed in Gelfond and Lifschitz's main theorem[Gelfond and Lifschitz, 1988a]. We consider to evaluate a query* W *w.r.t the following circumscription:*

$$Circum(\widetilde{\forall} A \wedge U; P^1 > \ldots > P^k; Z).$$

Hereafter let $Circum$ *stand for this circumscription. Then, suppose that after applying operations:* $Resolve, Add, Equiv$ *sequentially,* $Circum$ *is equivalently transformed into the following formula:*

$$Circum(\widetilde{\forall} A' \wedge U; H^1 > \ldots > H^n; Z) \wedge \bigwedge_i \forall x(Q_i(x) \equiv P_i(x)),$$

where Q_i *and* P_i *are predicate symbols where the priority of* P_i *is higher than the priority of* Q_i, *and* Q_i *does not occur in* A' *as well as* $H^1, \ldots, H^n$.

Now let Π *be a stratified logic program, logically equivalent to* $Replace(A')$, *or* $Delete(Replace(A'))$ *such that the partition*

$$H^1;\ \ldots;\ H^n;\ Z, \overline{Z}$$

is a stratification of Π, *and let* $Ans(\Pi, W)$ *be an answer of a logic programming interpreter for any ground atom* W *whose predicate symbol occurs in* A. *Then for any tuple* a *of individual constants in* U,

1. *if* $W = Q_i(a)$,

$$\begin{array}{lll} Circum \models W & \textit{iff} & Ans(\Pi, P_i(a)) = \text{yes}; \\ Circum \models \neg W & \textit{iff} & Ans(\Pi, P_i(a)) = \text{no}; \end{array}$$

2. *if* $W \neq Q_i(a)$,

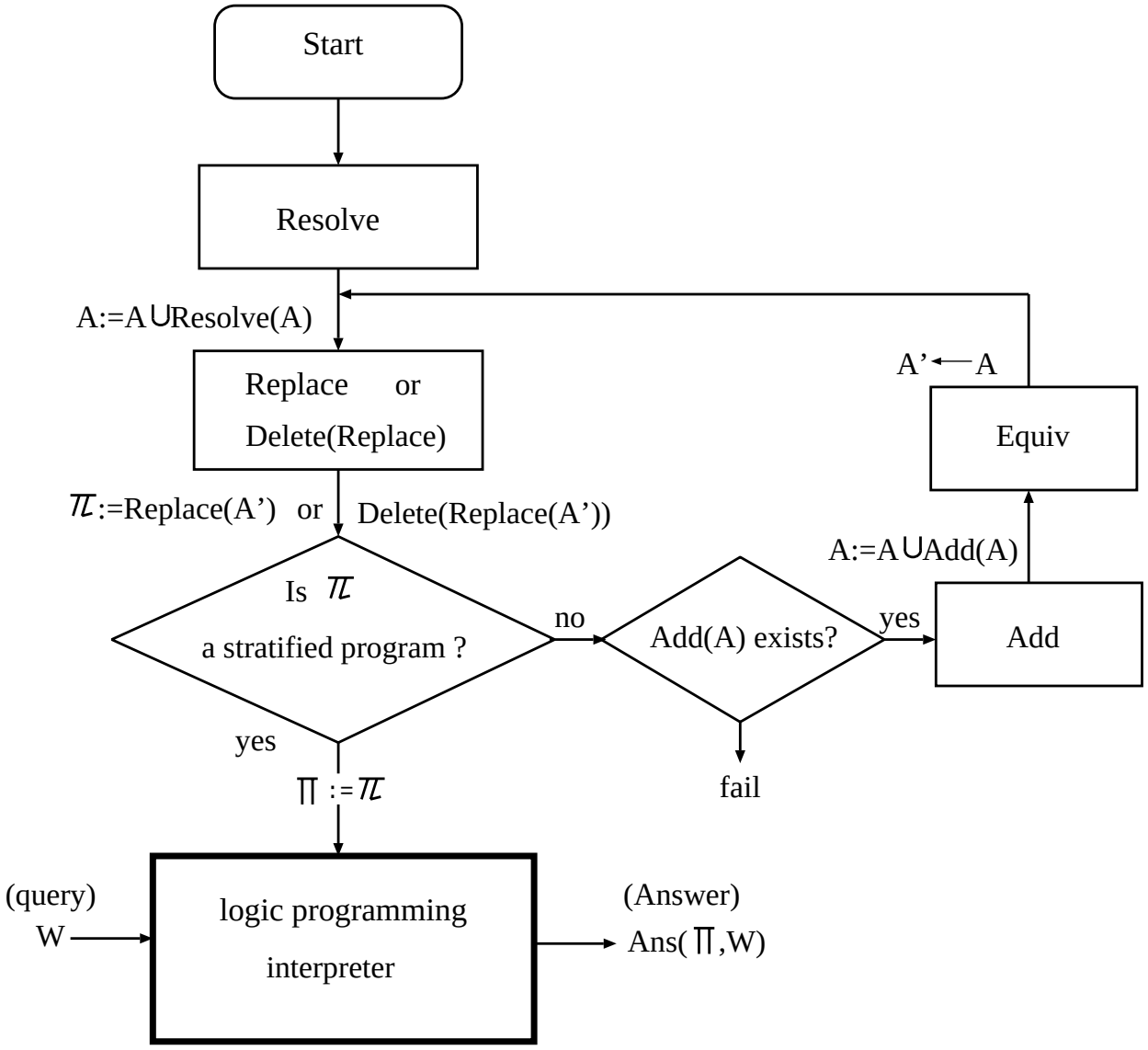

Figure 3.1 The process of compilation

(a) $Circum \models W$ *iff* $Ans(\Pi, W) =$yes;

(b) If the predicate symbol of W *belongs to* $P^1, \ldots, P^k$*, then*
$Circum \models \neg W$ *iff* $Ans(\Pi, W) =$no;

(c) If the predicate symbol of W *belongs to* Z*, then*
$Circum \models \neg W$ *iff* $Ans(\Pi, Replace(\neg W)) =$yes.

Remarks The process of compilation based on Theorem 3.12 is shown in Figure 3.1. $Add(A)$ is computed for the predicates which violate the required stratification.

3.4 Examples

Example 3.13
Here we examine Example 3.1 again. In Example 3.1,

$$Circum(\widetilde{\forall} A \wedge U; ab > bird; fly), \qquad (\mathbf{C1})$$

is given where,

U : $Tweety \neq Joe$, (1)
A : $x = Tweety \vee x = Joe$, (2)
$bird(Tweety) \vee bird(Joe)$, (3)
$\neg fly(Joe)$, (4)
$bird(x) \wedge \neg ab(x) \supset fly(x)$. (5)

1. Application of *Resolve*

   $\{bird(J) \supset ab(J)\}$ is generated as $Resolve(A)$.

2. Application of *Add*

   Since a clause in $Resolve(A)$ violates the required priority $ab > bird$ and A is solitary with respect to ab, we try to compute the parallel circumscription: $Circum(A; ab)$. Using Definition 2.8, the following is finally derived by the syntactical transformation:

   $$Circum(\widetilde{\forall} A; ab) \models bird(J) \equiv ab(J).$$

   Now, we define $Add(A)$ as $\{bird(J) \equiv ab(J)\}$.

   Then, according to Corollary 3.4 and Corollary 3.6, it holds that,

   $Circum(\widetilde{\forall}\ A \wedge U; ab > bird; fly)$

   $\equiv Circum(\widetilde{\forall}\ (A \cup Add(A) \cup Resolve(A)) \wedge U; ab > bird; fly).$

   Here, if we expand $\widetilde{\forall}\ (A \cup Add(A) \cup Resolve(A))$ by using the domain closure axiom (2), we can obtain the following equivalence relation:

$$Circum(\widetilde{\forall}\,(A \cup Add(A) \cup Resolve(A)) \wedge U; ab > bird; fly)$$
$$\equiv Circum((bird(J) \equiv ab(J)) \wedge A^1 \wedge U;$$
$$ab(T), ab(J) > bird(T), bird(J); fly(T), fly(J)). \quad (\mathbf{C2})$$

A^1 in **(C2)** is defined as a set of clauses {(2), (3), (4), (5'), (5"), (6)}, where (5') and (5") are the following clauses:

$$bird(T) \wedge \neg ab(T) \supset fly(T), \quad (5')$$
$$bird(J) \wedge \neg ab(J) \supset fly(J), \quad (5'')$$
$$bird(J) \supset ab(J). \quad (6)$$

3. Application of *Equiv*

   Let us apply Corollary 3.8 to the right hand one of **(C2)**. Then it follows that

$$Circum((bird(J) \equiv ab(J)) \wedge A^1 \wedge U;$$
$$ab(T), ab(J) > bird(T), bird(J); fly(T), fly(J))$$
$$\equiv Circum(A^2 \wedge U; ab(T), ab(J) > bird(T); fly(T), fly(J))$$
$$\wedge (bird(J) \equiv ab(J)), \quad (\mathbf{C3})$$

   where A^2 is obtained by replacing $bird(J)$ in each clause of A^1 by $ab(J)$ and therefore represented as follows:

$$A^2: \quad \forall x(x = T \vee x = J), \quad (2)$$
$$bird(T) \vee ab(J), \quad (3')$$
$$\neg fly(J), \quad (4)$$
$$bird(T) \wedge \neg ab(T) \supset fly(T). \quad (5')$$

   Now we can apply Gelfond and Lifschitz's method to the circumscription of (**C3**). Since the operation of *Resolve* has already been applied to A^2, only $Replace(A^2)$ is necessary.

4. Application of *Replace*

   As the result of *Replace*, $\neg fly(J)$ is transformed into $\overline{fly}(J)$. Futhermore, $bird(T) \vee ab(J)$ can be represented by a rule: $\neg ab(J) \supset bird(T)$ since $ab(J) > bird(T)$ is specified in (**C3**). Consequently A^2 is compiled into the following locally stratified logic program Π:

$$\Pi: \quad \neg ab(J) \supset bird(T), \quad (3'')$$
$$\overline{fly}(J), \quad (4')$$
$$bird(T) \wedge \neg ab(T) \supset fly(T). \quad (5')$$

   So a perfect model[Przymusinski, 1987] (i.e. an iterated fixed pont[Apt,et. 1987]) which is computed by a logic programming interpreter of Π, is obtained as

follows:

$$\left\{bird(T),\ fly(T),\ \overline{fly}(J)\right\}.$$

5. Query evaluation w.r.t. the prioritized circumscription

   Here, how to evaluate a query by using Theorem 3.12 is shown. $Circum$ denotes the formula **(C1)**. A query is W or $\neg W$ where W is a ground atomic formula whose predicate symbol occurs in A.

   (a) if $W = bird(J)$, then,

   $$Circum \models \neg bird(J),$$

   since $Ans(\Pi, ab(J))$=no.

   (b) if $W \neq bird(J)$, then,

   i. W such as $Ans(\Pi, W)$=yes is $bird(T)$ or $fly(T)$, so,
   $$Circum \models bird(T),$$
   $$Circum \models fly(T),$$

   ii. with respect to the circumscribed predicates, W such as $Ans(\Pi, W)$=no is $ab(J)$ or $ab(T)$, therefore,
   $$Circum \models \neg ab(J),$$
   $$Circum \models \neg ab(T),$$

   iii. with respect to the varied predicate fly, $Ans(\Pi, \overline{fly}(J))$=yes, so
   $$Circum \models \neg fly(J).$$

Example 3.14
Let us consider the following prioritized circumscription:

$$Circum(\widetilde{\forall}\ A \wedge U; ab > Japanese, diligent, poor, notpoor; have\_a\_car). \qquad \textbf{(C4)}$$

where A is given as follows:

$$A: \quad Japanese(x) \wedge \neg ab(x) \supset diligent(x), \tag{7}$$
$$ab(x) \supset poor(x), \tag{8}$$
$$diligent(x) \supset notpoor(x), \tag{9}$$
$$Japanese(x) \wedge \neg poor(x) \supset have\_a\_car(x), \tag{10}$$
$$Japanese(x) \wedge \neg notpoor(x) \supset \neg have\_a\_car(x). \tag{11}$$

Let us apply Gelfond's method to A. Then, $Replace(A) \cup Resolve(A)$ is obtained as a set of clauses $\pi \stackrel{def}{=} \{(7), (8), (9), (10), (11'), (12)\}$, where (11') and (12) are as follows:

$$Japanese(x) \wedge \neg notpoor(x) \supset \overline{have\_a\_car}(x), \tag{11'}$$
$$Japanese(x) \supset poor(x) \vee notpoor(x). \tag{12}$$

Two positive literals, $poor(x)$ and $notpoor(x)$, appear in the clause (12). Thus, the clause (12) is harmful because it cannot be eliminated as a subsumed clause as well as it cannot be represented by a rule. Therefore, π is not a stratified logic program. But it holds that,

$$\pi - \{Japanese(x) \supset poor(x) \vee notpoor(x)\} \vdash Japanese(x) \supset poor(x) \vee notpoor(x).$$

Then, we can apply the operation of *Delete* to π. So let Π be π-$\{(12)\}$ and according to Corollary 3.10, we can compile A into a stratified logic program Π instead of a non-stratified program π.

Example 3.15
We consider the meeting scheduling problem proposed by Satoh[Satoh, 1990].

"The meeting must be held on one day of day 1, 2 or 3. In general, the president, the vice president and the manager should preferably attend the meeting. The schedule of the president are prioritized to the schedule of the vice president and the schedule of the vice president are prioritized to the schedule of the manager. Suppose that the president cannot attend the meeting on day 1, the vice president cannot on day 3 and the manager cannot on day 2 respectively. Then, for what day is the meeting scheduled?"

Now, suppose $C(x)$ represents that the meeting will be held on day x.
$P(x), V(x), M(x)$ represent that the president, the vice president and the manager attend the meeting on day x respectively. E_1, E_2, E_3 are abnormal predicates about the schedules of the president, the vice president and the manager respectively. Then, this meeting scheduling problem can be represented as follows:

$$Circum(\widetilde{\forall} A \wedge U;\ E_1 > E_2 > E_3 > C;\ P, V, M), \qquad (\mathbf{C6})$$

where,

$$\begin{aligned}
U: \quad & 1 \neq 2 \wedge 2 \neq 3 \wedge 3 \neq 1, && (15)\\
A: \quad & x = 1 \vee x = 2 \vee x = 3, && (16)\\
& C(1) \vee C(2) \vee C(3), && (17)\\
& P(x) \supset C(x), && (18)\\
& V(x) \supset C(x), && (19)\\
& M(x) \supset C(x), && (20)\\
& C(x) \wedge \neg E_1(x) \supset P(x), && (21)\\
& C(x) \wedge \neg E_2(x) \supset V(x), && (22)\\
& C(x) \wedge \neg E_3(x) \supset M(x), && (23)\\
& \neg P(1), && (24)\\
& \neg V(3). && (25)\\
& \neg M(2). && (26)
\end{aligned}$$

According to Gelfond and Lifschitz's method, $Resolve(A) \cup Replace(A)$ is the following π:

$$\begin{aligned}
\pi: \quad & C(1) \vee C(2) \vee C(3), && (17)\\
& \neg C(x) \supset \overline{P}(x), && (18'')\\
& \neg C(x) \supset \overline{V}(x), && (19'')\\
& \neg C(x) \supset \overline{M}(x), && (20'')\\
& C(x) \wedge \neg E_1(x) \supset P(x), && (21)\\
& C(x) \wedge \neg E_2(x) \supset V(x), && (22)\\
& C(x) \wedge \neg E_3(x) \supset M(x), && (23)\\
& \overline{P}(1), && (24'')
\end{aligned}$$

$$\overline{V}(3), \tag{25''}$$
$$\overline{M}(2), \tag{26''}$$
$$C(1) \supset E_1(1), \tag{27}$$
$$C(3) \supset E_2(3), \tag{28}$$
$$C(2) \supset E_3(2). \tag{29}$$

Obviously π is not a stratified logic program because

- (17) is a positive disjunctive clause and cannot be transformed into a rule,
- (27), (28) and (29) violate the required priorities: $E_1 > C$, $E_2 > C$ and $E_3 > C$ respectively.

Therefore using their method, compilation of the circumscription **(C6)** fails, but we can compile it successfully if we use our method. We explain about this according to the process of compilation shown in Fig. 3.1 as follows.

1. Application of *Resolve*

   $$\{C(1) \supset E_1(1),\ \ C(3) \supset E_2(3),\ \ C(2) \supset E_3(2)\},$$

   is generated as $Resolve(A)$.

2. Application of *Add*

   Each clause in $Resolve(A)$ violates the required priorities as mentioned above. However, for example, A can be represented with respect to E_1 as follows:

   $$(A - \{C(x) \wedge \neg E_1(x) \supset P(x)\}) \cup \{C(x) \wedge \neg P(x) \supset E_1(x)\}.$$

   Now let us define $N(E_1)$ and Q as follows:

   $$N(E_1) \stackrel{def}{=} \widetilde{\forall}\,(A - \{C(x) \wedge \neg E_1(x) \supset P(x)\}),$$
   $$Q \stackrel{def}{=} \lambda x(C(x) \wedge \neg P(x)).$$

   Then, A is rewritten as follows:

   $$N(E_1) \wedge (Q \leq E_1),$$

   where $Q \leq E_1$ stands for $\forall x(C(x) \wedge \neg P(x) \supset E_1(x))$.

   Therefore, A is a solitary formula with respect to E_1. So we get

   $$Circum(\widetilde{\forall} A;\ E_1) \equiv N(E_1) \wedge (Q = E_1),$$

   where $Q = E_1$ stands for

   $$\forall x(C(x) \wedge \neg P(x) \equiv E_1(x)). \tag{30}$$

   Then from (24) and (30), we get

$$Circum(\widetilde{\forall} A;\ E_1) \models C(1) \equiv E_1(1).$$

In similar ways, we get for E_2 and E_3 as follows:

$$Circum(\widetilde{\forall} A;\ E_2) \models C(3) \equiv E_2(3),$$
$$Circum(\widetilde{\forall} A; E_3) \models C(2) \equiv E_3(2).$$

Therefore, we define $Add_1(A)$, $Add_2(A)$ and $Add_3(A)$ as follows:

$$Add_1(A) \stackrel{def}{=} C(1) \equiv E_1(1),$$
$$Add_2(A) \stackrel{def}{=} C(3) \equiv E_2(3),$$
$$Add_3(A) \stackrel{def}{=} C(2) \equiv E_3(2).$$

Then, according to Corollary 3.4 and Corollary 3.6, it holds that,

$$Circum(\widetilde{\forall}\ A \wedge U; E_1 > E_2 > E_3 > C;\ P, V, M)$$
$$\equiv Circum(\widetilde{\forall}\ (A \wedge Add_1(A) \wedge Add_2(A) \wedge Add_3(A) \wedge Resolve(A)) \wedge U;$$
$$E_1 > E_2 > E_3 > C;\ P, V, M).$$

Here, if we expand $\widetilde{\forall}\ (A \wedge Add_1(A) \wedge Add_2(A) \wedge Add_3(A) \wedge Resolve(A))$ by using the domain closure axiom (16), we can transform **(C6)** into the following circumscription:

$$Circum((C(1) \equiv E_1(1)) \wedge (C(3) \equiv E_2(3)) \wedge (C(2) \equiv E_3(2)) \wedge A^1 \wedge U;$$
$$\mathbf{E_1} > \mathbf{E_2} > \mathbf{E_3} > \mathbf{C};\ \mathbf{P}, \mathbf{V}, \mathbf{M}), \qquad (\mathbf{C7})$$

where $\mathbf{E_1}, \mathbf{E_2}, \mathbf{E_3}$ represent

$$E_1(1), E_1(2), E_1(3);\ E_2(1), E_2(2), E_2(3);\ E_3(1), E_3(2), E_3(3),$$

C, P, V, M represent

$$C(1), C(2), C(3);\ P(1), P(2), P(3);\ V(1), V(2), V(3);\ M(1), M(2), M(3),$$

respectively and A^1 is as follows:

$$A^1: \quad \forall x(x = 1 \vee x = 2 \vee x = 3), \qquad (16)$$
$$C(1) \vee C(2) \vee C(3), \qquad (17)$$
$$P(2) \supset C(2), \qquad (18_2)$$
$$P(3) \supset C(3), \qquad (18_3)$$
$$V(1) \supset C(1), \qquad (19_1)$$
$$V(2) \supset C(2), \qquad (19_2)$$
$$M(1) \supset C(1), \qquad (20_1)$$
$$M(3) \supset C(3), \qquad (20_3)$$
$$C(1) \wedge \neg E_1(1) \supset P(1), \qquad (21_1)$$
$$C(2) \wedge \neg E_1(2) \supset P(2), \qquad (21_2)$$

$$C(3) \wedge \neg E_1(3) \supset P(3), \tag{21_3}$$
$$C(1) \wedge \neg E_2(1) \supset V(1), \tag{22_1}$$
$$C(2) \wedge \neg E_2(2) \supset V(2), \tag{22_2}$$
$$C(3) \wedge \neg E_2(3) \supset V(3), \tag{22_3}$$
$$C(1) \wedge \neg E_3(1) \supset M(1), \tag{23_1}$$
$$C(2) \wedge \neg E_3(2) \supset M(2), \tag{23_2}$$
$$C(3) \wedge \neg E_3(3) \supset M(3), \tag{23_3}$$
$$\neg P(1), \tag{24}$$
$$\neg V(3), \tag{25}$$
$$\neg M(2). \tag{26}$$
$$C(1) \supset E_1(1), \tag{27}$$
$$C(3) \supset E_2(3), \tag{28}$$
$$C(2) \supset E_3(2). \tag{29}$$

Remarks In A^1, clauses $P(1) \supset C(1)$, $V(3) \supset C(3)$ and $M(2) \supset C(2)$ are eliminated because they are subsumed clauses of $\neg P(1), \neg V(3), \neg M(2)$ respectively.

3. Application of *Equiv*

   Next, we apply *Equiv* according to Corollary 3.8. Then **(C7)** is transformed into the following circumscription **(C8)**:

   $Circum(A^2 \wedge U; \mathbf{E_1} > \mathbf{E_2} > \mathbf{E_3};\ \mathbf{P}, \mathbf{V}, \mathbf{M})$

   $$\wedge(C(1) \equiv E_1(1)) \wedge (C(3) \equiv E_2(3)) \wedge (C(2) \equiv E_3(2)), \qquad (\mathbf{C8})$$

   where A^2 is as follows:

A^2 :
$$\forall x(x = 1 \vee x = 2 \vee x = 3), \tag{16}$$
$$E_1(1) \vee E_3(2) \vee E_2(3), \tag{$17'$}$$
$$P(2) \supset E_3(2), \tag{$18'_2$}$$
$$P(3) \supset E_2(3), \tag{$18'_3$}$$
$$V(1) \supset E_1(1), \tag{$19'_1$}$$
$$V(2) \supset E_3(2), \tag{$19'_2$}$$
$$M(1) \supset E_1(1), \tag{$20'_1$}$$
$$M(3) \supset E_2(3), \tag{$20'_3$}$$
$$E_3(2) \wedge \neg E_1(2) \supset P(2), \tag{$21'_2$}$$
$$E_2(3) \wedge \neg E_1(3) \supset P(3), \tag{$21'_3$}$$
$$E_1(1) \wedge \neg E_2(1) \supset V(1), \tag{$22'_1$}$$

$$E_3(2) \wedge \neg E_2(2) \supset V(2), \tag{$22'_2$}$$
$$E_1(1) \wedge \neg E_3(1) \supset M(1), \tag{$23'_1$}$$
$$E_2(3) \wedge \neg E_3(3) \supset M(3), \tag{$23'_3$}$$
$$\neg P(1), \tag{24}$$
$$\neg V(3), \tag{25}$$
$$\neg M(2). \tag{26}$$

Remarks Clauses $(21_1), (22_3), (23_2), (27), (28)$ and (29) are eliminated because they become tautology by replacing $C(1),\ C(3),\ C(2)$ by $E_1(1),\ E_2(3),\ E_3(2)$.

4. Application of *Replace*

Since both $E_1(1)$ and $E_2(3)$ have higher priorities than $E_3(2)$ in the circumscription **(C8)**, a clause (17') can be transformed into the following rule:

$$\neg E_1(1) \wedge \neg E_2(3) \supset E_3(2).$$

Consequently, A^2 is compiled into the following stratified logic program Π.

Π :
$$\neg E_1(1) \wedge \neg E_2(3) \supset E_3(2), \tag{$17''$}$$
$$\neg E_3(2) \supset \overline{P}(2), \tag{$18''_2$}$$
$$\neg E_2(3) \supset \overline{P}(3), \tag{$18''_3$}$$
$$\neg E_1(1) \supset \overline{V}(1), \tag{$19''_1$}$$
$$\neg E_3(2) \supset \overline{V}(2), \tag{$19''_2$}$$
$$\neg E_1(1) \supset \overline{M}(1), \tag{$20''_1$}$$
$$\neg E_2(3) \supset \overline{M}(3), \tag{$20''_3$}$$
$$E_3(2) \wedge \neg E_1(2) \supset P(2), \tag{$21'_2$}$$
$$E_2(3) \wedge \neg E_1(3) \supset P(3), \tag{$21'_3$}$$
$$E_1(1) \wedge \neg E_2(1) \supset V(1), \tag{$22'_1$}$$
$$E_3(2) \wedge \neg E_2(2) \supset V(2), \tag{$22'_2$}$$
$$E_1(1) \wedge \neg E_3(1) \supset M(1), \tag{$23'_1$}$$
$$E_2(3) \wedge \neg E_3(3) \supset M(3), \tag{$23'_3$}$$
$$\overline{P}(1), \tag{$24''$}$$
$$\overline{V}(3), \tag{$25''$}$$
$$\overline{M}(2). \tag{$26''$}$$

The stratification of Π is as follows:

$$\mathbf{E_1};\quad \mathbf{E_2};\quad \mathbf{E_3};\quad \mathbf{P}, \mathbf{V}, \mathbf{M}, \overline{\mathbf{P}}, \overline{\mathbf{V}}, \overline{\mathbf{M}}$$

where $\overline{\mathbf{P}}, \overline{\mathbf{V}}, \overline{\mathbf{M}}$ denote

$$\overline{P}(1), \overline{P}(2), \overline{P}(3); \overline{V}(1), \overline{V}(2), \overline{V}(3); \overline{M}(1), \overline{M}(2), \overline{M}(3).$$

The perfect model of Π is as follows:

$$\left\{E_3(2),\ \overline{P}(3),\ \overline{V}(1),\ \overline{M}(1),\ \overline{M}(3),\ P(2),\ V(2),\ \overline{P}(1),\ \overline{V}(3),\ \overline{M}(2)\right\}.$$

5. Query evaluation w.r.t. the prioritized circumscription

   Let *Circum* denote the Circumscription **(C6)** and *Ans* be the response of the logic programming interpreter of Π. Then, the following results are derived:

   (a) $Circum \models C(2)$, $Circum \models \neg C(1)$, $Circum \models \neg C(3)$,
   since $Ans(\Pi, E_3(2))$=yes, $Ans(\Pi, E_1(1))$=no and $Ans(\Pi, E_2(3))$=no;

   (b) i. $Circum \models P(2)$, $Circum \models V(2)$,
   since $Ans(\Pi, P(2))$=yes and $Ans(\Pi, V(2))$=yes;

   ii. $Circum \models \neg M(2)$,
   since $Ans(\Pi, \overline{M}(2))$=yes.

   Thus we have obtained the most preferable solution that "the meeting is scheduled for day 2 and the president as well as the vice president attend the meeting on day 2, but the manager does not."

3.5 Discussion

Gelfond and Lifschitz's method is very efficient by making use of logic programming. But, its applicable class is restricted due to some strong assumptions about the syntax of circumscriptive theory. In this chapter, we showed the extended one of their method which expands the applicable class of their method with keeping its efficiency.

In the following, some comparisons to other works which do not make use of logic programming are mentioned.

- Przymusinski presents an algorithm for prioritized circumscription in his paper [Przymusinski, 1989]. His algorithm is based on the result that the prioritized circumscription can be equivalently represented by a conjunction of multiple parallel circumscriptions and therefore, the prioritized circumscription is computed by invoking recursively his parallel circumscriptive theorem prover based on MILO-resolution at each prioritization level. That is, the priorities of the predicates are not made full use in the strategy of the prover, which might lead to inefficiency of his algorithm w.r.t. prioritized circumscription.

- Baker and Ginsberg propose a theorem prover for prioritized circumscription [Baker and Ginsberg, 1989]. But since their algorithm is theoretically based on the computation of the labels of ATMS, the application of their algorithm is limited to the theories which are logically equivalent to a class of the propositional logic. On the other hand, our method as well as Gelfond and Lifschitz's can be applied to some class of theories of the first-order predicate logic.

However, the applicable class of our method presented in this chapter is still limited since the target language to which circumscription is compiled is a class of stratified logic programs. To overcome this problem, we provide our second method to compute circumscription by logic programming, which is shown in the next chapter.

3.6 Proofs

In this section, we show the proofs of Theorems 3.7 and 3.12.

Definition 3.16 *(comparability of interpretations)*
Let M and M' be interpretations. M and M' are *comparable* with respect to a tuple of predicate constants $\mathbf{P}$ if and only if the following conditions are satisfied.

1. M and M' have the same domain D.
2. For every individual constant, function constant, and predicate constant not in $\mathbf{P}$, M and M' have the same interpretation.

Proof of Theorem 3.7 :
Let us define T and A_1 as follows:

$$T \stackrel{def}{=} \forall x(P_m(x) \equiv P_l(x)) \wedge A(P_1, \ldots, P_l, \ldots, P_m, \ldots, P_k, Z),$$
$$A_1 \stackrel{def}{=} A(P_1, \ldots, P_l, \ldots, P_l, \ldots, P_k, Z).$$

Then the left-hand and the right-hand formulas of equivalence to be proved, are rewritten as follows:

$$Circum(T; P_1 > \ldots > P_l > \ldots > P_{m-1} > P_m > \ldots > P_k; Z), \quad (1)$$
$$Circum(A_1; P_1 > \ldots > P_l > \ldots > P_{m-1} > P_{m+1} > \ldots > P_k; Z) \wedge \forall x(P_m(x) \equiv P_l(x)), \quad (2)$$

where $T = \forall x(P_m(x) \equiv P_l(x)) \wedge A_1$.

1. ($\Rightarrow$) Let $\mathcal{M}$ be a model of (1). Then,

$$\mathcal{M} \models Circum(T; P_1 > \ldots > P_l > \ldots > P_{m-1} > P_m > \ldots > P_k; Z). \quad (3)$$

This means the following condition: Let M be any model of T which is *comparable* with $\mathcal{M}$ with respect to a tuple of predicate constants $P_1, \ldots, P_k, Z$,

and does not have the same interpretation for $P_1, \ldots, P_k$ as $\mathcal{M}$. Then the following condition (a) holds:

Condition (a) There exists some i for any M where $1 \leq i \leq k$,

$$\mathcal{M}[P_i] \subset M[P_i];$$

and for all j where $1 \leq j \leq i-1$ and $i \neq 1$,

$$\mathcal{M}[P_j] = M[P_j];$$

where the symbol $\subset$ indicates a *strict* set inclusion.

First, we will show that, for any model of (1), it follows that

$$i \neq m,$$

w.r.t the condition (a).

Now we assume $i = m$ for some model M' of T which is *comparable* with $\mathcal{M}$ with respect to a tuple of predicate constants $P_1, \ldots, P_k, Z$, and does not have the same interpretation for $P_1, \ldots, P_k$ as $\mathcal{M}$. Then, for M',

$$\mathcal{M}[P_m] \subset M'[P_m]; \tag{4}$$

and for all j where $1 \leq j \leq m-1$ and $m \neq 1$,

$$\mathcal{M}[P_j] = M'[P_j]. \tag{5}$$

Therefore, it can be obtained from (5) that

$$\mathcal{M}[P_l] = M'[P_l];$$

since $l < m$. This contradicts (4) since

$$\mathcal{M} \models \forall x(P_m(x) \equiv P_l(x)),$$
$$M' \models \forall x(P_m(x) \equiv P_l(x)).$$

Therefore

$$i \neq m.$$

Then condition (a) becomes the following condition (a'):

Condition (a') There exists some i for any M where $1 \leq i \leq k$ *and* $i \neq m$,

$$\mathcal{M}[P_i] \subset M[P_i];$$

and for all j where $1 \leq j \leq i-1$ and $i \neq 1$,

$$\mathcal{M}[P_j] = M[P_j].$$

It is apparent that $\mathcal{M}$ is a model of A_1. Therefore from the condition (a'),

$$\mathcal{M} \models Circum(A_1; P_1 > \ldots > P_l > \ldots > P_{m-1} > P_{m+1} > \ldots > P_k; Z).$$

As a result,

$$\mathcal{M} \models Circum(A_1; P_1 > \ldots > P_l > \ldots > P_{m-1} > P_{m+1} > \ldots > P_k; Z) \wedge \forall x(P_m(x) \equiv P_l(x)).$$

2. ($\Leftarrow$) With respect to from right to left, it can also be proved in a similar way. □

Proof of Theorem 3.12 :

According to Corollary 3.4, Corollary 3.6 and Corollary 3.8, it is apparent that the following circumscription:

$$Circum \stackrel{def}{=} Circum(\widetilde{\forall} A \wedge U; P^1 > \ldots > P^k; Z),$$

is logically equivalent to

$$Circum(\widetilde{\forall} A' \wedge U; H^1 > \ldots > H^n; Z) \wedge \bigwedge_i \forall x(Q_i(x) \equiv P_i(x)), \tag{1}$$

where the predicate symbol Q_i occurs in A, but does not occurs in A'.

Now, we need to determine whether $Circum$ implies W or $\neg W$, where W is a ground atomic formula whose predicate symbol occurs in A. That is,

$$Circum \models W \quad or \quad \neg W \quad ? \tag{2}$$

Let M be any model of $Circum$, then (2) denotes that

$$M \models W \quad or \quad \neg W \quad ? \tag{3}$$

W.r.t. the model M, it follows that

$M \models Circum(\widetilde{\forall} A' \wedge U; H^1 > \ldots > H^n; Z)$, (4)

$M \models \forall x(Q_i(x) \equiv P_i(x))$. (5)

Let us examine the following two cases.

1. If the predicate symbol of W occurs in A', then $W \neq Q_i(a)$ for any tuple a of individual constants in U. According to (3) and (4), the problem about (2) is transformed into

$$Circum(\widetilde{\forall} A' \wedge U; H^1 > \ldots > H^n; Z) \models W \quad or \quad \neg W \quad ? \tag{6}$$

   W.r.t. the problem about (6), we can immediately make use of Gelfond and Lifschitz's main theorem. That is, since Π is equivalent to $Resolve(A') \cup Replace(A')$, we can derive the part 2 of Theorem 3.12 by using the value of $Ans(\Pi, W)$ for any W such as $W \neq Q_i(a)$.

2. If the predicate symbol of W does not occur in A', then $W = Q_i(a)$ for any tuple a of individual constants in U. According to (5),

$$M \models Q_i(t) \quad iff \quad M \models P_i(t), \tag{7}$$

$$M \models \neg Q_i(t) \quad iff \quad M \models \neg P_i(t). \tag{8}$$

Since the predicate symbol of P_i occurs in A', (a) and (b) of the part 2 in Theorem 3.12, which is proved above, can be used w.r.t P_i as follows: for a tuple a of individual constants in U,

$$Ans(\Pi, P_i(a)) = yes \qquad iff \qquad M \models P_i(a), \tag{9}$$

$$Ans(\Pi, P_i(a)) = no \qquad iff \qquad M \models \neg P_i(a), \tag{10}$$

By using (7) and (8), (9) and (10) can be rewritten as follows:

$$Ans(\Pi, P_i(a)) = yes \qquad iff \qquad M \models Q_i(a), \tag{11}$$

$$Ans(\Pi, P_i(a)) = no \qquad iff \qquad M \models \neg Q_i(a). \tag{12}$$

Thus, the part 1 of Theorem 3.12 is derived since M in (11) and (12) is a model of $Circum$. □

Chapter 4

Compiling Circumscription into ELP

4.1 Introduction

Gelfond and Lifschitz [1988a] were the first to explore the compiling approach to compute circumscription based on the relationship between the semantics of circumscription and the semantics of logic programs. They presented a method of compiling some restricted class of prioritized circumscription into a stratified logic program. Though their method is computationally efficient, the applicable class is too limited.

So, as discussed in Chapter 3, we proposed an extension of Gelfond and Lifschitz's method which also compile prioritized circumscription into a stratified logic program [Wakaki and Satoh, 1995]. With keeping the efficiency of Gelfond and Lifschitz's method, this method expands the applicable class of circumscription by making use of the Lifschitz's result [Lifschitz, 1985] about parallel circumscription of a solitary formula. However, as far as a class of stratified logic programs is considered as the target language to which circumscription is compiled, the applicable class is limited within a class of circumscription which has a unique minimal model since every stratified logic program has a unique perfect model [Przymusinski, 1987]. But there are many examples of nonmonotonic reasoning whose intended meaning cannot be represented by a unique model such as *multiple extension problem*, a class of circumscription which has fixed predicates, and so on.

Recently Sakama and Inoue [1995; 1996] proposed two methods both of which compile circumscription into classes of more general logic programs whose semantics are given by stable models for the first one [Sakama and Inoue, 1995] and by preferred answer sets for the second one [Sakama and Inoue, 1996]. Though both of their methods can handle the multiple extension problem as well as circumscription with fixed predicates, the first one is only applicable to parallel circumscription but not to prioritized circumscription. On the other hand, the second one is applicable to prioritized circumscription, but it gives only the semantic aspects and is lack of the feasible logic programming interpreter for prioritized logic programs proposed by them as the target language into which prioritized circumscription is

compiled.

In this chapter, we present a new method of compiling circumscription into *extended logic programs* proposed by Gelfond and Lifschitz [1991] as its target language. It is widely applicable to a class of parallel circumscription as well as a class of prioritized circumscription. Showing the semantic correspondence between circumscription with fixed predicates and Reiter's default theory which generalizes Theorem 2 in [Etherington, 1987a] (see Theorem 2.18 in chapter 2), we can give not only the semantic relationship between a class of parallel circumscription and a class of extended logic programs but also the one between a class of prioritized circumscription and a class of extended logic programs. As a result, circumscription whose theory contains both the domain closure axiom and the uniqueness of names axioms and does not contain function symbols can always be compiled into an extended logic program Π, so that, whether a ground literal is provable from circumscription, can always be evaluated by deciding whether the literal is true in all answer sets of Π.* This can be computed by running Π under the existing logic programming interpreter such as Satoh and Iwayama's top-down query evaluation procedure for abductive logic programming [Satoh and Iwayama, 1992].

Finally, we present that our approach exploiting classical negation $\neg$ can also give an extension of Sakama and Inoue's first method [1995] to make it possible to compute prioritized circumscription.

This chapter is organized as follows. In Section 4.2, we provide two syntactical definitions of extended logic programs into which parallel circumscription and prioritized circumscription are compiled respectively. Then, we present theorems and corollaries on which our method is theoretically based, along with examples. In Section 4.3, we compare our method with related researches. We finish Section 4.4 with giving proofs of theorems in this chapter.

4.2 Translation from Circumscription to ELP

We give a translation from circumscription to extended logic programs. Parallel circumscription is translated into ELP Π^{α} and prioritized circumscription is translated into ELP Π^{β} respectively.

Firstly, we show the definition of *characteristic clauses* [Inoue, 1992a] which are needed in our translation.

Definition 4.1 [Inoue, 1992a] Let Σ be a set of clauses. Then $Th(\Sigma)$ stands for a set of clauses which are theorems of Σ. A clause in $Th(\Sigma)$ which is not properly subsumed by any theorem in $Th(\Sigma)$ is called a *characteristic clause*. $\mu Th(\Sigma)$ denotes the set of all characteristic clauses in $Th(\Sigma)$.

In the following Definition 4.2 and Definition 4.3, we restrict a first order theory T to the one which is function-free and contains both the *domain closure axiom*

* In this paper, we call this methd "Wakaki and Satoh's method.

(DCA) and *uniqueness of names axioms* (UNA). We consider T as a set of clauses as follows:

$$T \stackrel{def}{=} \Sigma \cup \mathrm{DCA} \cup \mathrm{UNA}. \tag{4.1}$$

Let Σ^{DCA} denote a set of ground clauses such that every clause in Σ containing variables is replaced by all its ground instances obtained by substituting every ground term in DCA for each variable. t stands for a tuple of ground terms occurring in DCA.

Definition 4.2 Given parallel circumscription $Circum(\widetilde{\forall}\, T; P; Z)$ where $\widetilde{\forall}$ is a universal closure, ELP Π^{α} is constructed as follows:

1. For any minimized predicate p from P and any t, Π^{α} has a rule:

$$\neg p(t) \leftarrow \ not\ p(t).$$

2. For any fixed predicate q from Q and any t, Π^{α} has two rules as follows:

$$\begin{aligned} \neg q(t) &\leftarrow \ not\ q(t), \\ q(t) &\leftarrow \ not\ \neg q(t). \end{aligned}$$

3. For any characteristic clause $C \stackrel{def}{=} \ell_1 \vee \ell_2 \vee \ldots \vee \ell_n$ in $\mu Th(\Sigma^{DCA})$ and any contrapositive of C such as:

$$\neg\ell_1 \wedge \ldots \wedge \neg\ell_{i-1} \wedge \neg\ell_{i+1} \wedge \ldots \wedge \neg\ell_n \supset \ell_i \qquad (1 \leq i \leq n),$$

Π^{α} has rules as follows:

$$\ell_i \leftarrow \ \neg\ell_1 \wedge \ldots \wedge \neg\ell_{i-1} \wedge \neg\ell_{i+1} \wedge \ldots \wedge \neg\ell_n.$$

Definition 4.3 Given prioritized circumscription as follows:

$$Circum(\widetilde{\forall}\, T(P^1 \ldots P^k, Z, Q); P^1 > P^2 > \ldots > P^k; Z),$$

where $P^r (1 \leq r \leq k)$,Z and Q are tuples of predicate symbols such as $[(P^r)_1, \ldots, (P^r)_{\ell_r}]$, $[Z_1, \ldots, Z_m]$ and $[Q_1, \ldots, Q_n]$. ELP Π^{β} is constructed in the following two steps:

1. According to Theorem 2.10, a given prioritized circumscription is represented by the conjunction of k parallel circumscriptions. So, let every ith $(1 \leq i \leq k)$ parallel circumscription,

$$Circum(\widetilde{\forall}\, T(P^1 \ldots P^k, Z, Q); P^i; P^{i+1}, \ldots, P^k, Z)$$

be transformed in such a way that all predicate symbols occurring in it are renamed by using $Pi^1, \ldots, Pi^k$, Zi,Qi instead of $P^1, \ldots, P^k$, Z,Q, which leads to as follows:

$$Circum(\widetilde{\forall}\, Ti; Pi^i; Pi^{i+1}, \ldots, Pi^k, Zi), \tag{4.2}$$

where Ti denotes $T(Pi^1, \ldots, Pi^k, Zi, Qi)$, and Pi^r, Zi, Qi are tuples of predicate symbols such as $[(Pi^r)_1, \ldots, (Pi^r)_{\ell_r}]$,$[(Zi)_1, \ldots, (Zi)_m]$, $[(Qi)_1, \ldots, (Qi)_n]$.

2. ELP Π^β consists of all rules from $(\Pi^\alpha)_1, \ldots, (\Pi^\alpha)_k$ and Π_γ where

   (a) each $(\Pi^\alpha)_i$ is an extended logic program (ELP) which is constructed from the ith renamed parallel circumscription (4.1) according to Definition 4.2.

   (b) Π_γ is an extended logic program which consists of the following rules: For any predicate u from P, Z, Q and any t,

$$u(t) \leftarrow u_i(t),$$
$$\neg u(t) \leftarrow \neg u_i(t). \qquad (\ 1 \leq i \leq k)$$

   where u and u_i stand for any predicate symbol of $(P^r)_f, Z_g, Q_h$ and any one of $(Pi^r)_f, (Zi)_g, (Qi)_h$ respectively.
   $(1 \leq f \leq \ell_r, 1 \leq g \leq m, 1 \leq h \leq n)$

First of all, we show the following theorem which generalizes Theorem 2.18 (i.e. Theorem 2 in [Etherington, 1987a]) based on Theorem 1 in [de Kleer and Konolige, 1989].

Theorem 4.4 *Assume that T is a first order theory including DCA and UNA. Let P,Q and Z be a tuple of minimized predicates, a tuple of fixed predicates and a tuple of variable predicates respectively. Then, for any first-order formula F of the language of T, it holds that,*

$$Circum(T; P; Z) \models F$$
iff for any extension E of a default theoryΔ, $E \models F$
where Δ is defined as follows :
$$\Delta \stackrel{def}{=} \left(\left\{ \frac{: \neg p(x)}{\neg p(x)}, \frac{: \neg q(x)}{\neg q(x)}, \frac{: q(x)}{q(x)} \middle| p \in P, q \in Q \right\}, T \right)$$

Based on Theorem 4.4 as well as a 1-1 correspondence between the extensions of a default theory and the answer sets of an extended logic program shown in [Gelfond and Lifschitz, 1991], the semantic relationships between circumscription and the translated ELPs Π^α,Π^β are given as follows.

Theorem 4.5 *For any ground literal G of the language of T whose predicate symbol is not equality, it holds that,*

$Circum(\widetilde{\forall} T; P; Z) \models G$
iff *G is true in all answer sets of* ELP Π^α,
where ELP Π^α *is constructed from $Circum(\widetilde{\forall} T; P; Z)$ by using definition 4.2.*

Theorem 4.6 *For any ground literal G of the language of T whose predicate symbol is not equality, it holds that,*

$Circum(\widetilde{\forall}T; P^1 > P^2 > \ldots > P^k; Z) \models G$
iff G *is true in all answer sets of* ELP Π^β.

where ELP Π^β *is constructed from* $Circum(\widetilde{\forall}T; P^1 > P^2 > \ldots > P^k; Z)$, *by using Definition 4.3.*

Remarks. In this thesis, we refer to Theorem 4.5 and Theorem 4.6 as "Wakaki and Satoh's method".

Satoh and Iwayama [1992] show that whether a ground atom G is true in all stable models of a normal logic program Π, can be decided by running their top-down query evaluation procedure for abductive logic programs where an abductive framework is given by $\langle \Pi, A \rangle$ in which A is a set of predicate symbols called abducible predicates. Their result is given as follows:

Suppose that a normal logic program Π is consistent, which means that there exists a stable model of Π, and all ground rules obtained by replacing all variables in each rule in Π by every element of its Herbrand universe are finite. Then it holds that,

G *is true in all stable models of* Π
iff *derive*$(not\ G, \{\})$ *fails*
under the abductive framework $\langle \Pi, \{\} \rangle$.

where *derive* is a procedure given by them and *not* in its first argument denotes the negation-as-failure operator.

Since all ground rules in ELP Π^α and Π^β constructed by using Definition 4.2 and Definition 4.3 respectively are finite, we can make use of their result in a slightly extended form, and Theorem 4.5 and Theorem 4.6 can be said in other words by the following corollaries.

Corollary 4.7 *Suppose that T is consistent and function-free. For any ground literal G of the language of T whose predicate symbol is not equality, it holds that,*

$Circum(\widetilde{\forall}T; P; Z) \models G$
iff *derive*$(not\ G, \{\})$ *fails*
under the abductive framework $\langle \Gamma^\alpha, \{\} \rangle$,

where Γ^α *consists of all rules of* ELP Π^α *constructed from* $Circum(\widetilde{\forall}T; P; Z)$ *by using Definition 4.2 plus rules,* $\leftarrow u(t), \neg u(t)$, *for every predicate u in T.*

Corollary 4.8 *Suppose that T is consistent and function-free. For any ground literal G of the language of T whose predicate symbol is not equality, it holds that,*

$Circum(\widetilde{\forall}T; P^1 > P^2 > \ldots > P^k; Z) \models G$
iff *derive*$(not\ G, \{\})$ *fails*
under the abductive framework $\langle \Gamma^\beta, \{\} \rangle$.

where Γ^β *consists of all rules of* ELP Π^β *constructed from* $Circum(\widetilde{\forall}T; P^1 > P^2 > \ldots > P^k; Z)$ *by using Definition 4.3 plus rules,* $\leftarrow u(t), \neg u(t)$, *for every predicate u in T.*

Remarks. There is Ginsberg's early work [Ginsberg, 1989] on evaluating circumscription using abduction. But his method does not exploit the logic programming.

Example 4.9
Our compilation can handle circumscription representing *multiple inheritance*. Consider parallel circumscription:

$$Circum(\widetilde{\forall}T; ab_1, ab_2; pacifist) \quad (1)$$

where $T \stackrel{def}{=} \Sigma \cup$ DCA $\cup$ UNA. Σ consists of the following clauses and DCA is $x = Nixon$:

$$pacifist(x) \vee ab_1(x) \vee \neg quaker(x),$$
$$\neg pacifist(x) \vee ab_2(x) \vee \neg republican(x),$$
$$republican(Nixon), \qquad quaker(Nixon).$$

A set Q of fixed predicates is $\{republican, quaker\}$ in this example. Hereafter we abbreviate *pacifist*, *republican*, *quaker* and *Nixon* to *pac*, *rep*,*quak* and N respectively.

After replacing a variable x in every clause in Σ by a ground term N in DCA, $\mu Th(\Sigma^{DCA})$ is obtained as follows:

$$pac(N) \vee ab_1(N),$$
$$\neg pac(N) \vee ab_2(N),$$
$$ab_1(N) \vee ab_2(N),$$
$$rep(N), \qquad quak(N).$$

According to Definition 4.2, let us construct ELP Π^{α} from circumscription (1).

1. For minimized predicates ab_1, ab_2 and a term N,
$$\neg ab_1(N) \leftarrow not\ ab_1(N),$$
$$\neg ab_2(N) \leftarrow not\ ab_2(N),$$

2. For fixed predicates *rep*, *quak* and a term N,
$$\neg rep(N) \leftarrow not\ rep(N),$$
$$rep(N) \leftarrow not\ \neg rep(N),$$
$$\neg quak(N) \leftarrow not\ quak(N),$$
$$quak(N) \leftarrow not\ \neg quak(N),$$

3. For all contrapositives of all clauses of $\mu Th(\Sigma^{DCA})$,
$$ab_1(N) \leftarrow \neg pac(N),$$
$$pac(N) \leftarrow \neg ab_1(N),$$
$$ab_2(N) \leftarrow pac(N),$$
$$\neg pac(N) \leftarrow \neg ab_2(N),$$
$$ab_1(N) \leftarrow \neg\ ab_2(N),$$
$$ab_2(N) \leftarrow \neg\ ab_1(N),$$
$$rep(N) \leftarrow, \qquad quak(N) \leftarrow .$$

As a result, Π^α has two answer sets:

$$\{\neg ab_1(N), ab_2(N), pac(N), rep(N), quak(N)\},$$
$$\{ab_1(N), \neg ab_2(N), \neg pac(N), rep(N), quak(N)\}.$$

Example 4.10 Let us compile the following prioritized circumscription:

$$Circum(\{p \vee q, q \vee r\}; p > q; r)$$

It holds that $\Sigma = \mu Th(\Sigma)$ since $\Sigma = \{p \vee q, q \vee r\}$.
Contrapositives of $p \vee q$ are as follows:

$$\neg q \supset p, \qquad \neg p \supset q,$$

and those of $q \vee r$ are as follows:

$$\neg r \supset q, \qquad \neg q \supset r.$$

According to Theorem 2.10, it holds that

$$\begin{aligned} &Circum(\{p \vee q, q \vee r\}; p > q; r) \\ &\equiv Circum(\{p \vee q, q \vee r\}; p; q, r) \\ &\qquad \wedge Circum(\{p \vee q, q \vee r\}; q; r). \end{aligned}$$

Then according to Definition 4.3, predicate symbols occurring each parallel circumscription are renamed as follows:

$$Circum(\{p1 \vee q1, q1 \vee r1\}; p1; q1, r1), \tag{2}$$
$$Circum(\{p2 \vee q2, q2 \vee r2\}; q2; r2). \tag{3}$$

Therefore ELPs $(\Pi^\alpha)_1$, $(\Pi^\alpha)_2$ constructed from (2) and (3) respectively as well as Π_γ are obtained as follows:

$$\begin{aligned} (\Pi^\alpha)_1 : \quad & \neg p1 \leftarrow \ not\ p1, \\ & p1 \leftarrow \ \neg\, q1, \qquad q1 \leftarrow \ \neg\, p1, \\ & q1 \leftarrow \ \neg\, r1, \qquad r1 \leftarrow \ \neg\, q1. \end{aligned}$$

$$\begin{aligned} (\Pi^\alpha)_2 : \quad & \neg q2 \leftarrow \ not\ q2, \\ & \neg p2 \leftarrow \ not\ p2, \\ & p2 \leftarrow \ not\ \neg p2, \\ & p2 \leftarrow \ \neg\, q2, \qquad q2 \leftarrow \ \neg\, p2, \\ & q2 \leftarrow \ \neg\, r2, \qquad r2 \leftarrow \ \neg\, q2. \end{aligned}$$

$$\begin{array}{lll}\Pi_\gamma: & p \leftarrow p1, & p \leftarrow p2,\\ & q \leftarrow q1, & q \leftarrow q2,\\ & r \leftarrow r1, & r \leftarrow r2,\\ & \neg p \leftarrow \neg p1, & \neg p \leftarrow \neg p2,\\ & \neg q \leftarrow \neg q1, & \neg q \leftarrow \neg q2,\\ & \neg r \leftarrow \neg r1, & \neg r \leftarrow \neg r2.\end{array}$$

ELP Π^β has the only one answer set:

$$\{\neg p, q, \neg p1, q1, \neg p2, q2\},$$

where $\Pi^\beta \stackrel{def}{=} (\Pi^\alpha)_1 \cup (\Pi^\alpha)_2 \cup \Pi_\gamma$.
Thus according to Theorem 4.6, we can conclude that

$$Circum(\{p \vee q, q \vee r\}; p > q; r) \models \neg p,$$
$$Circum(\{p \vee q, q \vee r\}; p > q; r) \models q,$$

but neither r nor $\neg r$ is provable from this prioritized circumscription.

In the following, we give an extension of Sakama and Inoue's first method [1995]. According to their method, parallel circumscription is translated into a *general disjunctive program* (GDP) whose semantics is given by stable models. Alternatively, they also show the translation of parallel circumscription into an *Extended disjunctive program* (EDP) whose semantics is given by the answer sets, which just corresponds to the stable models of the translated GDP. Each of their methods is applicable to a class of parallel circumscription, but not to prioritized circumscription. Our target language ELP has the expressive power of classical negation $\neg$, which enables us to compute prioritized circumscription as is shown in Theorem 4.6. Since the classical negation is also available to EDP, we can make use of our method to extend their alternative method whose target language is EDP, so that it may become applicable to a class of prioritized circumscription as follows.

Definition 4.11 [Sakama and Inoue, 1995]
Given parallel circumscription $Circum(\widetilde{\forall} T; P; Z)$, EDP Π^α is constructed as follows, where DCA and UNA are incorporated in a first order theory T since only its Herbrand models are considered, and $p_1, \ldots, p_\ell$, $z_1, \ldots, z_m$, and $q_1, \ldots, q_n$ are used to denote atoms from P,Z and Q respectively.

1. For any clause in T of the form:

$$p_1 \vee \ldots \vee p_\ell \vee z_1 \vee \ldots \vee z_m \vee q_1 \vee \ldots \vee q_n \vee \neg p_{\ell+1} \vee \ldots$$
$$\vee \neg p_s \vee \neg z_{m+1} \vee \ldots \vee \neg z_t \vee \neg q_{n+1} \vee \ldots \vee \neg q_u,$$

Π^α has the rule:

$$z_1 | \ldots | z_m | q_1 | \ldots | q_n \leftarrow p_{\ell+1}, \ldots, p_s, z_{m+1}, \ldots, z_t, q_{n+1},$$
$$\ldots, q_u, notp_1, \ldots, notp_\ell.$$

2. For every clause in $\mu Th(\Sigma)$ of the form:

$$p_1 \vee \ldots \vee p_\ell \vee q_1 \vee \ldots \vee q_n \vee \neg q_{n+1} \vee \ldots \vee \neg q_u,$$

Π^α has the rule:

$$p_1 | \ldots | p_\ell | q_1 | \ldots | q_n \leftarrow q_{n+1}, \ldots, q_u,$$

3. For any atom p, z, q, Π^α has the rule:

$$\neg p \leftarrow \ notp,$$
$$z \mid \neg z \leftarrow, \qquad q \mid \neg q \leftarrow .$$

Remarks. It is shown in [Sakama and Inoue, 1995] that there is a 1-1 correspondence between the PZ-minimal Herbrand models of parallel circumscription and the answer sets of the translated EDP Π^α.

Definition 4.12
Let T be a first order theory without function symbols including DCA and UNA. Then given prioritized circumscription:

$$Circum(\widetilde{\forall}\, T(P^1 \ldots P^k, Z, Q); P^1 > P^2 > \ldots > P^k; Z),$$

EDP Π^β is constructed in the following two steps:

1. This is the same as the first step given by Definition 4.3.

2. EDP Π^β consists of all rules from $(\Pi^\alpha)_1$,..., $(\Pi^\alpha)_k$ and Π_γ where

   (a) each $(\Pi^\alpha)_i$ is an extended disjunctive program (EDP) which is constructed from the ith renamed parallel circumscription (4.1) according to Definition 4.11.

   (b) Π_γ is the same set given by Definition 4.3.

Then the relationship between prioritized circumscription and the translated EDP Π^β are given as follows.

Theorem 4.13 *Let T be a first order theory without function symbols including DCA and UNA and F be a ground formula of the language of T in which equality does not occur as a predicate symbol. Then, it holds that, for any F,*

$Circum(\widetilde{\forall} T; P^1 > P^2 > \ldots > P^k; Z) \models F$
iff *F is true in all answer sets of* EDP Π^β.

where EDP Π^β *is constructed from* $Circum(\widetilde{\forall} T; P^1 > P^2 > \ldots > P^k; Z)$, *by using Definition 4.12.*

Example 4.14

Consider prioritized circumscription given in Example 4.10. We apply Theorem 4.13 instead of Theorem 4.6 to it. Then according to Definition 4.12, the translated EDPs $(\Pi^\alpha)_1$ and $(\Pi^\alpha)_2$ are as follows:
Since $P = \{p1\}$, $Z = \{q1, r1\}$, $Q = \phi$ in the renamed circumscription: $Circum(\{p1 \vee q1, q1 \vee r1\}; p1; q1, r1)$,

$$\begin{aligned}(\Pi^\alpha)_1 : \quad & q1 \leftarrow \ not\ p1, \\ & q1|\ r1, \\ & \neg p1 \leftarrow \ not\ p1, \\ & q1|\ \neg q1\ \leftarrow, \qquad r1|\ \neg r1\ \leftarrow .\end{aligned}$$

Since $P = \{q2\}$, $Z = \{r2\}$, $Q = \{p2\}$ in the renamed circumscription: $Circum(\{p2 \vee q2, q2 \vee r2\}; q2; r2)$,

$$\begin{aligned}(\Pi^\alpha)_2 : \quad & p2 \leftarrow \ not\ q2, \\ & r2 \leftarrow \ not\ q2, \\ & p2|\ q2 \leftarrow, \\ & \neg q2 \leftarrow not\ q2, \\ & r2|\ \neg r2\ \leftarrow, \qquad p2|\ \neg p2\ \leftarrow .\end{aligned}$$

Π_γ is obtained as the same one shown in Example 4.10.
As a result, EDP Π^β has the following two answer sets:

$$\{\neg p, q, r, \neg p1, q1, r1, \neg p2, q2, r2\},$$
$$\{\neg p, q, \neg r, \neg p1, q1, \neg r1, \neg p2, q2, \neg r2\},$$

where Π^β consists of all rules in the above $(\Pi^\alpha)_1$, $(\Pi^\alpha)_2$ and Π_γ. Both $\neg p$ and q are true in all of these answer sets, but so is neither r nor $\neg r$. Notice that this is the same evaluation result as the one in Example 4.10.

4.3 Related Works and Summary

In this chapter, we present a method of compiling circumscription into extended logic programs which is widely applicable to parallel circumscription as well as prioritized circumscription. Our method always enables us to compute any circumscription whose theory includes both DCA and UNA by compiling it into ELP Π, which can be evaluated by using the existing logic programming interpreter such as Satoh and Iwayama's top-down query evaluation procedure for abductive logic programming.

In the following, we compare our method with related researches from the viewpoints of the applicable class as well as the computational efficiency and feasibility. These researches are characterized by the semantics of a class of logic programs considered as the target language to which circumscription is compiled.

- Gelfond and Lifschitz's method [1988a] as well as our previous method shown in Chapter 3 [Wakaki and Satoh, 1995], are the most efficient since they are as efficient as the evaluation of a stratified logic program. But their applicable classes are limited within a class of circumscription which has a unique model as mentioned in the introduction of this chapter.

- In Sakama and Inoue's first method [1995], parallel circumscription is translated into a general disjunctive program (GDP). This method is applicable to a wide class of parallel circumscription, but inapplicable to prioritized circumscription though a class of GDP has the expressive power of the *positive occurrences* of negation as failure [Inoue and Sakama, 1994]. The most important difference between their GDP and our ELP is whether classical negation $\neg$ is available or not. Theorem 4.13 shows that our approach exploiting classical negation can also give an extension of their alternative method whose target language is EDP, so that it may become applicable to prioritized circumscription.

  As for the computational aspects, characteristic clauses should be computed in the compilation process of our method as well as their method whose time complexity is the exponential order.

- In Sakama and Inoue's second method [1996], parallel circumscription as well as prioritized circumscription is translated into prioritized logic programs proposed by them whose declarative meaning is given by preferred answer sets defined by them. Their method, however, is immature for the purpose of the automation of circumscription since their prioritized logic program is not feasible because their method gives only the semantic aspects, but procedural issues for the query evaluation are left as their future works.

Our future work is to implement our compiling method presented in this chapter as efficiently as possible.

4.4 Proofs

Proof of Theorem 4.4 :

According to Theorem 2.11 [de Kleer and Konolige, 1989], $Circum(T; P; Z)$ with a tuple Q of fixed predicates can be logically equivalently reduced into circumscription without fixed predicates by introducing a new tuple Q' of similar predicate symbols not in T. Therefore it holds that

$$\begin{aligned} &Circum(T(P,Q,Z); P; Z) \wedge \bigwedge_{q\in Q, q'\in Q'} \forall x(q(x) \equiv \neg q'(x)) \\ &\equiv Circum(T(P,Q,Z) \wedge \bigwedge_{q\in Q, q'\in Q'} \forall x(q(x) \equiv \neg q'(x)); P, Q, Q'; Z). \qquad (1) \end{aligned}$$

According to Theorem 2.18, i.e. [Etherington, 1987a,Theorem 2], it holds that, with respect to a right-hand circumscription of (1),

$$Circum(T(P,Q,Z) \wedge \bigwedge_{q\in Q, q'\in Q'} \forall x(q(x) \equiv \neg q'(x)); P, Q, Q'; Z) \models F$$

iff for any extension E_1 of a default theoryΔ_1, $E_1 \models F$

where

$$\Delta_1 \stackrel{def}{=} (\{: \neg p(x)/\neg p(x), : \neg q(x)/\neg q(x), : \neg q'(x)/\neg q'(x) \mid p \in P, q \in Q, q' \in Q'\}, \\ T \wedge \bigwedge_{q\in Q, q'\in Q'} \forall x(q(x) \equiv \neg q'(x)) \qquad (2)$$

Now, let us define a default theory Δ_2 as follows:

$$\Delta_2 \stackrel{def}{=} (\{: \neg p(x)/\neg p(x), : \neg q(x)/\neg q(x), : q(x)/q(x) \mid p \in P, q \in Q\}, \\ T \wedge \bigwedge_{q\in Q, q'\in Q'} \forall x(q(x) \equiv \neg q'(x)))$$

Then according to the definition of the default theory, we can easily prove a 1-1 correspondence that, for any extension E_1 of a default theory Δ_1, there exists an extension E_2 of a default theory Δ_2 such that

$$E_1 = E_2, \qquad (3)$$

and vice versa.

Since default rules of Δ_2 coincide with default rules of Δ as well as there occurs no predicate symbols of q' in both these default rules and T, we can also easily prove a 1-1 correspondence that, for any extension E_2 of a default theory $\Delta_2(P,Q,Q',Z)$, there exists an extension $E(P,Q,Z)$ of a default theory Δ such that

$$Th(E_2(P,Q,Q',Z)) = Th(E(P,Q,Z) \cup \bigwedge_{q\in Q, q'\in Q'} \{\forall x(q(x) \equiv \neg q'(x))\}) \qquad (4)$$

and vice versa.

Therefore according to (1),(2),(3),(4), it holds that,

$$Circum(T; P; Z) \wedge \bigwedge_{q\in Q, q'\in Q'} \forall x(q'(x) \equiv \neg q(x)) \models F$$

iff for any extension E_1 of a default theoryΔ_1, $E_1 \models F$ ($\because$ (1), (2))

iff for any extension E_2 of a default theoryΔ_2, $E_2 \models F$ ($\because$ (3))

iff for any extension E of a default theoryΔ,

$$E \wedge \bigwedge_{q\in Q, q'\in Q'} \forall x(q(x) \equiv \neg q'(x)) \models F \qquad (\because (4))$$

Thus since any predicate symbol $q' \in Q'$ does not occur in $Circum(T; P; Z)$, E and F, it holds that,

$Circum(T; P; Z) \models F$

iff for any extension E of a default theoryΔ, $E \models F$ $\square$

Before proving Theorem 4.5, firstly we provide Definition 4.15 and prove Lemma 4.16 as follows.

Definition 4.15 *Let Σ be a set of ground clauses and C be a characteristic clause from $\mu Th(\Sigma)$. We define a set D of default rules which are generated from all contrapositives of all characteristic clauses in $\mu Th(\Sigma)$ as follows.*

For every C whose form is as follows:

$$C: \quad \ell_1 \vee \ell_2 \vee \ldots \vee \ell_n$$

and every its contrapositive such as

$$\neg\ell_1 \wedge \ldots \wedge \neg\ell_{i-1} \wedge \neg\ell_{i+1} \wedge \ldots \wedge \neg\ell_n \supset \ell_i \qquad (i = 1, \ldots, n),$$

D has the inference rules of the following form:

$$\frac{\neg\ell_1, \ldots, \neg\ell_{i-1}, \neg\ell_{i+1}, \ldots, \neg\ell_n : }{\ell_i} \qquad (i = 1, \ldots, n)$$

where we identify $\neg(\neg p)$ with a ground atom p.

Lemma 4.16 *Let Σ be a consistent set of ground clauses, D be a set of default rules generated from Σ according to Definition 4.15 and $p_i \in Lit$ $(1 \leq i \leq n)$ where Lit is a set of ground literals in the language of Σ. Δ and Δ' are default theories defined as follows.*

$$\Delta \stackrel{def}{=} \left(\bigcup_{i=1}^{n} \left\{ \frac{: \neg p_i}{\neg p_i} \right\}, \Sigma \right)$$

$$\Delta' \stackrel{def}{=} \left(\bigcup_{i=1}^{n} \left\{ \frac{: \neg p_i}{\neg p_i} \right\} \cup D, \phi \right)$$

Then there is a 1-1 correspondence that for any extension E of a default theory Δ, there exists an extension E' of a default theory Δ' such that $E \cap Lit = E' \cap Lit$, and vice versa.

Proof:

We prove this lemma by using a non-deterministic procedure of computing extensions [Satoh, 1992] shown in Section 2.2.2.
Suppose that in **Step 1** of the procedure, usable defaults are selected in the sequence of $\frac{:\neg p_1}{\neg p_1}, \frac{:\neg p_2}{\neg p_2}, \ldots$. Let E_k be an extension of Δ computed at **Step 2** in the kth iteration of the procedure as follows:

$$E_k = Th\left(E_{k-1} \cup \left\{ \frac{: \neg p_k}{\neg p_k} \right\} \right).$$

Then it is obvious that E_k becomes also an extension of a default theory Δ_k as follows.

$$\Delta_k \stackrel{def}{=} \left(\bigcup_{i=1}^{k}\left\{\frac{:\neg p_i}{\neg p_i}\right\}, \Sigma\right) \qquad (where\ k \neq 0)$$

Now, corresponding to Δ_k, we define Δ'_k as follows:

$$\Delta'_k \stackrel{def}{=} \left(\bigcup_{i=1}^{k}\left\{\frac{:\neg p_i}{\neg p_i}\right\} \cup D, \phi\right)$$

and let E'_k be an extension of Δ'_k. Since **Step 0** corresponds to $k = 0$, i.e. 0th iteration, we define Δ'_0 as follows:

$$\Delta'_0 \stackrel{def}{=} (D, \phi).$$

and let E'_0 be an extension of Δ'_0.

In the following, based on the induction by numbers of the iteration of the procedure, we prove $E_{k+1} \cap Lit = E'_{k+1} \cap Lit$ at the $(k+1)$th iteration by assuming $E_k \cap Lit = E'_k \cap Lit$ at the kth iteration, which leads to the proof of $E \cap Lit = E' \cap Lit$.

1. In case of $i = 0$, according to **step 0** of the procedure, we obtain that

   $E_0 = Th(\Sigma)$.

   On the other hand, it holds that, $Th(\Sigma) \cap Lit = \mu Th(\Sigma) \cap Lit$.

   Then, for $\forall \ell_0 \in E_0 \cap Lit$, it holds that,

   $\ell_0 \in \mu Th(\Sigma)$, where ℓ_0 is a unit literal.

   Therefore, there exists a default rule in D of Δ'_0 whose form is $\frac{true:}{\ell_0}$.

   $\therefore\ \ell_0 \in E'_0 \cap Lit$,

   $\therefore\ E_0 \cap Lit \subseteq E'_0 \cap Lit$.

   In a similar way, we can also prove that $E'_0 \cap Lit \subseteq E_0 \cap Lit$.

   Thus it is shown that, $E'_0 \cap Lit = E_0 \cap Lit$.

2. In case of $i = k$,

   suppose that in the process of computing an extension E of Δ, usable defaults $\frac{:\neg p_i}{\neg p_i}$ $(i = 1, \ldots, k-1)$ have already been selected and at **Step 1** of the kth iteration, a usable default $\frac{:\neg p_k}{\neg p_k}$ is selected. Then E_k is obtained at the next **Step 2** as follows:

   $$\begin{aligned} E_k &= Th(E_{k-1} \cup \{\neg p_k\}) \\ &= Th(\Sigma \cup \{\neg p_1, \ldots, \neg p_k\}) \end{aligned} \qquad (1)$$

   Obviously E_k becomes an extension of Δ_k. Now according to the induction hypothesis, we assume that w.r.t. an extension E'_k of Δ'_k which is constructed corresponding to Δ_k, it holds that

$$E_k \cap Lit = E'_k \cap Lit \tag{2}$$

Then according to (1),(2), it holds that $\neg p_i \in E_k$ $(i = 1, \ldots, k)$, so does

$$\neg p_i \in E'_k \quad (i = 1, \ldots, k). \tag{3}$$

3. In case of $i = k + 1$,

   suppose that in the process of computing an extension E of Δ, a usable default $\frac{:\neg p_{k+1}}{\neg p_{k+1}}$ is selected at **Step 1** of the $(k+1)$th iteration, which means $p_{k+1} \notin E_k$. Thus according to (2), i.e. the induction hypothesis, it holds that,

$$p_{k+1} \notin E'_k. \tag{4}$$

   According to (4), since $\frac{:\neg p_{k+1}}{\neg p_{k+1}}$ becomes a usable default w.r.t.

$$\Delta'_{k+1} = \left(\bigcup_{i=1}^{k+1} \left\{ \frac{: \neg p_i}{\neg p_i} \right\} \cup D, \phi \right)$$

   at **Step 1** of the $(k+1)$th iteration of the procedure, the following result is obtained.

$$\neg p_{k+1} \in E'_{k+1} \tag{5}$$

   Thus, according to (3),(5), it holds that,

$$\neg p_i \in E'_{k+1} \quad (i = 1, \ldots, k+1). \tag{6}$$

   Now, for any ground literal ℓ such that

$$\ell \in E_{k+1} \cap Lit \quad \text{and} \quad \ell \notin E_k \cap Lit, \tag{7}$$

   it holds that as follows:

$$\Sigma \cup \{\neg p_1, \ldots, \neg p_k, \neg p_{k+1}\} \vdash \ell, \tag{8}$$
$$\Sigma \cup \{\neg p_1, \ldots, \neg p_k\} \nvdash \ell.$$

   (8) denotes that ℓ is inferred by using some clauses from Σ, say $C_i \in \Sigma$ $(i = 1, \ldots, m)$ and $\neg p_{j1}, \ldots, \neg p_{js}$, $\neg p_{k+1}$ where

$$\{\neg p_{j1}, \ldots, \neg p_{js}\} \subseteq \{\neg p_1, \ldots, \neg p_k\}. \quad (1 \leq s \leq k) \tag{9}$$

   Then it holds that

$$C_1, \ldots, C_m \vdash \neg p_{j1} \wedge \ldots \wedge \neg p_{js} \wedge \neg p_{k+1} \supset \ell.$$

   Therefore

$$\neg p_{j1} \wedge \ldots \wedge \neg p_{js} \wedge \neg p_{k+1} \supset \ell \in Th(\Sigma).$$

   Due to $\ell \notin \mu Th(\Sigma)$ and $p_{k+1} \notin \mu Th(\Sigma)$, there exists a clause $\tau \in \mu Th(\Sigma)$ such that includes both ℓ and p_{k+1} as follows.

$$\tau \stackrel{def}{=} \neg p'_{j1} \wedge \ldots\ldots \wedge \neg p'_{js} \wedge \neg p_{k+1} \supset \ell \in \mu Th(\Sigma)$$

where $\{p'_{j1}, \ldots, p'_{js}\} \subseteq \{p_{j1}, \ldots, p_{js}\}$ (10)

Therefore there exists a following default rule in D which is constructed from τ according to Definition 4.15.

$$\frac{\neg p'_{j1}, \ldots, \neg p'_{js}, \neg p_{k+1} :}{\ell} \in D \tag{11}$$

Since every rule in D is a delault rule of a default theory Δ'_{k+1} and $\neg p_{k+1}$ as well as $\neg p'_{jt}$ $(t = 1, \ldots, s)$ are members of E'_{k+1}, (11) derives ℓ as a member of the extension E'_{k+1} as follows:

$$\ell \in E'_{k+1}.$$

Therefore $$\ell \in E'_{k+1} \cap Lit \tag{12}$$

From (7),(12), it is proved that,

$$E_{k+1} \cap Lit \subseteq E'_{k+1} \cap Lit.$$

In a similar way, we can prove as follows.

$$E'_{k+1} \cap Lit \subseteq E_{k+1} \cap Lit,$$

Finally we can obtain the result as follows:

$$E_{k+1} \cap Lit = E'_{k+1} \cap Lit \qquad \square$$

Proof of Theorem 4.5 :

In case that T (i.e. Σ) is inconsistent, it is easily shown that $Circum(T; P; Z) \models G$ for any $G \in Lit$. On the other hand, there exists an integrity constraint $\leftarrow \in \Pi_\alpha$ because of $\mu Th(\Sigma) = \{\Box\}$. Therefore the answer set of Π^α has a unique answer set, Lit, which means that, for any $G \in Lit$, G is true in the unique answer set Lit of Π^α. Thus Theorem 4.5 is proved.

Next, we consider the case that T (i.e. Σ) is consistent as follows. According to Theorem 4.4, for any ground literal G of the language of T whose predicate symbol is not equality, it holds that,

$Circum(T; P; Z) \models G$

iff for any extension E of a default theory Δ, $E \models G$ (1)

where Δ is defined as follows :

$$\Delta \stackrel{def}{=} \left(\left\{ \frac{: \neg p(x)}{\neg p(x)}, \frac{: \neg q(x)}{\neg q(x)}, \frac{: q(x)}{q(x)} \middle| p \in P, q \in Q \right\}, T \right)$$

By the way, according to (4.1),

$$T \stackrel{def}{=} \Sigma \cup \mathrm{DCA} \cup \mathrm{UNA} \equiv \Sigma^{DCA} \cup \mathrm{DCA} \cup \mathrm{UNA}$$

where Σ^{DCA} is a set of ground clauses obtained by replacing all variables in each clause in Σ by every ground term of DCA.

Now, we define a default theory Δ_1 as follows:

$$\Delta_1 \stackrel{def}{=} \left(\left\{ \frac{: \neg p(x)}{\neg p(x)}, \frac{: \neg q(x)}{\neg q(x)}, \frac{: q(x)}{q(x)} \middle| p \in P, q \in Q \right\}, \Sigma^{DCA} \right)$$

Then we can easily prove that for any extension E of Δ, there exists an extension E_1 of Δ_1 which corresponds to E with a 1-1 correspondence such that

$$Th(E) = Th(E_1 \cup DCA \cup UNA) \tag{2}$$

and vice versa.

Next, let Δ_2 be a default theory defined as follows:

$$\Delta_2 \stackrel{def}{=} \left(\left\{ \frac{: \neg p(x)}{\neg p(x)}, \frac{: \neg q(x)}{\neg q(x)}, \frac{: q(x)}{q(x)} \middle| p \in P, q \in Q \right\} \cup D, \phi \right)$$

where D is a set of default rules each of which is generated from each characteristic clause in $\mu Th(\Sigma^{DCA})$ according to Definition 4.15.

Then according to Lemma 4.16, for any extension E_1 of Δ_1, there exists an extension E_2 of Δ_2 which corresponds to E with a 1-1 correspondence such that

$$E_1 \cap Lit = E_2 \cap Lit \tag{3}$$

and vice versa.

According to Theorem 2.26 [Gelfond and Lifschitz, 1991, Proposition 3], for any extension E_2 of a default theory Δ_2, there exists an answer set S of ELP Π^α with a 1-1 correspondence such that

$$E_2 \cap Lit = S \tag{4}$$

and vice versa.

Therefore, based on the results of (1)$\sim$ (4), for any ground literal G of the language of T whose predicate symbol is not equality, it holds that

$Circum(\widetilde{\forall} T; P; Z) \models G$

iff for any extension E of a default theory Δ, $G \in E$ (according to (1))

iff for any extension E_1 of a default theory Δ_1, $G \in E_1 \cup DCA \cup UNA$ (according to (2))

iff for any extension E_1 of a default theory Δ_1, $G \in E_1$ (because the predicate symbol of G is not equality.)

iff for any extension E_2 of a default theory Δ_2, $G \in E_2$ (according to (3))

iff for any answer set S of ELP Π^α, $G \in S$ (according to (4))

□

In the following, we firstly prepare some definitions and lemmas, and then prove Theorem 4.13.

Definition 4.17 Let Π be an extended disjunctive program. We write $AS(\Pi)$ as a set of all answer sets of Π. We say an answer set of Π is *consistent* if every atom p and its negation $\neg p$, both is not simultaneously in the answer set. For any ground formula ϕ, we write $\Pi \models \phi$ iff ϕ is true in every answer set of Π.

Theorem 4.18 [Sakama and Inoue, 1995] *Let Π^α be an extended disjunctive program constructed from $Circum(\widetilde{\forall}\, T; P; Z)$ by using definition 4.11 and F be a ground formula. Then it holds that*

$$Circum(\widetilde{\forall} T; P; Z) \models F \quad \textit{iff} \quad \Pi^\alpha \models F$$

Remarks. It is proved that there is a 1-1 correspondence between the *Herbrand* models of $Circum(\widetilde{\forall} T; P; Z)$ and the answer sets of the translated EDP Π^α where DCA and UNA are incorporated in T. That is, S is a PZ-minimal model of $\widetilde{\forall}\, T$ iff S is an answer set of the translated EDP Π^α. The positive and negative literals contained explicitly in an answer set S of EDP Π^α represent elements which are true and false respectively in the corresponding PZ-minimal model S (see [Sakama and Inoue, 1995]).

Lemma 4.19 *Suppose that $\widetilde{\forall}\, T$ is consistent. Let $(\Pi^\alpha)_i$ be an EDP defined by Definition 4.11, and $(\Pi_\gamma)_i$, $(\Pi^{\alpha\gamma})_i$ and $(\Pi^\beta)_i$ be sets defined as follows.*

$(\Pi_\gamma)_i \stackrel{def}{=} \{L \leftarrow L_i \mid L \leftarrow L_i \in \Pi_\gamma$ *and L_i is the i-th name changed literal.*$\}$.

$(\Pi^{\alpha\gamma})_i \stackrel{def}{=} ((\Pi^\alpha)_i \cup (\Pi_\gamma)_i)$

$(\Pi^\beta)_i \stackrel{def}{=} \bigcup_{j=1}^{i}((\Pi^{\alpha\gamma})_j)$

Then it holds that

$$AS((\Pi^\beta)_i) = \{S_i^{\alpha\gamma} \cup S_{i-1}^\beta \mid S_i^{\alpha\gamma} \cup S_{i-1}^\beta \text{ is consistent where } S_i^{\alpha\gamma} \in AS((\Pi^{\alpha\gamma})_i) \text{ and } S_{i-1}^\beta \in AS((\Pi^\beta)_{i-1})\}.$$

Lemma 4.20 *Suppose that $\widetilde{\forall} T$ is consistent. Let Φ, ϕ_1 and ϕ_2 be ground formulas of the language of T. $(\Pi^\beta)_i \models \phi$ iff there exists ϕ_1 and ϕ_2 such that $(\Pi^{\alpha\gamma})_i \models \phi_1$ and $(\Pi^\beta)_{i-1} \models \phi_2$ and $\phi_1 \wedge \phi_2 \models \phi$.*

Proof:

$\Rightarrow$) According to Lemma 4.19, $(\Pi^\beta)_i \models \phi$ means that for every consistent $S_i^{\alpha\gamma} \cup S_{i-1}^\beta$, $S_i^{\alpha\gamma} \cup S_{i-1}^\beta \models \phi$. Even if $S_i^{\alpha\gamma} \cup S_{i-1}^\beta$ is not consistent, obviously $S_i^{\alpha\gamma} \cup S_{i-1}^\beta \models \phi$. Therefore it holds that for every $S_i^{\alpha\gamma} \cup S_{i-1}^\beta$ w.r.t. any pair of $S_i^{\alpha\gamma} \in AS((\Pi^{\alpha\gamma})_i)$ and $S_{i-1}^\beta \in AS((\Pi^\beta)_{i-1})$, $S_i^{\alpha\gamma} \cup S_{i-1}^\beta \models \phi$. (1)

So, let $S_i^{\alpha\gamma 1}, ..., S_i^{\alpha\gamma n}$ be answer sets from $AS((\Pi^{\alpha\gamma})_i)$, and let each of $\sigma_i^{\alpha\gamma 1}, ..., \sigma_i^{\alpha\gamma n}$ be the conjunction of ground literals of the language of T occurring in $S_i^{\alpha\gamma j}(1 \leq j \leq n)$. That is, each $\sigma_i^{\alpha\gamma j}(1 \leq j \leq n)$ is the conjunction of elements contained in a Herbrand model of T. In a similar way, let $S_{i-1}^{\beta 1}, ..., S_{i-1}^{\beta m}$ be answer sets from $AS((\Pi_\beta)^{i-1})$, and let each of $\sigma_{i-1}^{\beta 1}, \ldots, \sigma_{i-1}^{\beta m}$ be the conjunction of ground literals of the language of T occurring in $S_{i-1}^{\beta 1}, ..., S_{i-1}^{\beta m}$ respectively. Then each $\sigma_{i-1}^{\beta j}(1 \leq j \leq m)$ is also the conjunction of elements contained in a Herbrand model of T.

Since ϕ is a formula of the language of T, (1) denotes that for any j and k,

$$\sigma_i^{\alpha\gamma j} \wedge \sigma_{i-1}^{\beta k} \models \phi. \quad (2)$$

In (2), if $\sigma_i^{\alpha\gamma j} \wedge \sigma_{i-1}^{\beta k}$ is consistent, $\sigma_i^{\alpha\gamma j}$ and $\sigma_{i-1}^{\beta k}$ should be equivalent each other since each of them represents the Herbrand model of T.

Now let us define ϕ_1 and ϕ_2 as follows.

$$\phi_1 \stackrel{def}{=} \sigma_i^{\alpha\gamma 1} \vee \ldots \vee \sigma_i^{\alpha\gamma n},$$
$$\phi_2 \stackrel{def}{=} \sigma_{i-1}^{\beta 1} \vee \ldots \vee \sigma_{i-1}^{\beta m},$$

Then it holds that $(\Pi^{\alpha\gamma})_i \models \phi_1$ and $(\Pi^{\beta})_{i-1} \models \phi_2$ and $\phi_1 \wedge \phi_2 \models \phi$.
$\Leftarrow$) By Lemma 4.19, $(\Pi_\beta)^i \models \phi_1$ and $(\Pi_\beta)^i \models \phi_2$. Therefore, $(\Pi_\beta)^i \models \phi$. □

Definition 4.21 We define $Circum_i$ and $ACircum_i$ as follows:

$$Circum_i \stackrel{def}{=} Circum(\widetilde{\forall} T; P^i; P^{i+1}, ..., P^k; Z),$$
$$ACircum_i \stackrel{def}{=} \bigwedge_{j=1}^{i} Circum_j.$$

Then according to Theorem 2.10, it holds that

$$Circum(\widetilde{\forall} T; P^1 > P^2 > \ldots > P^k; Z) \equiv ACircum_k$$

Theorem 4.13 *Let F be a ground formula of the language of T and equality does not occur as a predicate symbol in F. Then, it holds that, for any F and every $i = 1, \ldots, k$,*

$$ACircum_i \models F \quad \textit{iff} \quad (\Pi^{\beta})_i \models F$$

Proof:

In case that T is inconsistent, it is easily proved that this theorem holds.
So, when T is consistent, we prove the theorem by the induction of level of $Circum_i$ as follows.

1. In case of **i=1**, according to Theorem 4.18,

   $$ACircum_1 \models F \quad \text{iff} \quad (\Pi^{\beta})_1 \models F$$

2. By assuming that this Theorem holds in case of **i=j-1**, we show that this Theorem also holds in case of **i=j**. According to Lemma 4.20, it holds that

   $(\Pi^{\beta})_j \models F$ iff there exist F_1, F_2 such that $(\Pi^{\alpha\gamma})_j \models F_1$ and $(\Pi^{\beta})_{j-1} \models F_2$ and $F_1 \wedge F_2 \models F$. (1)

   According to Theorem 4.18, it holds that

   $$(\Pi^{\alpha\gamma})_j \models F_1 \quad \text{iff} \quad Circum_j \models F_1. \quad (2)$$

   By induction hypothesis,

   $$(\Pi^{\beta})_{j-1} \models F_2 \quad \text{iff} \quad ACircum_{j-1} \models F_2. \quad (3)$$

   Then according to $ACircum_j = ACircum_{j-1} \wedge Circum_j$ and (1), (2), (3), this theorem is proved. □

Chapter 5

Reasoning about Dynamic Preferences

5.1 Introduction

The needs to handle dynamic preferences in the frameworks of the non-monotonic reasoning are remarkably well-recognized recently ([Brewka, 1996], [Delgrande, 1997], [Junker, 1997]).

Preferences are the metalevel information about which rule should be preferably used among involved conflicting rules in the commonsense knowledge and its assessment plays a crucial role in the reasoning. However, in the conventional frameworks of non-monotonic reasoning, not only the preference information itself cannot be expressed in the logical language within their frameworks, but also it is impossible to handle *dynamic preferences*, that is, to derive an object-level goal with reasoning about preferences.

For example, in prioritized circumscription, information about "what should be the minimized predicates, their priorities and variable predicates?", is pre-specified by the user. In addition, priorities among minimized predicates are also specified in an external manner such as the metalevel reasoning of a human being who takes into account various kinds of preferences in each particular application. Priorities specified in prioritized circumscription are called *static preferences* because priorities are not reasoned about in the framework of prioritized circumscription.

The following legal reasoning example [Gorden, 1993] illustrates that the *dynamic preferences* are quite natural in human commonsense reasoning.

Assume a person wants to find out if her security interest in a certain ship is perfected. She currently has possession of the ship. According to the Uniform Commercial Code(UCC) which is a state law, a security interest in goods may be perfected by taking possession of the collateral. However, there is a federal law called the Ship Mortgage Act(SMA) according to which a security interest in a ship may only be perfected by filing a financing statement. Such a statement has not been filed.

This can be represented by the following axiom set A where $ab1$ and $ab2$ are

abnormal predicates[Lifschitz, 1985]:

$$\begin{aligned} A: \quad & possession \wedge \neg ab1 \supset perfected, && (UCC)\\ & ship \wedge \neg fin{-}statement \wedge \neg ab2 \supset \neg perfected, && (SMA)\\ & possession,\\ & ship,\\ & \neg fin{-}statement. \end{aligned}$$

Now the question is whether the UCC or the SMA takes precedence in this case, that is, in other words: whether $ab1$ or $ab2$ has the higher priority for minimization in circumscription. There are two known legal principles for resolving conflicts of this kind.

The principle of Lex Posterior gives precedence newer laws. In our case, UCC is newer than the SMA. On the other hand, the principle of Lex Superior gives precedence to laws supported by the higher authority. In our case the SMA has higher authority since it is federal law.

The above meta-knowledge about preferences between laws are represented by the following axiom set M where $y \prec x$ means that a rule x is preferable to y and $LP(x, y)$ and $LS(x, y)$ are rule names:

$$\begin{aligned} M: \quad & more{-}recent(x, y) \supset y \prec x, && (LP(x, y))\\ & fed{-}law(x) \wedge state{-}law(y) \supset y \prec x, && (LS(x, y))\\ & more - recent(UCC, SMA),\\ & fed - law(SMA),\\ & state - law(UCC). \end{aligned}$$

When formalizing this example by using circumscription, firstly, a human being may do the meta-reasoning from M and obtains the following result:

$$M \vdash SMA \prec UCC, \qquad M \vdash UCC \prec SMA$$

Therefore, he (she) decides that there is no precedence between SMA and UCC, that is, there is no priorities between $ab1$ and $ab2$, which leads to parallel circumscription as follows:

$$Circum(\widetilde{\forall} A; ab1, ab2; perfected),$$

where neither $perfected$ nor $\neg perfected$ is proved.

Now, assume the following information is added to M:

$$LP(SMA, UCC) \prec LS(UCC, SMA)$$

Since $UCC \prec SMA$ and $SMA \prec UCC$ are derived by $LS(UCC, SMA)$ and $LP(SMA, UCC)$ respectively, he (she) decides that $UCC \prec SMA$ is preferable to

$SMA \prec UCC$, and the priority of $ab2$ should be assigned higher than $ab1$, which leads to prioritized circumscription as follows:

$$Circum(\tilde{\forall} A; ab2 > ab1; perfected) \models \neg perfected$$

The aim of the research in this chapter is to integrate consistently such a human metalevel reasoning about dynamic preferences and the object-level non-monotonic reasoning formalized by circumscription. As is shown in this example, the non-monotonic reasoning which can treat dynamic preferences correctly is inevitably required in case of applying three well-known general principles Lex Superior, Lex Specialis and Lex Posterior in the legal domain.

As a previous approach, Brewka [Brewka, 1996] presented a method to reason dynamic preferences which is based on the modified well-founded semantics for extended logic programs.

In this chapter, we present a method of handling some class of dynamic preferences in the framework of circumscription. This enables us to compute consistently its metalevel and object-level reasoning by expressing them in an extended logic program, whose declarative meaning is given by the answer set semantics. In order to reason the priorities, we makes use of the policy axioms whose meaning is similar (but not equivalent) to Lifschitz's policy axioms [Lifschitz, 1987a] which permit to describe circumscription policy by axioms. By using our policy axioms, domain-independent inference rules from preferences to priorities are provided. Thus priorities can be derived from the preferences dynamically, which triggers to compute the object-level circumscriptive theory by logic programming based on Wakaki and Satoh's method [Wakaki and Satoh, 1997] of compiling circumscription into extended logic programs.

5.2 Reasoning about Dynamic Preferences

We consider the reasoning formalized by a circumscriptive theory (A, C) [Gelfond and Lifschitz, 1988a] (see Section 2.2.1) with a set M where A is a sentence represented by a set of clauses, C is the circumscription policy and M is a set of preference informations. We suppose that A and M are given as well as a tuple P of minimized predicates, a tuple Z of variable predicates and a tuple Q of fixed predicates are already specified as the circumscription policy C, but priorities among predicates in P, which are defined by a set S_{pri} in the subsequent section, is unknown and is required to be determined based on the reasoning from preferences in M.

5.2.1 Meta Level Reasoning from Preferences to Priorities

In this thesis, the similar notions, i.e. priorities and preferences are introduced. We use "priorities" as the priorities expressed in prioritized circumscription which formalizes the object-level reasoning. On the other hand, there exist mBR96any

kinds of preferences which are not only between the object-level rules but also between the metalevel rules, i.e. metarules in the application domain. The preferences about the object-level rules can be easily corresponded to priorities in prioritized circumscription, but the correspondence between priorities and preferences about metarules is not clear though a human being does it based on his metalevel reasoning. In this section, first we show how to represent preferences in M and then model the human's metalevel reasoning from various preference information to priorities which is described by a normal logic program:$\Pi \cup \Gamma$ as follows.

[1] Representation of Priorities

There are two ways to represent relative priorities between minimized predicates. One is the priority relation used to represent prioritized circumscription and another is Lifschitz's policy axiom [Lifschitz, 1987a] between minimized predicates. Prioritized circumscription:

$$Circum(\widetilde{\forall} A; P^1 > P^2 > \ldots > P^k; Z).$$

denotes the following priority informations:

$$p_i > p_j, \quad for\ 1 \leq i < j \leq k \quad \forall p_i \in P^i,\ \forall p_j \in P^j$$

which expresses the priority relation $>$ between p_i and p_j as a strict partial order, i.e. an irreflexive and transitive relation. According to the semantics of prioritized circumscription, $p_i > p_j$ means that minimizing p_i is prior to minimizing p_j.

According to Lifschitz's policy axiom, the specification that p is a minimized predicate is given by $V[p : p]$.

In the process of minimizing both p_1 and p_2 ($p_1 \neq p_2$), assigning a higher priority of the minimizing order to p_1 than to p_2 are represented by $V[p_1 : p_2]$. Then the relation V represented by the policy axiom $V[p_1 : p_2]$ is a pre-order with respect to minimized predicates p_1 and p_2, that is, a reflexive and transitive relation.

In this chapter, we also use policy axioms $V[p_1 : p_2]$ which have the above properties about minimized predicates (i.e. pre-order). Strictly speaking, however, our policy axioms are not equivalent to Lifschitz's policy axiom under the semantics of circumscription. The reason is that as the minimality condition for minimized predicates, we take into account either parallel circumscription or prioritized circumscription while he postulates the pointwise approach of circumscription based on his policy axioms. We only use the policy axiom in order to express the minimizing order among minimized predicates by a pre-order relation. If there exist both $V[p_1 : p_2]$ and $V[p_2 : p_1]$, we adopt such a circumscription policy that there is no priority between them and both p_1 and p_2 should be minimized in parallel.

Example 5.1

The circumscription policy in the regal example in Section 5.1 is given as follows:

$$P = \{ab1, ab2\},\ Z = \{perfected\},$$
$$Q = \{possession,\ ship,\ \neg fin{-}statement\}$$

The following policy axioms exist corresponding to all predicates in P.

$$V[ab1 : ab1], \quad V[ab2 : ab2],$$

But the priority relation $>$ between $ab1$ and $ab2$ are unknown and required to be determined whether $ab1 > ab2$ or $ab2 > ab1$ or neither of them (i.e. $ab1, ab2$ which means no priorities between $ab1$ and $ab2$). For example, $ab1 > ab2$ is expressed by $V[ab1 : ab2]$ as well as no existence of $V[ab2 : ab1]$ since the relation V is a pre-order while the priority relation $>$ is a strict partial order.

[2] Representation of Preferences

A human being does not consider the conflict between preferences as a contradiction, but only consider there is no precedence between them, which does not mean contradiction. So, we represent the preference information by a pre-order explicitly though Brewka[Brewka, 1994] expresses the preference by a strict partial order. We use a special (infix) symbol $\prec$ that can take rule names as arguments to represent preferences among rules and has a simple mathematical property, that is, pre-order (i.e., a reflexive and transitive relation) on the class of all rule names. Intuitively, $n_1 \prec n_2$ where n_1 and n_2 are rule names means the rule with name n_2 is more or equally preferable to n_1. In practical application, special preference informations are often referred by unique names, e.g. $Lex\ Superior, Lex\ Posterior$. So, we introduce a 3-ary predicate symbol $\prec$ whose three arguments are rule names. $\prec (n_1, n_2, n_3)$ denotes that a preference information $n_1 \prec n_2$ is derived due to a rule n_3.

There exist sometimes both $\prec (n_1, n_2, n_3)$ and $\prec (n_2, n_1, n_4)$, or equivalently $(n_1 \prec n_2) \wedge (n_2 \prec n_1)$. This means that there is no precedence between n_1 and n_2 (i.e. n_1 and n_2 have the equal preference each other) due to the conflict between n_3 and n_4.

Example 5.2

Preference Information of a legal example in Section 5.1 can be represented by using our notation as follows:

Since the Uniform Commercial Code (UCC) is newer than the Ship Mortgage Act (SMA), the principle of Lex Posterior gives a following preference:

$$SMA \prec UCC$$

Since this Lex Posterior is named by a rule name $LP(SMA, UCC)$, the above preference is expressed by

$$\prec (SMA, UCC, LP(SMA, UCC))$$

where the rule name is explicitly represented along with the preference.

[3] Inference from Preferences to Priorities

Inference from preferences to priorities are represented by a normal logic program as follows.

A policy axiom $V[p1 : p2]$ is represented by a ground literal $V(p1, p2)$ in which V is a predicate symbol and $p1$ and $p2$ are the individual constants corresponding to minimized predicates $p1$ and $p2$ of the object-level circumscription.

Hereafter $>$ denotes a binary (infix) predicate symbol and a priority relation is represented by a ground literal: $p1 > p2$ which we call a *priority axiom*.

We suppose that at most one minimized predicate occurs in each clause from A and each minimized predicate does not occur multiple clauses. When $p1, p2 \in P$ occur in different clauses (rules) $r1, r2$ of a set A of clauses respectively, assigning the higher priority to $p1$ than to $p2$ can be said in such other words that a rule $r1$ takes precedence over a rule $r2$. Therefore according to this 1-1 correspondence between an ordered pair of $p1$, $p2$ and an ordered pair of $r1$, $r2$, we allow the predicate $>$ as well as V to have rule names as its arguments like $r1 > r2$ as well as $V(r1, r2)$.

The correspondence between $r1 > r2$ and $p1 > p2$ intervenes between the metalevel and the object-level reasoning, which is described as follows:

$$p1 > p2 \leftarrow \ r1 > r2. \qquad (\forall p1 \in P,\ \forall p2 \in P)$$

This is the domain-dependent information.

By allowing $>$ as well as V to have rule names as its arguments, the precedence between meta-rules can also be expressed by using the predicate symbols $>$ as well as V.

Definition 5.3 Meta-knowledge to reason from preference to priorities are represented by

$$\Pi \cup \Gamma,$$

where Π is a set of the domain-independent inference rules from preferences to priorities as follows and Γ is a set of the domain-dependent rules which describe various preference information of M in particular application domain by means of $\prec$. We suppose that Γ is a stratified logic program. Thus both Π and Γ are normal logic programs, but more strictly speaking, they are stratified logic programs. Every rule containing variables stands for all its ground instances obtained by replacing each variable by every ground term of the rule name and all rule names are supposed to be finite.

Π :

(1) $x > y \leftarrow \ V(x, y), not\ V(y, x).$

(2) $V(x, z) \leftarrow \ V(x, y), V(y, z).$

(3) $V(x, y) \leftarrow \ y \prec x, not\ x \prec y.$

(4) $V(x,y) \leftarrow \prec(x,y,z_1), \prec(y,x,z_2), z_1 \prec z_2, not\ z_2 \prec z_1.$

(5) $\prec(x,y,u) \leftarrow \prec(x,y,z_1), \prec(y,x,z_2), \prec(z_2,z_1,u).$

(6) $\prec(x,z,Tran(u,v)) \leftarrow \prec(x,y,u), \prec(y,z,v).$

(7) $x \prec y \leftarrow \prec(x,y,u).$

where *not* denotes the negation-as-failure operator, $Tran$ is a function symbol and *dom* is a unary predicate symbol. $Tran(u,v)$ is a name of a rule (6). Every rule name r_i should be described by a ground atom such as $dom(r_i)$ in Γ.

A rule (1) shows how to obtain the priority relation $>$ between a rule x and a rule y which is a strict partial order from the policy axiom $V(x,y)$ which is a pre-order. The rule (2) expresses a transitive relation as one of properties of policy axioms. $V(x,x)$ expressing a reflexive relation also holds, but it is omitted because it is redundant to find out priorities. The rule (3) means that if there is a fact that a rule x is preferable to a rule y and there is no evidence to the contrary, then x takes the higher precedence or equal to y (or equivalently, a minimized predicate of x has the higher or equal priority than that of y). The rule (4) denotes that if there exist both $x \prec y$ and $y \prec x$ due to the conflict of two meta rules z_1 and z_2 respectively , but z_2 is preferable to z_1 as well as there is no evidence to the contrary, then x takes the higher or equal precedence to y. A rule (5) gives a kind of the transitive relation about $\prec$ such that, if there exist both $x \prec y$ and $y \prec x$ due to the conflict of meta rules z_1 and z_2, a preference between x and y due to the meta-meta rule u is transitively derived from the corresponding preference between meta rules of z_1 and z_2 due to the meta-meta rule u. A rule (6) expresses a property of pre-order $\prec$, i.e. a transitive relation.

Lemma 5.4 *A normal logic program $\Pi \cup \Gamma$ has a unique stable model since it is a stratified logic program.*

Definition 5.5 *Let π be a normal logic program and L be a ground literal. Then $\pi \models_{sm} L$ denotes that L is true in some stable model of π.*

Definition 5.6 Let M be a set of preference information. Then, let $\Pi \cup \Gamma$ be a normal logic program constructed from M according to Definition 5.3 and N be a unique stable model of $\Pi \cup \Gamma$. Then, a set S_{pri} is defined as a set of priority atoms each of which represents a priority relation between the minimized predicates in P as follows:

$$S_{pri} \stackrel{def}{=} N \cap Pri$$

where $Pri \stackrel{def}{=} \{p_i > p_j |\ \forall (p_i, p_j) \in P \times P,\ p_i \neq\ p_j\}$.

Therefore, w.r.t. minimized predicates p_i, p_j, it holds that,

$$p_i > p_j \in S_{pri} \quad \text{iff} \quad \Pi \cup \Gamma \models_{sm} p_i > p_j$$

Note: Strictly speaking, when $p_i > p_j$ is a priority atom representing a priority relation, symbols of both p_i and p_j should be used by newly introduced individual constants which represent the corresponding predicate symbols: p_i, p_j.

Example 5.7
Let us consider an legal example in Section 5.1 again.
Since all clauses in M are domain-dependent, Γ is described by a normal logic program as follows:

$$\begin{aligned}
\Gamma: \quad & \prec(y, x, LP(x, y)) \leftarrow \ more{-}recent(x, y), \\
& \prec(y, x, LS(x, y)) \leftarrow \ fed{-}law(x), state{-}law(y), \\
& \quad more{-}recent(UCC, SMA) \leftarrow, \\
& \quad fed{-}law(SMA) \leftarrow, \\
& \quad state{-}law(UCC) \leftarrow, \\
& \quad dom(UCC) \leftarrow, \\
& \quad dom(SMA) \leftarrow, \\
& \quad dom(LP(x, y)) \leftarrow, \\
& \quad dom(LS(x, y)) \leftarrow, \\
& \quad ab1 > ab2 \leftarrow \ UCC > SMA, \\
& \quad ab2 > ab1 \leftarrow \ SMA > UCC.
\end{aligned}$$

In practice, a logic programming interpreter for stable model semantics has the difficulty to handle function symbols such as $Tran$, LP, LS. So, for example, rule names: $LP(x, y), LS(x, y)$ occurring in Γ should be generally replaced by the ground rule names without function symbols such as $LP1$, $LS1$ instead of $LP(SMA, UCC)$, $LS(UCC, SMA)$ as follows:

$$\begin{aligned}
& \prec(SMA, UCC, LP1) \leftarrow \\
& \qquad more{-}recent(UCC, SMA), \\
& \prec(UCC, SMA, LS1) \leftarrow \\
& \qquad fed{-}law(SMA), state{-}law(UCC), \\
& dom(LP1) \leftarrow , \\
& dom(LS2) \leftarrow .
\end{aligned}$$

Hereafter, we suppose that all rule names occurring in $\Pi \cup \Gamma$ are replaced by ground rule names without function symbols.

Let N_0 be a stable model of $\Pi \cup \Gamma$ given above. Then according to Definition 5.6, the following result is derived:

$$S_{pri} \stackrel{def}{=} N_0 \cap \{ab1 > ab2, \ ab2 > ab1\} = \phi, \qquad (1)$$

where ϕ is an empty set. This means that there is no priority between $ab1$ and $ab2$, which leads to parallel circumscription: $Circum(\widetilde{\forall} A; ab1, ab2; perfected)$.

Next, suppose that a new preference information $\prec (LP1, LS1, Lex1)$ whose rule name is $Lex1$ is added to Γ as follows:

$$\Gamma_1 \stackrel{def}{=} \Gamma \cup \{\prec (LP1, LS1, Lex1)\}$$

where N_1 is a stable model of $\Pi \cup \Gamma_1$. Then the following result is derived according to Definition 5.6:

$$S_{pri} \stackrel{def}{=} N_1 \cap \{ab1 > ab2,\ ab2 > ab1\} = \{ab2 > ab1\},$$

which denotes a priority $ab2 > ab1$ and leads to prioritized circumscription as follows:

$$Circum(\widetilde{\forall} A; ab2 > ab1; perfected) \models \neg perfected.$$

Furthermore, let us suppose that meta-meta preferences are added to Γ_1 as follows:

$$\Gamma_2 \stackrel{def}{=} \Gamma_1 \cup \{\prec (LS1, LP1, Lex2)\} \cup \{\prec (Lex1, Lex2, Lexs)\},$$

and N_2 be a stable model of $\Pi \cup \Gamma_2$. Then the following priority is derived according to Definition 5.6,

$$S_{pri} \stackrel{def}{=} N_2 \cap \{ab1 > ab2,\ ab2 > ab1\} = \{ab1 > ab2\},$$

where a inference rule (5) plays a significant role for the derivation of this priority. As a result, we can obtain again a reasonable result under Γ_2:

$$Circum(\widetilde{\forall} A; ab1 > ab2; perfected) \models perfected.$$

5.2.2 Integrating Meta level Reasoning with the Framework of Circumscription

In the following, it is shown that metalevel reasoning from preferences to priorities presented in Section 5.2.1 and the object-level circumscription can be integrated consistently based on Theorem 4.5 and Theorem 4.6, i.e. Wakaki and Satoh's method [Wakaki and Satoh, 1997].

In prioritized circumscription:

$$Circum(\widetilde{\forall} A; P^1 > P^2 > \ldots > P^k; Z),$$

a tuple P of n minimized predicates is broken into disjoint parts $P^1, \ldots, P^k$ with assigning a higher priority to the members of P^i than to the members of P^j for $i < j$ where for any i $(1 \leq i \leq k)$,

$$P^i \stackrel{def}{=} \left[P_{i_1}, \ldots, P_{i_{n_i}}\right] \qquad (\sum_i n_i = n)$$

and P_{i_j} $(1 \leq j \leq n_i)$ is a predicate constant from P. The special case of $k = 1$ corresponds to parallel circumscription: $Circum(\widetilde{\forall} A; P; Z)$. Therefore every different division $\left[P^1, \ldots, P^k\right]$ of a given P can specify every corresponding prioritized circumscription including parallel circumscription.

Definition 5.8 *Prioritized circumscription :*

$$Circum(\widetilde{\forall}A; P^1 > P^2 > \ldots > P^k; Z) \qquad (1 \leq k \leq n)$$

including parallel circumscription in case of $k = 1$ *is given where* $\left[P^1, \ldots, P^k\right]$ *represents a division of a tuple* P *of* n *minimized predicates. We define a 1-1 correspondence between* $\left[P^1, \ldots, P^k\right]$ *and a following set* $S \subset Pri$ *such that*

- *includes the priority axioms as follows:*

  $P_{i\ell} > P_{j_m}$ *for any* i, j, ℓ, m
  $(1 \leq i < j \leq k,\ 1 \leq \ell \leq n_i,\ 1 \leq m \leq n_j)$

- *and does not includes the priority axioms as follows:*

  $P_{i\ell} > P_{i_m}$ *for any* i, ℓ, m
  $(1 \leq i \leq k,\ 1 \leq \ell, m \leq n_i,\ \ell \neq m)$

and vice versa. Then with respect to this 1-1 correspondence, we define a function md *such that* $md(S) = \left[P^1, \ldots, P^k\right]$.

In the following definition, firstly a set of extended logic programs *ELP* Π^β (or *ELP* Π^α) corresponding to all possible prioritized circumscription is generated according to the method in [Wakaki and Satoh, 1997] and then conditions about priority relations are introduced to each rule in every generated *ELP* Π^β (or *ELP* Π^α).

Definition 5.9 *Let* A *be a set of clauses and* P, Z *be tuples of minimized and variable predicates occurring in* A*. Then* $\mathcal{E}$ *is an extended logic program which is defined as follows:*

1. *Generate all possible division* $\left[P^1, \ldots, P^k\right]$ *of a given* P.

2. *For each* $\left[P^1, \ldots, P^k\right]$*, do from (a) to (c),*

   (a) *Let* $Circum(\widetilde{\forall}A; P^1 > \ldots > P^k; Z)$
   be compiled into an extended logic program: ELP Π^β *according to Definition 4.3. If* $k = 1$*, compile* $Circum(\widetilde{\forall}A; P; Z)$ *into* ELP Π^α *according to Definition 4.2.*

   (b) *For every rule of* Π^β *(or* Π^α*) obtained in (a) such as*

   $$L \leftarrow L_1, \ldots, L_m, notL_{m+1}, \ldots, notL_n,$$

   add all priority atoms in S *such that* $md(S) = \left[P^1, \ldots, P^k\right]$ *to its body according to Definition 5.8 as follows:*

   $$L \leftarrow \bigwedge_{i,j,\ell.m} (P_{i\ell} > P_{j_m}) \wedge \bigwedge_{i,\ell,m} not\ (P_{i\ell} > P_{i_m})$$

$$\wedge L_1 \wedge \ldots, L_m \wedge not L_{m+1} \wedge \ldots \wedge not L_n,$$

where a symbol "$\wedge$" is used instead of a symbol "," for the notational clarity.

(c) Let $\mathcal{E}(\left[P^1, \ldots, P^k\right])$ *be a set of all rules generated in (b).*

3. $\mathcal{E}$ *is a union of all* $\mathcal{E}(\left[P^1, \ldots, P^k\right])$.

A set S_{pri} defined in Definition 5.6 does not always forms some division $\left[P^1, \ldots, P^k\right]$ of a tuple P suitable for prioritized circumscription defined in Definition 5.8 such that $md(S_{pri}) = \left[P^1, \ldots, P^k\right]$. However, if the set forms such a division, the following theorem holds.

Theorem 5.10 *A set M of preference information, a set A of clauses and tuples P, Z of n minimized predicates and variable predicates occurring in A respectively are given. Then let $\Pi \cup \Gamma$ be constructed from M according to Definition 5.3, $\mathcal{E}$ be constructed from A,P,Z according to Definition 5.9 and S_{Pri} be a set of priority atoms derived from given M according to Definition 5.6. Suppose that there is a 1-1 correspondence between S_{pri} and some division $\left[P^1, \ldots, P^k\right]$ $(1 \leq k \leq n)$ of a given P such that $md(S_{pri}) = \left[P^1, \ldots, P^k\right]$ according to Definition 5.8. Then for any ground literal G of the language of A whose predicate symbol is not equality, it holds that,*

$Circum(\widetilde{\forall} A; P^1 > P^2 > \ldots > P^k; Z) \models G$
iff G *is true in all answer sets of* $\Pi \cup \Gamma \cup \mathcal{E}$
where $\Pi \cup \Gamma \cup \mathcal{E}$ *is an extended logic program.*

Proof: See Appendix.

Example 5.11

Let us consider a legal example in Section 5.1 again.

Given $P \stackrel{def}{=} \{ab1, ab2\}$, all different divisions of P are represented as follows:

$$[[ab1], [ab2]], \quad [[ab2], [ab1]], \quad [[ab2, ab1]]$$

each of which corresponds to the following circumscription respectively:

$$Circum(\widetilde{\forall} A; ab1 > ab2; Z),$$
$$Circum(\widetilde{\forall} A; ab2 > ab1; Z),$$
$$Circum(\widetilde{\forall} A; ab1, ab2; Z).$$

Then an extended logic program $\mathcal{E}$ is obtained as follows:

$$\mathcal{E} \stackrel{def}{=} \mathcal{E}([[ab1], [ab2]]) \cup \mathcal{E}([[ab2], [ab1]]) \cup \mathcal{E}([[ab1, ab2]]).$$

Here by using Definition 4.2 and Definition 4.3 of compiling parallel circumscription and prioritized circumscription into ELP Π^{α} and Π^{β} respectively, $\mathcal{E}([[ab1],[ab2]])$ is constructed by compiling $Circum(\widetilde{\forall}A; ab1 > ab2; Z)$,
into ELP Π^{β} and adding a literal: $ab1 > ab2$ to the body of each rule of the Π^{β},
$\mathcal{E}([[ab2],[ab1]])$ is constructed by compiling $Circum(\widetilde{\forall}A; ab2 > ab1; Z)$,
into ELP Π^{β} and adding a literal: $ab2 > ab1$ to the body of each rule of the Π^{β}
and $\mathcal{E}([[ab1, ab2]])$ are constructed by compiling $Circum(\widetilde{\forall}A; ab1, ab2; Z)$,
into ELP Π^{α} and adding literals: both $not\ ab1 > ab2$ and $not\ ab2 > ab1$ to the body of each rule of the Π^{α}.

As a result, an extended logic program: $\Pi \cup \Gamma \cup \mathcal{E}$ has two answer sets where $perfected$ is true in the one answer set and $\neg perfected$ is true in the another. Thus according to Theorem 5.10, neither $perfected$ nor $\neg perfected$ is proved, i.e.,

$$Circum(\widetilde{\forall}A; ab1, ab2; perfected) \not\models perfected$$
$$Circum(\widetilde{\forall}A; ab1, ab2; perfected) \not\models \neg perfected$$

since $S_{pri} = \phi$ is derived under the given preferences of Γ as is shown in Example 5.7. On the other hand, $\Pi \cup \Gamma_1 \cup \mathcal{E}$ has a unique answer set in which $\neg perfected$ is true. According to Theorem 5.10, this means

$$Circum(\widetilde{\forall}A; ab2 > ab1; perfected) \models \neg perfected$$

since $S_{pri} = \{ab2 > ab1\}$ is derived under the corresponding preferences of Γ_1 shown in Example 5.7. In a similar way, $\Pi \cup \Gamma_2 \cup \mathcal{E}$ has also unique answer set in which $perfected$ is true. This means

$$Circum(\widetilde{\forall}A; ab1 > ab2; perfected) \models perfected$$

according to Theorem 5.10 as wellas $S_{pri} = \{ab1 > ab2\}$ under Γ_2 shown in Example 5.7.

5.3 Related Works

Lifschitz [Lifschitz, 1987a] proposed a method which allows us to formalize some forms of metalevel reasoning in the circumscriptive theory by using his policy axioms. In his formalism, when both $V[ab1 : ab2]$ and $V[ab2 : ab1]$ as the policy axioms exist, there is possibility that the circumscriptive theory becomes inconsistent due to the minimality condition (p.139 in [Lifschitz, 1989]). We do not consider such a case as contradiction, but would like to consider that there is no priority between $ab1$ and $ab2$ since Lifschitz's policy axiom represents a pre-order relation of the priority. In our method, the circumscriptive theory does not become inconsistent under the existence of such policy axioms since $ab1$ and $ab2$ are minimized in parallel.

Brewka [Brewka, 1996] presented a method to integrate dynamic preferences into default logic and logic programming. His method can be regarded as giving the model-theoretical formulation to the approach presented by Prakken and Sartor [1995]. His approach is achieved by using a prioritized logic program(PLP) proposed by him as well as the semi-normal default theory. PLP can describe preferences between rules expressed by extended logic programs and its time complexity is polynomial since the declarative meaning is given by an extended well-founded semantics. However our method is more profitable for the purpose of the legal reasoning since more useful informations of the application domain can be obtained by stable semantics and answer set semantics than the well-founded one.

For example, in the regal reasoning, preferences and priorities among rules as well as metarules are often required for attorney-at-law to find out in order to derive a desired conclusion which gives him the advantage in the argumentation of court. Since our approach is especially based on the stable models as well as answer sets which are not always unique like a well-founded model, we can find out such not-unique candidate preferences or priorities as explanations in abduction from the stable models or answer sets, each of which leads to derive a desired conclusion. Thus there is such a merit in our approach that the attorney can either choose the most adequate preference or priority for him or obtains some or all of the candidate ones depending on his needs and situation. Our research about finding preferences and priorities will be reported in our subsequent paper.

Besides as far as adequate preferences are given, the reasoning with dynamic preferences can be formalized by prioritized circumscription or parallel circumscription and can be computed by our approach. On the other hand, sometimes reasonable conclusions are not obtained by using his method since its semantics inherits some drawbacks of the well-founded semantics which is addressed in Brewka's paper [Brewka, 1996].

Regarding Prakken and Sartor's approach, their method is procedural though our approach as well as Brewka's one are based on a model theoretic semantics. We think that the model theoretic approach is important in a sense that, it is easier for the model theoretic approach than the procedural approach to examine and verify various properties of the reasoning about dynamic preferences since the model theoretic approach has the mathematical foundation and is more abstract than the procedural one.

5.4 Summary

The crucial characteristics of the legal reasoning regarded as the commonsense reasoning is that it inevitably requires the metalevel reasoning about preferences among laws like Lex Superior, etc. along with the object-level legal reasoning, that is, the needs of reasoning to handle dynamic preferences correctly.

When we formalize such a reasoning by means of circumscription, the metalevel reasoning of preferences among laws leads to determine the circumscription policy,

i.e. priorities among the abnormal predicates of circumscription in the object-level reasoning. The framework of circumscription, however, obliges us to pre-specify the circumscription policy. Thus, there is no way to reason its circumscription policy itself as well as no way to intervene between the metalevel reasoning of the circumscription policy and the non-monotonic reasoning of the object domain based on the circumscription policy derived in the metalevel.

In this paper, we propose an approach which overcomes such difficulties. Our approach enables us to handle some class of dynamic preferences in the framework of circumscription and compute it by logic programming based on Wakaki and Satoh's method [Wakaki and Satoh, 1997].

When priority relations obtained in the metalevel reasoning do not have correspondence to a division of minimized predicates given in Definition 5.8, our method shown in Theorem 5.10 does not become applicable. However this does not always mean the limitation of our method. This is because depending on the lack of preferences or the information structures of preferences, various priority orderings among minimized predicates each of which does not correspond to parallel circumscription or prioritized circumscription, may be derived in the metalevel reasoning. For example, such a case may be obtained in the metalevel reasoning that there exist many disjoint subsets of minimized predicates where each subset has some priority ordering among its predicates, but there are no ordering among those subsets. How to formalize such cases by extending the framework of circumscription is our future problem.

5.5 Proofs

Proof of Theorem 5.10:

According to Lemma 5.4, a normal logic program $\Pi \cup \Gamma$ has a unique stable model. Let N_1 be a unique stable model of $\Pi \cup \Gamma$ constructed from a given M according to Definition 5.3. Then according to Definition 5.6, S_{pri} can be obtained as follows:

$$S_{pri} = N_1 \cap \{p_i > p_j |\ \forall (p_i, p_j) \in P \times P,\ p_i \neq\ p_j\}.$$

Here it is supposed that this S_{pri} has a 1-1 correspondence to some division $\left[P^1, \ldots, P^k\right]$ of a given P such that $md(S_{pri}) = \left[P^1, \ldots, P^k\right]$.

Therefore we can conclude that prioritized circumscription :

$$Circum(\widetilde{\forall} A; P^1 > \ldots > P^k; Z)$$

is derived from the given A, P, Z and M.
Now according to Theorem 4.5 as well as Theorem 4.6, for any ground literal G of the language of A whose predicate symbol is not equality, it holds that,

$Circum(\widetilde{\forall} A; P^1 > \ldots > P^k; Z) \models G,$

iff G is true in all answer sets of either *ELP* Π^β (in case of $k \neq 1$)
constructed from $Circum(\widetilde{\forall} A; P^1 > \ldots > P^k; Z)$ by using Definition 4.3

or $ELP\ \Pi^{\alpha}$ (in case of $k = 1$) constructed by using Definition 4.2.

On the other hand, it holds that

G is true in all answer sets of either $ELP\ \Pi^{\beta}$ (in case of $k \neq 1$)
constructed from $Circum(\widetilde{\forall} A; P^1 > \ldots > P^k; Z)$ by using Definition 4.3
or $ELP\ \Pi^{\alpha}$ (in case of $k = 1$) constructed by using Definition 4.2,

iff G is true in all answer sets of
$$\mathcal{E}(\left[P^1, \ldots, P^k\right]) \cup S_{pri}.$$

because according to Definition 5.9, the conditional parts of all rules in $\mathcal{E}(\left[P^1, \ldots, P^k\right])$ are satisfied by literals in S_{pri} such that $md(S_{pri}) = \left[P^1, \ldots, P^k\right]$, which has the same effect in the answer set semantics as that of a extended logic program: $ELP\Pi^{\beta}$ compiled from $Circum(\widetilde{\forall}\ A; P^1 > \ldots > P^k; Z)$ (or $ELP\ \Pi^{\alpha}$ in case of k=1).

Furthermore, it holds that

G is true in all answer sets of $\mathcal{E}(\left[P^1, \ldots, P^k\right]) \cup S_{pri}$,

iff G is true in all answer sets of $\mathcal{E} \cup S_{pri}$.

because any rule in $\mathcal{E}$ except rules in $\mathcal{E}(\left[P^1, \ldots, P^k\right])$ is not applicable since its conditional part is not satisfied by S_{pri}.

Since N_1 is a unique stable model of $\Pi \cup \Gamma$ such that,
$$N_1 \models p_i > p_j \quad \text{iff} \quad p_i > p_j \in S_{pri}$$
it holds that,

G is true in all answer sets of $\mathcal{E} \cup S_{pri}$,

iff G is true in all answer sets of $\mathcal{E} \cup \Pi \cup \Gamma$. $\square$

Chapter 6

Finding Priorities of Circumscription Policy

6.1 Introduction

With respect to circumscription, a formula G is a theorem of a *circumscriptive theory* (A, C) if G is true in all models of an axiom set A that are minimal under some condition given by a *circumscription policy* C. There may be several different circumscription policies which specify the forms of circumscription. For example, with respect to prioritized circumscription, minimized predicates and their priorities as well as variable predicates should be pre-specified as the circumscription policy.

So far, it has been required in the commonsense reasoning that, precedences (i.e. priorities) among the conflicting rules should be found out in order to derive a natural conclusion of human reasoning. The following Yale Shooting Problem [Hanks and McDermott, 1987] is a motivating example of our research.
In general, it holds that if the gun is loaded at the time T_i, it is also loaded at the next time T_{i+1} and if a person is alive at the time T_j, he is also alive at the next time T_{j+1}. If the gun is loaded, then the person will not be alive after an action, SHOOT. Information about an action, WAIT, is not given. Initially, at the time T_0, the gun is loaded and an action, Wait, is done. Next, at the time T_1, it is known that the person is alive and an action, SHOOT is done. We intend that the person is dead at the time T_2 due to the sequence of actions WAIT and SHOOT and we would like to know how to express this reasoning.

The above knowledge is represented by the following axiom set A where $ab1$ and $ab2$ are abnormal predicates [Lifschitz, 1985]. $loaded_i$ and $alive_j$ denote that the gun is loaded at the time T_i and the person is alive at the time T_j respectively.

$$
\begin{aligned}
A: \quad & loaded_0 \wedge \neg ab1 \supset loaded_1, && (1)\\
& alive_1 \wedge \neg ab2 \supset alive_2, && (2)\\
& loaded_1 \supset \neg alive_2,\\
& loaded_0,\\
& alive_1.
\end{aligned}
$$

Since we do not know the reason which default of (1) or (2) should take precedence, that is, in other words: which $ab1$ or $ab2$ should have the higher priority for minimization in circumscription, this problem is formalized by parallel circumscription as follows:

$$Circum(\widetilde{\forall} A; ab1, ab2; alive_2)$$

To the contrary of our intention that $\neg alive_2$ should be inferred, this leads to the conclusion that neither $alive_2$ nor $\neg alive_2$ is decided which reveals so called the multiple extension problem. Historically since prioritized circumscription was proposed to solve such multiple extension problem, the criterion (i.e. priority) was finally found out manually for this Yale Shooting Problem (for example, see [Satoh, 1988]) that $ab1$ should be minimized with higher priority than $ab2$ in order to derive the human natural conclusion $\neg alive_2$ in the framework of circumscription.

Since the Yale Shooting Problem has historically clarified that the human commonsense reasoning needs the preferences (or priorities) between the commonsense knowledge, the reasoning about the the priorities among the commonsense knowledge is inevitably required in order to realize the nonmonotonic reasoning on a computer-aided intelligent system which can handle the commonsense reasoning. Furthermore, the Yale Shooting Problem is so simple that we can find the above priorities manually. However, since it becomes more complicated and difficult the commonsense reasoning. Furthermore, the Yale Shooting Problem is so simple that we can find the above priorities manually. However, since it becomes more complicated and difficult to find priorities among a lot of complicated default (commonsense) knowledge used in a real world's commonsense reasoning, we can no longer find them manually and hence, a method of finding such priority information automatically is required.

Thus we aim at providing a method for finding priorities, i.e. a part of a circumscription policy C automatically in order to infer a desired conclusion G as a theorem of the circumscriptive theory (A, C).

In this chapter, first we present top-down as well as bottom-up procedures to compute skeptical explanations from a consistent abductive framework and an observation, and secondly show that priorities of circumscription to infer a desired conclusion can be found out as a skeptical explanation in abductive logic programming. In our approach, since priority relations between minimized predicates are represented by axioms, we can handle them as hypotheses or abducibles. Thus priorities can be computed based on the procedure to compute skeptical explanations provided in this chapter as well as Wakaki and Satoh's method [Wakaki and Satoh, 1997] of compiling circumscription into extended logic programs.

The rest of this chapter is organized as follows. In Section 6.2, we give a method of computing a skeptical explanation by using abductive logic programming. In Section 6.3, we present our method of finding priorities as a part of a circumscription policy and gives the summary in Section 6.4.

6.2 Computing Skeptical Explanation by Abductive Logic Programming

So far, procedures (e.g. [Satoh and Iwayama, 1991],[Satoh and Iwayama, 1992]) to compute *credulous* explanations from a consistent abductive framework and an observation have been proposed, but as far as we know, neither any method has been proposed to compute skeptical explanations, nor needs for skeptical explanations have been reported.

In this section, we present top-down as well as bottom-up procedures to compute skeptical explanations which we need in the next section. Firstly we slightly extend an abductive framework of [Kakas and Mancarella, 1990] based on the relationship between stable model semantics and answer set semantics [Gelfond and Lifschitz, 1991] in order to use not a normal logic program but an extended logic program as follows.

Definition 6.1 *An abductive framework is a pair $\langle T, H \rangle$ where T and H are as follows:*

1. *H is a set of ground literals of the language of T and is called hypotheses or abducibles.*
2. *T is an extended logic program called the facts. The head of each rule from T are not in H.*

Definition 6.2 *Let $\langle T, H \rangle$ be an abductive framework and E be a subset of H. A hypothetical answer set $As(E)$ of $\langle T, H \rangle$ is an answer set of $T \cup \mathcal{F}(E)$ where $\mathcal{F}(E)$=$\{h \leftarrow | h \in E\}$. $\langle T, H \rangle$ is consistent if it has a consistent hypothetical answer set.*

The following definitions of *credulous explanation* and *skeptical explanation* are given by [Inoue and Sakama, 1998].

Definition 6.3 *Let $\langle T, H \rangle$ be a consistent abductive framework, G be a ground literal called an observation and E be a subset of H. G has a credulous explanation E (or E is a credulous explanation of G) w.r.t. $\langle T, H \rangle$ if and only if there exists a hypothetical answer set $As(E)$ of $\langle T, H \rangle$ such that $G \in As(E)$. On the other hand, G has a skeptical explanation E (or E is a skeptical explanation of G) w.r.t. $\langle T, H \rangle$ if and only if for every hypothetical answer set $As(E)$ of $\langle T, H \rangle$, $G \in As(E)$.*

Theorem 6.4 *Let $\langle T, H \rangle$ be a consistent abductive framework, G be an observation and G' be a ground literal which does not occur in T and H. Then G has a skeptical explanation E with respect to $\langle T, H \rangle$ if and only if E is a credulous explanation of G w.r.t. $\langle T, H \rangle$, but not a credulous explanation of G' w.r.t. $\langle T \cup \{G' \leftarrow not\ G\ \}, H \rangle$.*
Proof: *See Appendix.*

Step 1: Compute a set $C1$ which consists of every credulous explanation $E1$ of G w.r.t. $\langle T, H\rangle$ such that $E1 = S1 \cap H$ for every answer set $S1$ of an extended logic program: $T \cup \Gamma(H) \cup \{\leftarrow not\ G\}$.

Step 2: Compute a set $C2$ which consists of every credulous explanation $E2$ of G' w.r.t. $\langle T, H\rangle$ such that $E2 = S2 \cap H$ for every answer set $S2$ of $T \cup \{G' \leftarrow not\ G\ \} \cup \Gamma(H) \cup \{\leftarrow not\ G'\}$.

Step 3: Compute a set $C3$ which consists of all skeptical explanations of G w.r.t. $\langle T, H\rangle$ such that $C3 = \{E | E \in C1$ and $E \notin C2\}$.

Figure 6.1 A Bottom-up Procedure for Skeptical Explanations

6.2.1 A Bottom-up Procedure for Skeptical Explanations

A consistent abductive framework $\langle T, H\rangle$ and a ground literal G as an observation are given. Then a bottom-up procedure described in Figure 6.1 computes a set $C3$ of skeptical explanations of G w.r.t. $\langle T, H\rangle$, which faithfully performs Theorem 6.4 by using Theorem 6.5 below.

Theorem 6.5 *Let $\langle T, H\rangle$ be a consistent abductive framework and G be an observation. We define $\Gamma(H)$ as a set of rules as follows:*

$$\Gamma(H) \stackrel{def}{=} \{h' \leftarrow not\ h,\ h \leftarrow not\ h' |\ h \in H\ \}$$

where h' is a newly introduced ground literal corresponding to each ground literal $h \in H$. Then G has a credulous explanation E w.r.t. $\langle T, H\rangle$ if and only if there is an answer set S for $T \cup \Gamma(H) \cup \{\leftarrow not\ G\}$ such that $E = S \cap H$.

Proof: This theorem is proved in a slightly extended way of the proof of Theorem 2.9 in [Satoh and Iwayama, 1991].

6.2.2 A Top-down Procedure for a Skeptical Explanation

In general, a top-down procedure is more efficient than a bottom-up one since it searches only the parts of a search tree which depend on a given goal whereas a bottom-up procedure must generate hypotheses even which do not depend on a goal.

Satoh and Iwayama [Satoh and Iwayama, 1992] presented a top-down query evaluation procedure which answers w.r.t. a credulous explanation. In Figure 6.2, we present a top-down procedure to compute a skeptical explanation, which extends Satoh and Iwayama's top-down query evaluation procedure to make it possible to answer w.r.t. a skeptical explanation. Given a consistent abductive

Step 1: **Select** E such that $derive(G, \{\})$ succeeds with (ε, E) under an abductive framework $\langle T \cup \Phi, H \rangle$ and go to **Step 2**. If such E does not exist, then return.

Step 2: If $derive(not\ G, \{\})$ fails under an abductive framework $\langle T \cup \Phi \cup \mathcal{F}(E), \{\} \rangle$, then return E, else go back to the recent choice point in **Step 1**.

Figure 6.2 A Top-down Procedure for a Skeptical Explanation

framework $\langle T, H \rangle$ and a ground literal G as an observation, a skeptical explanation E of G w.r.t. $\langle T, H \rangle$ is computed by the top-down procedure, which also performs Theorem 6.4 where Satoh and Iwayama's *derive* [Satoh and Iwayama, 1992] is used by slightly extending their Theorem 6.6 below.

Theorem 6.6 *(Satoh and Iwayama, [Satoh and Iwayama, 1992])*
Let $\langle T, \mathcal{A} \rangle$ be a consistent abductive framework, p be a non-abducible atom, θ be a substitution and Δ be a subset of abducibles. Suppose $derive(p, \{\})$ succeeds with (θ, Δ). Then there exists a generalized stable model $M(\Theta)$ for T such that Θ includes all positive abducibles in Δ and $M(\Theta) \models p\theta$.

In Figure 6.2, Φ is an extended logic program defined as follows:

$$\Phi \stackrel{def}{=} \bigcup_i \{ \leftarrow u_i(x), \neg u_i(x) \}$$

where u_i is a predicate symbol occurring in T and H. ε stands for identity substitution. **Select** in **Step 1** expresses nondeterminism.

6.3 Finding Priorities of Circumscription

We consider a priority-finding problem in the framework of a circumscriptive theory (A, C) where A is a sentence and C is the circumscription policy. We suppose that A is given by a set of clauses and a tuple P of minimized predicates, a tuple Z of variable predicates are already specified as the circumscription policy, but priorities among minimized predicates which is a part of C are unknown and are required to be found out in order to infer a desired formula G as a theorem of the circumscriptive theory (A, C). Then, in this chapter, we express a circumscriptive theory as (A, P, Z, I_P) instead of (A, C) for notational convenience as follows.

Definition 6.7 *Let (A, P, Z, I_P) be a circumscriptive theory where A is a set of clauses, P, Z are tuples of n minimized predicates, and variable predicates respectively and I_P is the priority information among the members of P which is represented by a disjoint division $\left[P^1, \ldots, P^k\right]$ $(1 \leq k \leq n)$ of a tuple P with assigning a higher priority to the members of P^i than to the members of P^j for $i < j$.*

Then P, Z and I_P give the complete information for a circumscription policy C w.r.t parallel circumscription as well as prioritized circumscription. Hereafter we identify $Circum(\widetilde{\forall} A; I_P; Z)$ where $I_P = \left[P^1, \ldots, P^k\right]$ with prioritized circumscription such as $Circum(\widetilde{\forall} A; P^1 > P^2 > \ldots > P^k; Z)$.

In our main theorem, we show that priority information among minimized predicates given by I_P can be found out as a skeptical explanation in abduction.

6.3.1 Representation of Priorities

With respect to prioritized circumscription, there is no way to express the priorities among minimized predicates within its framework as well as no way of reasoning about priorities. In order to describe the priority information explicitly, we use a special binary (infix) symbol $>$ that can take minimized predicates as arguments. $>$ represents a priority relation between them by a strict partial order, i.e. an irreflexive and transitive relation. Then the priority information of prioritized circumscription is described by using priority relations as follows:

Definition 6.8 *Prioritized circumscription:*
$Circum(\widetilde{\forall} A; I_P; Z)$
where $I_P = \left[P^1, \ldots, P^k\right] \qquad (1 \leq k \leq n)$
including parallel circumscription in case of $k = 1$ is given. Let P^i $(1 \leq i \leq k)$ be a tuple as follows:

$$P^i \stackrel{def}{=} \left[P_{i_1}, \ldots, P_{i_{n_i}}\right] \qquad (\textstyle\sum_i n_i = n)$$

Then I_P stands for the following priority relations:

- *the existence of*

  $P_{i_\ell} > P_{j_m}$ *for any* i, j, ℓ, m
  $(1 \leq i < j \leq k,\ 1 \leq \ell \leq n_i,\ 1 \leq m \leq n_j)$

- *and no existence of*

  $P_{i_\ell} > P_{i_m}$ *for any* i, ℓ, m
  $(1 \leq i \leq k,\ 1 \leq \ell, m \leq n_i,\ \ell \neq m)$

and vice versa.

6.3.2 Abducing Priorities as Hypotheses

In order to represent priority relations in a logical language, hereafter we use a symbol $>$ as a binary predicate symbol and introduce a new individual constant p_{i_j} corresponding to each minimized predicate P_{i_j} $(1 \leq i \leq k, 1 \leq j \leq n_i)$. Let p^i be a tuple of newly introduced individual constants which corresponds to a tuple P^i of minimized predicates as follows:

$$p^i \stackrel{def}{=} \left[p_{i_1}, \ldots, p_{i_{n_i}}\right]. \qquad (\textstyle\sum_i n_i = n)$$

Thus each priority relation: $P_{i_\ell} > P_{j_m}$ is expressed by a corresponding priority axiom $p_{i_\ell} > p_{j_m}$ for any $i,\ j,\ \ell,\ m\ \ (1 \leq i < j \leq k, 1 \leq \ell \leq n_i, 1 \leq m \leq n_j)$.

Definition 6.9 *We define a set Pri which consists of all priority axioms as follows:*

$$Pri \stackrel{def}{=} \{p_i > p_j |\ \forall (p_i, p_j) \in P_c \times P_c,\ p_i \neq\ p_j\}$$

where P_c be a tuple of all newly introduced constants corresponding to a tuple P of all minimized predicates.

Definition 6.10 *We define a program Ω which expresses one of mathematical property of priority axioms, that is, an irreflexive relation as follows:*

$$\Omega : \qquad\qquad \leftarrow x > y,\ y > x,$$

where x, y, z are variables.

Definition 6.8 is said in other words as follows:

Definition 6.11 *A circumscriptive theory (A, P, Z, I_P) is given where $I_P = \left[P^1, \ldots, P^k\right]$. We define a 1-1 correspondence between I_P and the following set $S \subset Pri$ which*

- *includes the priority axioms as follows:*

  $p_{i_\ell} > p_{j_m}$ *for any* i, j, ℓ, m
  $(1 \leq i < j \leq k,\ 1 \leq \ell \leq n_i,\ 1 \leq m \leq n_j)$

- *and does not include the priority axioms as follows:*

  $p_{i_\ell} > p_{i_m}$ *for any for any* i, ℓ, m
  $(1 \leq i \leq k,\ 1 \leq \ell, m \leq n_i,\ \ell \neq m)$

where $p_{i_\ell} \in p^i$ and $p_{j_m} \in p^j$ correspond to $P_{i_\ell} \in P^i$ and $P_{j_m} \in P^j$ respectively. Then with respect to this 1-1 correspondence, we define a function md such that $md(S) = I_P$.

Note: *The priority axioms included in S satisfy a transitive relation as follows:*

$$x > z \leftarrow x > y,\ y > z.$$

where x, y, z are variables.

In the following definition, firstly a set of extended logic programs *ELP* Π^β (or *ELP* Π^α) corresponding to all possible prioritized circumscription is generated according to the method in [Wakaki and Satoh, 1997] and then conditions about priority relations are introduced to each rule in every generated *ELP* Π^β (or *ELP* Π^α).

Definition 6.12 *Let A be a set of clauses and P, Z be tuples of minimized and variable predicates occurring in A. Then $\mathcal{E}$ is an extended logic program which is defined as follows:*

1. *Generate every division $\left[P^1, \ldots, P^k\right]$ of a given P.*

2. *For each $\left[P^1, \ldots, P^k\right]$, do from (a) to (c),*

   (a) *Let $Circum(\widetilde{\forall} A; P^1 > \ldots > P^k; Z)$*
   i.e. $Circum(\widetilde{\forall} A; \left[P^1, \ldots, P^k\right]; Z)$
   be compiled into an extended logic program: ELP Π^β *according to Definition 4.3 . If $k = 1$, compile $Circum(\widetilde{\forall} A; P; Z)$ into* ELP Π^α *according to Deffinition 4.2.*

   (b) *For every rule of Π^β (or Π^α) obtained in (a) such as*
   $$L \leftarrow L_1, \ldots, L_m, notL_{m+1}, \ldots, notL_n,$$
   add all priority axioms in S such that $md(S) = \left[P^1, \ldots, P^k\right]$ to its body according to Definition 6.11 as follows:
   $$L \leftarrow \bigwedge_{i,j,\ell.m} (p_{i_\ell} > p_{j_m}) \wedge \bigwedge_{i,\ell,m} not\ (p_{i_\ell} > p_{i_m})$$
   $$\wedge L_1 \wedge \ldots, L_m \wedge notL_{m+1} \wedge \ldots \wedge notL_n,$$
   where a symbol "$\wedge$" is used instead of a symbol "," for the notational clarity.

   (c) *Let $\mathcal{E}(\left[P^1, \ldots, P^k\right])$ be a set of all rules generated in (b).*

3. *$\mathcal{E}$ is a union of all $\mathcal{E}(\left[P^1, \ldots, P^k\right])$.*

We show a main theorem of this paper as follows.

Theorem 6.13 *For every circumscriptive theory (A, P, Z, I_P) and every ground literal G of the language of A whose predicate symbol is not equality,*

$Circum(\widetilde{\forall} A; I_P; Z) \models G$
iff G *has a skeptical explanation E w.r.t. $\langle \mathcal{E} \cup \Omega, Pri \rangle$*
such that $md(E) = I_P$

where $\mathcal{E} \cup \Omega$ is a union of logic programs defined in Definition 6.10 and Definition 6.12, and Pri is a set defined in Definition 6.9.

Proof: See Appendix.

Thus we have obtained the result that given A,P,Z, the priority information I_P to infer a desired conclusion G from circumscription can be computed as a skeptical explanation E of G w.r.t. $\langle \mathcal{E} \cup \Omega, Pri \rangle$ such that $md(E) = I_P$ based on Theorem 6.13.

Example 6.14 Let us consider the Yale Shooting Problem in Section 6.1 and formalize it by a circumscriptive theory (A, P, Z, I_P). A is given in Section 6.1 and P, Z, Q are specified as follows:

$$P = \{ab1,\ ab2\}, \quad Z = \{alive_2\},$$
$$Q = \{loaded_0,\ loaded_1,\ alive_1\},$$

but priorities between minimized predicates in P are unknown and is required to be found out as I_P in order to derive a desired conclusion $\neg alive_2$.

Firstly a tuple P_c corresponding to P is defined by using newly introduced individual constants $ab1', ab2'$ as follows: $P_c \stackrel{def}{=} \{ab1', ab2'\}$. According to Definition 6.9, Pri consists of the following priority axioms:

$$Pri = \{ab1' > ab2',\ ab2' > ab1'\}.$$

On the other hand, all different divisions of $P = \{ab1, ab2\}$ are represented as follows:

$$[[ab1], [ab2]], \quad [[ab2], [ab1]], \quad [[ab1, ab2]]$$

each of which corresponds to the following circumscription respectively:

$$Circum(\widetilde{\forall} A; ab1 > ab2; Z),$$
$$Circum(\widetilde{\forall} A; ab2 > ab1; Z),$$
$$Circum(\widetilde{\forall} A; ab1, ab2; Z).$$

According to Definition 6.12, the following extended logic program $\mathcal{E}$ is obtained:

$$\mathcal{E} \stackrel{def}{=} \mathcal{E}([[ab1], [ab2]]) \cup \mathcal{E}([[ab2], [ab1]]) \cup \mathcal{E}([[ab1, ab2]]),$$

by using the method in [Wakaki and Satoh, 1997] (i.e. Theorem 4.5 and 4.6) of compiling parallel circumscription and prioritized circumscription into ELP Π^α and Π^β respectively. Then, we can obtain a consistent abductive framework $\langle \mathcal{E} \cup \Omega, Pri \rangle$ of this example by Theorem 6.13.

If we use the bottom-up procedure described in Figure 6.1, $C1 = \{\{ab1' > ab2'\}, \phi\}$ is obtained in **Step 1** from $\langle \mathcal{E} \cup \Omega, Pri \rangle$, $C2 = \{\{ab2' > ab1'\}, \phi\}$ is obtained in **Step 2** and $C3 = \{\{ab1' > ab2'\}\}$ is obtained in **Step 3**.

Thus from $C3$, we find out $\{ab1' > ab2'\}$ as only one skeptical explanation of $\neg alive_2$. Since $md(\{ab1' > ab2'\})$=[[ab1],[ab2]], we get $I_P = [[ab1], [ab2]]$, which leads to the following result according to Theorem 6.13:

$$Circum(\widetilde{\forall} A; [[ab1], [ab2]]; Z) \models \neg alive_2$$
$$\text{i.e.} \quad Circum(\widetilde{\forall} A; ab1 > ab2; Z) \models \neg alive_2.$$

If we use the top-down procedure described in Figure 6.2, the same result is computed in the following way. Let Φ be a set defined in Section 6.2.

Step 1_1**:** If $E = \phi$ is selected such that $derive(\neg alive_2, \{\})$ succeeds with (ε, ϕ) under $\langle \mathcal{E} \cup \Omega \cup \Phi, Pri \rangle$, then go to **Step** 2_1.

Step 2_1**:** $derive(not\ \neg alive_2, \{\})$ succeeds with (ε, ϕ) under $\langle \mathcal{E} \cup \Omega \cup \Phi \cup \phi, \{\} \rangle$ and go back to the recent choice point in **Step** 1_2.

Step 1_2**:** If $E = \{ab1' > ab2'\}$ is selected such that $derive(\neg alive_2, \{\})$ succeeds

with $(\varepsilon, \{ab1' > ab2'\})$ under $\langle \mathcal{E} \cup \Omega \cup \Phi, Pri \rangle$, then go to **Step** 2_2.

Step 2_2**:** $derive(not\ \neg alive_2, \{\})$ fails under $\langle \mathcal{E} \cup \Omega \cup \Phi \cup \{ab1' > ab2' \leftarrow\}, \{\} \rangle$. Then return $\{ab1' > ab2'\}$.

Thus we obtain again the result as follows:

$$Circum(\widetilde{\forall} A; ab1 > ab2; Z) \models \neg alive_2.$$

Example 6.15 Let us consider a legal example in Chapter 5. We suppose that only the information expressed by a set A of axioms in Section 5.1 is given and she wants that her security interest in the ship is perfected. But since there is no preference information between the UCC and the SMA, neither $perfected$ nor $\neg perfected$ is not derived. Then the problem is that in order to let her security interest be $perfected$, we would like to know which the UCC and the SMA takes precedence, that is, in other words: which $ab1$ or $ab2$ has the higher priority for minimization in circumscription.

According to Theorem 6.13, this problem is formalized by a circumscriptive theory (A, P, Z, I_P) where A is given in Section 5.1 and P, Z, Q are specified as follows:

$$P = \{ab1, ab2\}, \qquad Z = \{perfected\},$$
$$Q = \{possession,\ ship,\ \neg fin{-}statement\}.$$

but priorities between minimized predicates in P are unknown and is required to be found out as I_P in order to derive a desired conclusion $prefected$. Now as this becomes substantially the same problem of example 6.14, we get $I_P = [[ab1], [ab2]]$, which leads to the following result according to Theorem 6.13.

$$Circum(\widetilde{\forall} A; [[ab1], [ab2]]; Z) \models perfected$$

Example 6.16 Consider the parallel circumscription $Circum(\{p \vee q,\ q \vee r\};\ p, q, r)$. Then, q is not its theorem, that is,

$$Circum(\{p \vee q,\ q \vee r\};\ p, q, r) \not\models q$$

Our problem is to find out I_P, i.e. priorities among p, q, r so that q may become a theorem of the circumscriptive theory (A, P, Z, I_P) where $A = \{p \vee q,\ q \vee r\}$, $P = \{p, q, r\}$ and $Z = \phi$. We can find such I_P's based on Theorem 6.13 as follows.

First, we define a tuple P_c of newly introduced constants as $\{p', q', r'\}$ where p', q', r' correspond to p, q, r respectively. Then a set Pri is as follows:

$$Pri = \{p' > q',\ q' > p',\ q' > r',\ r' > q',\ r' > p', p' > r'\}$$

On the other hand, $\mathcal{E}$ is an extended logic program constructed according to Definition 6.12 as follows:

$$\mathcal{E}([[p], [q], [r]]) \cup \mathcal{E}([[p], [r], [q]]) \cup \mathcal{E}([[q], [r], [p]]) \cup \mathcal{E}([[q], [p], [r]]) \cup$$
$$\mathcal{E}([[r], [p], [q]]) \cup \mathcal{E}([[r], [q], [p]]) \cup \mathcal{E}([[p], [q, r]]) \cup \mathcal{E}([[q], [r, p]]) \cup$$

$$\mathcal{E}([[r],[p,q]]) \cup \mathcal{E}([[p,q],[r]]) \cup \mathcal{E}([[q,r],[p]]) \cup \mathcal{E}([[r,p],[q]]) \cup \mathcal{E}([[p,q,r]])$$

Thus an abductive framework constructed from a given parallel circumscription is $< \mathcal{E} \cup \Omega, Pri >$ where $\mathcal{E}$ and Pri are obtained above and Ω is given by Definition 6.10.

By using a bottom-up or top-down procedure to compute a skeptical explanation shown in Section 6.2, seven skeptical explanations E's of q each of which is a subset of Pri are obtained, which lead to seven I_P's such that $md(E) = I_P$ according to Definition 6.11 as follows:

$$[[r,p],[q]], \quad [[p],[r],[q]], \quad [[p],[q,r]], \quad [[p],[q],[r]],$$
$$[[r],[p,q]], \quad [[r],[p],[q]], \quad [[r],[q],[p]].$$

Thus, these I_P's denote seven prioritized circumscription each of which derives q as its theorem as follows.

$$Circum(\{p \vee q,\ q \vee r\};\ p,\ r > q) \models q$$
$$Circum(\{p \vee q,\ q \vee r\};\ p > r > q) \models q$$
$$Circum(\{p \vee q,\ q \vee r\};\ p > q,\ r) \models q$$
$$Circum(\{p \vee q,\ q \vee r\};\ p > q > r) \models q$$
$$Circum(\{p \vee q,\ q \vee r\};\ r > p,\ q) \models q$$
$$Circum(\{p \vee q,\ q \vee r\};\ r > p > q) \models q$$
$$Circum(\{p \vee q,\ q \vee r\};\ r > q > p) \models q$$

6.4 Summary

So far, based on the theory of abduction, Ginsberg [Ginsberg, 1989] and Inoue [Inoue, 1992a] have proposed the algorithms for determining whether or not some potential conclusion follows from a given circumscriptive theory, but we do not know such a research as to find (a part of) the circumscription policy itself in order to derive a desired conclusion. In this chapter, we present a method of finding priorities, i.e. a part of the circumscription policy as a skeptical explanation in abduction, which are based on the abductive logic programming procedure to compute skeptical explanations presented in this paper as well as Wakaki and Satoh's method [Wakaki and Satoh, 1997] of compiling circumscription into extended logic programs.

There are many examples of non-monotonic reasoning that require to find out priorities among conflicting rules as the Yale shooting problem is shown in Section 6.1.

For example, in such a scheduling problem that employees or machines should be assigned to jobs in a factory, there are various factors such as the amount of products, the sum of working hours, hours of overtime work and so on which effect to the result of the schedule. Then it is required to compute automatically which factors should be more preferably considered than others (i.e. priorities among

conflicting factors to be considered in scheduling) in order to derive an employer's desirable result.

As another example, in the legal domain, priorities among the conflicting laws are often required for disputers to find out in order to derive a desired conclusion which give them the advantage in the argumentation of court. Our proposing method enables us to derive a desired goal by abducing the appropriate priorities in such cases.

Our method, however, has a problem of the computational complexity since it inherits the time complexity of Wakaki and Satoh's method mentioned in section 4.3. Our future work is to suppress the combinatorially increasing numbers of rules in an extended logic program $\mathcal{E}$ defined in this paper with increasing numbers of minimized predicates.

6.5 Proofs

Proof of Theorem 6.4

($\Rightarrow$) Suppose that E is a credulous explanation of G' w.r.t. $\langle T \cup \{G' \leftarrow not\ G\}, H\rangle$ where E is a credulous explanation of G w.r.t. $\langle T, H\rangle$.

Now according to Definition 6.3, there exists some hypothetical answer set $S_1(E)$ of $\langle T, H\rangle$ such that $G \in S_1(E)$.

On the other hand, since E is also a credulous explanation of G' w.r.t. $\langle T \cup \{G' \leftarrow not\ G\}, H\rangle$, there exists some answer set $S_2(E)$ of $T \cup \{G' \leftarrow not\ G\} \cup \mathcal{F}(E)$ such that $G' \in S_2(E)$.

It is easily proved that for any answer set S of $T \cup \{G' \leftarrow not\ G\} \cup \mathcal{F}(E)$,

(1) $G' \in S$ iff $G \notin S$.

(2) $S - \{G'\}$ is an answer set of $T \cup \mathcal{F}(E)$.

Thus we have obtained a result that there exists some answer set $S_2(E) - \{G'\}$ of $T \cup \mathcal{F}(E)$ such that $G \notin S_2(E) - \{G'\}$. This means that for some hypothetical answer set $As(E)$ of $\langle T, H\rangle$ (i.e. $S_2(E) - \{G'\}$), $G \notin As(E)$.

Therefore according to Definition 6.3, this contradicts that E is a skeptical explanation of G w.r.t $\langle T, H\rangle$.

($\Leftarrow$) With respect to from right to left, it can also be proved in a similar way. □

Proof of Theorem 6.13

According to Definition 6.7, let (A, P, Z, I_P) be a circumscriptive theory where I_P is $\left[P^1, \ldots, P^k\right]$ $(1 \leq k \leq n)$. Then (A, P, Z, I_P) denotes $Circum(\widetilde{\forall} A; I_P; Z)$ identified with $Circum(\widetilde{\forall} A; P^1 > \ldots > P^k; Z)$.

According to Theorem 4.5 and Theorem 4.6, it holds that, for any ground literal G of the language of A whose predicate symbol is not equality,

$Circum(\widetilde{\forall} A; P^1 > \ldots > P^k; Z) \models G$

iff G is true in all answer sets of either $ELP\ \Pi^{\beta}$ (in case of $k \neq 1$) constructed from $Circum(\widetilde{\forall} A; P^1 > \ldots > P^k; Z)$ by using Definition 4.3 or $ELP\ \Pi^{\alpha}$ (in case of $k = 1$) constructed from $Circum(\widetilde{\forall} A; P; Z)$ by using Definition 4.2.

Now it holds that

G is true in all answer sets of either $ELP\ \Pi^{\beta}$ (in case of $k \neq 1$) constructed from $Circum(\widetilde{\forall} A; P^1 > \ldots > P^k; Z)$ by using Definition 4.3 or $ELP\ \Pi^{\alpha}$ (in case of $k = 1$) constructed from $Circum(\widetilde{\forall} A; P; Z)$ by using Definition 4.2,

iff G is true in all answer sets of

$$\mathcal{E}(\left[P^1, \ldots, P^k\right]) \cup \mathcal{F}(E)$$

where E is a subset of Pri such that $md(E) = \left[P^1, \ldots, P^k\right] = I_P$,

because according to Definition 6.12, the conditional parts of all rules in $\mathcal{E}(\left[P^1, \ldots, P^k\right])$ are satisfied by a set of priority axioms from E (i.e. rules in $\mathcal{F}(E)$) satisfying a transitive relation as well as the 1-1 correspondence such that $md(E) = \left[P^1, \ldots, P^k\right]$, which has the same effect in the answer set semantics as that of a extended logic program: $ELP\Pi^{\beta}$ compiled from $Circum(\widetilde{\forall} A; P^1 > \ldots > P^k; Z)$ (or $ELP\ \Pi^{\alpha}$ in case of k=1).

On the other hand, it is obvious that each of $\mathcal{F}(E)$ and $\mathcal{F}(E) \cup \Omega$ has a unique answer set. So, let S_1 and S_2 be answer sets of $\mathcal{F}(E)$ and $\mathcal{F}(E) \cup \Omega$ respectively. Then it holds that for $\forall (p_i, p_j) \in P_c \times P_c$,

$$p_i > p_j \in E \quad \text{iff} \quad p_i > p_j \in S_1 \quad \text{iff} \quad p_i > p_j \in S_2,$$

because all priority axioms from E such that $md(E) = \left[P^1, \ldots, P^k\right] = I_P$ satisfy an irreflexive relation which is described by Ω.

Therefore, it holds that, for a subset E of Pri such that $md(E) = I_P$,

G is true in all answer sets of $\mathcal{E}(I_P) \cup \mathcal{F}(E)$

iff G is true in all answer sets of $\mathcal{E}(I_P) \cup \Omega \cup \mathcal{F}(E)$

Furthermore, it holds that, for a subset E of Pri such that $md(E) = I_P$,

G is true in all answer sets of $\mathcal{E}(I_P) \cup \Omega \cup \mathcal{F}(E)$

iff G is true in all answer sets of $\mathcal{E} \cup \Omega \cup \mathcal{F}(E)$

because any rule in $\mathcal{E}$ except rules in $\mathcal{E}(I_P)$ is not applicable since its conditional part is not satisfied by priority axioms from E such that $md(E) = I_P$.

Thus it holds that, for a subset E of Pri such that $md(E) = I_P$,

G is true in all answer sets of $\mathcal{E} \cup \Omega \cup \mathcal{F}(E)$

iff G has a skeptical explanation of E w.r.t. $\langle \mathcal{E} \cup \Omega, Pri \rangle$.

□

Chapter 7

Finding Dynamic Preferences as Hypotheses

7.1 Introduction

In Chapter 5, we have already presented our method which enables us to reason about dynamic preferences in the framework of circumscription and to compute consistently its meta-level and object-level reasoning by logic programming. In legal argumentation, however, it often occurs that due to the lack of dynamic preferences, a disputer cannot infer his desiring conclusion to defeat logically his opponent in the argumentation. As a result, the disputer wishes to prepare preferences as hypotheses with which a desired conclusion can be derived as well as to show them to his opponent as justification of his claim. The following example, a slightly modified version of the one [Gorden, 1993] shown in Section 5.1, illustrates the situation that dynamic preferences as hypotheses are required in the legal domain.

Assume a person who currently has possession of the ship. According to the Uniform Commercial Code(UCC) which is a state law, a security interest in goods may be perfected by taking possession of the collateral. However, there is a federal law called the Ship Mortgage Act(SMA) according to which a security interest in a ship may only be perfected by filing a financing statement. Such a statement has not been filed. However, she desires the judgment of the court that her security interest in a certain ship is perfected.

As is already shown in Section 5.1, the above knowledge is represented by the following axiom set A.

$$
\begin{aligned}
A: \quad & possession \wedge \neg ab1 \supset perfected, && (UCC)\\
& ship \wedge \neg fin\text{-}statement \wedge \neg ab2 \supset \neg perfected, && (SMA)\\
& possession,\\
& ship,\\
& \neg fin\text{-}statement.
\end{aligned}
$$

In addition, as is also mentioned in Section 5.1, there are two well-known legal principles, Lex Posterior and Lex Superior w.r.t. the UCC and the SMA for

resolving conflicts between them. They are represented by the axiom set M in Section 5.1 as follows:

$$\begin{aligned} M: \quad & more{-}recent(x,y) \supset y \prec x, && (LP(x,y)) \\ & fed{-}law(x) \wedge state{-}law(y) \supset y \prec x, && (LS(x,y)) \\ & more-recent(UCC, SMA), \\ & fed-law(SMA), \\ & state-law(UCC). \end{aligned}$$

If we apply our method of reasoning dynamic preferences presented in Chapter 5 to this example, the following results of parallel circumscription shown in Example 5.7 are derived from the given M and A,

$$Circum(\widetilde{\forall}A; ab1, ab2; perfected) \not\models perfected$$
$$Circum(\widetilde{\forall}A; ab1, ab2; perfected) \not\models \neg perfected.$$

This means that the phenomenon of so-called multiple extension problem arises as the result of the reasoning about dynamic preferences and we can no more infer $perfected$ which she desires under the given preferences M.

Now let us assume that there exists the following preference as a hypothesis in M,

$$\prec (LS1, LP1, lex1). \tag{7.1}$$

where $LS1$ and $LP1$ stand for $LS(UCC, SMA)$ and $LP(SMA, UCC)$ respectively and apply our method of reasoning dynamic preferences presented in Chapter 5 to this case again. Then in this case, the following result of prioritized circumscription are derived from $M \cup \{(7.1)\}$ and A based on Theorem 5.10:

$$Circum(\widetilde{\forall}A; ab1 > ab2; perfected) \models perfected$$

which means that by assuming the preference (7.1), we succeed to derive such a result that she desires.

Thus the aim of research of this chapter is to explore a method to find out such dynamic preferences as hypotheses, e.g. (7.1) in the above example, that make it possible to infer a desired result by assuming the hypotheses.

In Chapter 5, we present a method of reasoning dynamic preferences in the framework of circumscription. On the other hand, when a desired goal cannot be derived under the given knowledge-base due to the lack of dynamic preferences, adequate dynamic preferences as hypotheses are required to be found out so that we can make it possible to infer such a goal by assuming them as is shown in the above example. So, in this chapter, we present a method of finding dynamic preferences as hypotheses whose reasoning has the reverse direction of the reasoning presented in Chapter 5. As the significant characteristics of the method provided in this chapter, first it finds out the preferences conflicting each other based on the method

presented in Chapter 5 and secondly generates preferences as hypotheses to resolve the conflicts so that a desired object-level conclusion can be inferred successfully based on the reasoning of dynamic preferences presented in Chapter 5 by using such hypotheses.

In the subsequent section, we show that dynamic preferences as hypotheses mentioned above can be found out as a skeptical explanation of a desired object-level goal with respect to an abductive framework which are constructed based on the reasoning model provided in Chapter 5.

7.2 Finding Dynamic Preferences as Hypotheses

In Chapter 5, we present a method of reasoning dynamic preferences in the framework of circumscription. In this section, supposing such a situation that a desired goal can not be derived due to the lack of dynamic preferences, we show a method of finding dynamic preferences as hypotheses to make it possible to infer a desired goal by assuming the hypotheses, which is based on the reasoning model, i.e. framework, of dynamic preferences provided in Chapter 5 as well as the technique of abductive logic programming.

In Section 5.2.1, a priority axiom is introduced to represent the priority relation between minimized predicates. In the following Section 7.2.1, we suppose that a desired goal is given as a priority axiom which cannot be derived from the given preferences as facts. Then we show that dynamic preferences as hypotheses to derive the goal can be found out as a (credulous) explanation of the goal w.r.t. the abductive framework which is constructed based on the method of meta level reasoning from preferences to priorities shown in Section 5.2.1. In the next Section 7.2.2, a desired goal is given as a ground literal of the object domain. Then we show that dynamic preferences as hypotheses to derive the goal can be found out as a skeptical explanation of the object-level goal w.r.t. the abductive framework constructed by integrating the meta level abduction provides in Section 7.2.1 with the abductive reasoning provided in Chapter 6, that is, the method of finding priorities of circumscription policy.

In the following, the terminology and notation in Chapter 5 is used without any explanation. We suppose that a set M of preferences, the axiom set A, a tuple P of minimized predicates and a tuple Z of variable predicates occurring in A are also given in a way that is given in Chapter 5.

7.2.1 Finding Dynamic Preferences as Hypotheses in Meta Level Reasoning

According to the method of reasoning dynamic preferences presented in Chapter 5, the condition that there is no priority between rules r_1 and r_2 is as follows.

$$\Pi \cup \Gamma \not\models_{sm} r_1 > r_2 \; and \; \Pi \cup \Gamma \not\models_{sm} r_2 > r_1.$$

W.r.t. this condition, it holds that,

$$\Pi \cup \Gamma \not\models_{sm} r_1 > r_2 \textit{ and } \Pi \cup \Gamma \not\models_{sm} r_2 > r_1$$
if and only if either the following case 1 or case 2 is satisfied :
(*case* 1) $\Pi \cup \Gamma \not\models_{sm} V(r_1, r_2)$ *and* $\Pi \cup \Gamma \not\models_{sm} V(r_2, r_1)$
(*case* 2) $\Pi \cup \Gamma \models_{sm} V(r_1, r_2)$ *and* $\Pi \cup \Gamma \models_{sm} V(r_2, r_1)$.

In this chapter, we restrict ourselves to $\Pi \cup \Gamma$ which has no case 2. Under this restriction, the abductive framework to generate dynamic preferences as hypotheses in meta level reasoning is given as follows.

Definition 7.1 A set M of preferences, the axiom set A and a tuple P of minimized predicates occurring in A are given. Let Π be a normal logic program given by Definition 5.3, Γ be a stratified logic program constructed from M according to Definition 5.3 and $\prec_{ab}$ be a binary predicate symbol to express a preference as hypothesis which is distinguished from a preference as fact expressed by a predicate symbol $\prec$. We define Π' and Γ' as sets of rules constructed from Π and Γ by replacing predicate symbols $>, V$, $\prec$ occurring in every rule from Π by newly introduced similar predicate symbols $>', V'$, $\prec'$. Then an abductive framework to generate preferences as hypotheses is defined by $\langle \Pi_{ab},\ H_{\prec_{ab}} \rangle$ where

$$\Pi_{ab} \stackrel{def}{=} \Pi \cup \Gamma \cup I_c \cup \Pi' \cup \Gamma' \cup \{x \prec y \leftarrow\ x \prec_{ab} y\}$$
$$I_c \stackrel{def}{=} \{\leftarrow x >' y, not\ x > y\}$$
$$H_{\prec_{ab}} \stackrel{def}{=} \{x \prec_{ab} y | x \prec_{ab} y \in \mathcal{S}\}$$

and $\mathcal{S}$ is defined as a stable model of Π_1 as follows:

$$\Pi_1 \stackrel{def}{=} \Pi' \cup \Gamma' \cup \{[1], [2]\}.$$

In Π_1, [1] and [2] are given by the following rules:

$$[1\ ]\ z_1 \prec_{ab} z_2 \leftarrow \prec' (x, y, z_1), \prec' (y, x, z_2), not\ z_1 \prec' z_2, not\ z_2 \prec' z_1,$$
$$not\ V'(x, y), not\ V'(y, x),$$
$$[2\ ]\ x \prec_{ab} y \leftarrow dom0(x), dom0(y), not\ x \prec' y, not\ y \prec' x.$$

$dom0$ is a unary predicate symbol and $dom0(n)$ expresses that n is a name of clauses (rules) in A. From now on, we suppose that for every name of clauses in A, $dom0(n)$ is given in Γ as well as the following clause exists in Γ.

$$dom(x) \leftarrow\ dom0(x) \in \Gamma$$

Remarks: W.r.t. the definition of Π_{ab}, $\Pi \cup \Gamma$ plays a role to represent a knowlegde-base whose inference makes use of preferences as hypotheses along with preferences

as facts. On the other hand, the renamed $\Pi' \cup \Gamma'$ plays a role to represent a knowlegde-base whose inference uses only preferences as facts. $H_{\prec_{ab}}$ consists of preferences as hypotheses contained in a stable model of Π_1. They are generated from rules [1] and [2] of Π_1. If there is no priority between a rule x and a rule y, which means that none of $V(x, y)$ and $V(y, x)$ are derived only from $\Pi \cup \Gamma$ based on inference rules (4), (3) of Π (see Definition 5.3) due to the non-existence of preferences between meta-rules z_1 and z_2 in a inference rule (4) of Π (or between x and y in a inference rule (3)), then preferences as hypotheses such that $z_1 \prec_{ab} z_2$ (or $x \prec_{ab} y$) are generated by using [1] or [2] of Π_1. Since [1] and [2] of Π_1 corresponds to (4) and (3) of Π respectively, it becomes possible to derive $V(x, y)$ (or $V(y, x)$) by using such preference as hypothesis along with $\Pi \cup \Gamma$. The integrity constraint in I_c plays a role that priorities derived from $\Pi' \cup \Gamma'$ without using preferences as hypotheses are kept to be derived from $\Pi \cup \Gamma$ even if preferences as hypotheses are added to it.

Definition 7.2 Let $\mathcal{S}_D$ be a stable model of a normal logic program Π_D as follows:

$$\Pi_D \stackrel{def}{=} \Pi \cup \Gamma \cup \{d(x, y) \leftarrow V(x, y), V(y, x).\}$$

where d is a newly introduced predicate symbol. We define a set D as follows:

$$D \stackrel{def}{=} \{d(x, y) \mid d(x, y) \in \mathcal{S}_D, \quad x \neq y\}.$$

The following lemmas clarify the characteristics of hypotheses in $H_{\prec_{ab}}$. According to Lemma 7.4, any preference as hypothesis in $H_{\prec_{ab}}$ can be regarded as a minimal explanation to make it possible to derive such a priority that cannot be derived from $\Pi \cup \Gamma$ without it since if the preference assumed as a fact is added to $\Pi \cup \Gamma$, such a priority becomes to be derived. Lemma 7.5 denotes that, for any priority that cannot be derived from $\Pi \cup \Gamma$, there exists at least one preference as hypothesis in $H_{\prec_{ab}}$ which makes it possible to derive the priority from $\Pi \cup \Gamma$ along with the corresponding preference as fact.

Lemma 7.3 *It holds that $H_{\prec_{ab}} \subset U_{ab}$ where U_{ab} is a set defined as follows:*

$$U_{ab} \stackrel{def}{=} \{t_1 \prec_{ab} t_2 \mid \Pi \cup \Gamma \not\models_{sm} t_1 \prec t_2\},$$

in which t_1 and t_2 are elements of Herbrand universe of $\Pi \cup \Gamma$. That is,

$$If \ t_1 \prec_{ab} t_2 \in H_{\prec_{ab}} \ then \ \Pi \cup \Gamma \not\models_{sm} t_1 \prec t_2.$$

Lemma 7.4 *For any $t_1 \prec_{ab} t_2 \in H_{\prec_{ab}}$, there exists $p_1, p_2 \in P$ such that*

$$\Pi \cup \Gamma \not\models_{sm} p_1 > p_2 \ and \ \Pi \cup \Gamma \not\models_{sm} p_2 > p_1$$

as well as

$$\Pi \cup \Gamma \cup \{t_1 \prec t_2 \leftarrow\} \models_{sm} p_1 > p_2$$

.

Lemma 7.5 *For any pair of* $p_1, p_2 \in P$ *such that*

$$\Pi \cup \Gamma \not\models_{sm} p_1 > p_2 \text{ and } \Pi \cup \Gamma \not\models_{sm} p_2 > p_1,$$

there exists $t_1 \prec_{ab} t_2 \in H_{\prec_{ab}}$ *such that*

$$\Pi \cup \Gamma \cup \{t_1 \prec t_2 \leftarrow\} \models_{sm} p_1 > p_2$$

.

In legal argumentation, usually once some priority is derived as a conclusion, it is seldom denied after that. That is, once derived priorities are usually monotonically kept in legal argumentation. The following theorem takes into account such situations.

Theorem 7.6 *A set* M *of preferences, a set* A *of clauses and a set* P *of minimized predicates are given. Let* Π *and* Γ *be normal logic programs defined by Definition 5.3 where* Γ *is constructed from* M*. Let* $\langle \Pi_{ab},\ H_{\prec_{ab}} \rangle$ *be a consistent abductive framework constructed according to Definition 7.1 and* D *be a set defined by Definition 7.2. In this Theorem, we suppose* $D = \phi$ *(an empty set). In addition, we suppose that* p_1,p_2 *are minimized predicates (i.e.* $p_1, p_2 \in P$*) such that*

$$\Pi \cup \Gamma \not\models_{sm} p_1 > p_2, \qquad \Pi \cup \Gamma \not\models_{sm} p_2 > p_1,$$

and $p_1 > p_2$ *is a desired goal w.r.t.* $\Pi \cup \Gamma$*. Then if* E *is a (credulous) explanation of* $p_1 > p_2$ *w.r.t.* $\langle \Pi_{ab},\ H_{\prec_{ab}} \rangle$*, it holds that*

$$\Pi \cup \Gamma \cup \mathcal{F}(\{x \prec y \mid x \prec_{ab} y \in E\}) \models_{sm} p_1 > p_2,$$

as well as for any $q_1, q_2 \in P$ *such that* $\Pi \cup \Gamma \models_{sm} q_1 > q_2$*,*

$$\Pi \cup \Gamma \cup \mathcal{F}(\{x \prec y \mid x \prec_{ab} y \in E\}) \models_{sm} q_1 > q_2.$$

Theorem 7.6 means that even if a desired priority such as $p_1 > p_2$ is not inferred from the given preferences M, we can make it possible to derive it as a conclusion of preferences M as facts together with preferences E as hypotheses obtained as a credulous explanation with keeping the priorities derived from M.

It should be noticed that a credulous explanation E of $p_1 > p_2$ w.r.t. $\langle \Pi_{ab},\ H_{\prec_{ab}} \rangle$ in Theorem 7.6 can be computed by means of the ordinary abductive logic programming (see Theorem 6.5) based on the abductive framework $\langle T, H \rangle$ defined by Lemma 7.7 as follows.

Lemma 7.7 *Let* $domm$ *be a newly introduced binary predicate symbol and* $\langle T, H \rangle$ *be an abductive framework where* T *and* H *are given as follows:*

$$T \stackrel{def}{=} \Pi_{ab} \cup \Delta_{ab} \cup \{\leftarrow x \prec_{ab} y, not\ domm(x, y)\}$$
$$H \stackrel{def}{=} \{x \prec_{ab} y | x \text{ and } y \text{ are elements of Herbrand universe of } \Pi \cup \Gamma\}$$

where Δ_{ab} *is defined as follows:*

$$\Delta_{ab} \stackrel{def}{=} \{[3], [4]\}$$

and [3],[4] are rules of NLP as follows.

$$[3\ ]\ domm(z_1, z_2) \leftarrow \prec'(x, y, z_1), \prec'(y, x, z_2),\ not\ z_1 \prec' z_2, not\ z_2 \prec' z_1,$$
$$not\ V'(x, y), not\ V'(y, x).$$
$$[4\ ]\ domm(x, y) \leftarrow dom0(x), dom0(y),\ not\ x \prec' y, not\ y \prec' x.$$

Then w.r.t. Theorem 7.6, it holds that E is a credulous explanation of $p_1 > p_2$ *w.r.t.* $\langle \Pi_{ab},\ H_{\prec_{ab}} \rangle$ *if and only if E is a credulous explanation of* $p_1 > p_2$ *w.r.t.* $\langle T, H \rangle$.

Remarks: If both [3] and [4] in Δ_{ab} are used with an integrity constraint $\leftarrow x \prec_{ab} y, not\ domm(x, y)$, they plays the same role as both [1] and [2] of Π_1 in Definition 7.1.
For notational simplicity, we explain the following example not based on the framework $\langle T, H \rangle$ defined in Lemma 7.7, but $\langle \Pi_{ab},\ H_{\prec_{ab}} \rangle$ defined by Definition 7.1.

Example 7.8 Consider an example shown in Section 7.1. D becomes an empty set in this example. Though Γ constructed from M is already shown in example 5.7, according to the redefinition of Γ addressed in Definition 7.1,

$$dom0(UCC) \leftarrow,$$

$$dom0(SMA) \leftarrow,$$

$$dom(x) \leftarrow\ dom0(x).$$

should be added to Γ instead of

$$dom(UCC) \leftarrow,$$

$$dom(SMA) \leftarrow .$$

since UCC and SMA are names of clauses in the axiom set A.

Now, the result (1) derived in example 5.7 means that

$$ab1 > ab2 \notin S_{pri}, \qquad ab2 > ab1 \notin S_{pri},$$

or equivalently, $\Pi \cup \Gamma \not\models_{sm} ab1 > ab2$, $\Pi \cup \Gamma \not\models_{sm} ab2 > ab1$.
So, let $ab1 > ab2$ be a desired goal. According to Theorem 7.6, we compute a credulous explanation E of $ab1 > ab2$ w.r.t. $\langle \Pi_{ab},\ H_{\prec_{ab}} \rangle$. According to Definition 7.1, Π_{ab} is as follows:

$$\Pi_{ab} \stackrel{def}{=} \Pi \cup \Gamma \cup I_c \cup \Pi' \cup \Gamma' \cup \{x \prec y \leftarrow\ x \prec_{ab} y\}.$$

and a set $H_{\prec_{ab}}$ of hypotheses is obtained from a stable model of Π_1 as follows:

$$H_{\prec_{ab}} = \{LP1 \prec_{ab} LS1, \quad LS1 \prec_{ab} LP1\}$$

Then according to Theorem 6.5, $\Gamma(H_{\prec_{ab}})$ is obtained as follows:

$$\begin{aligned}\Gamma(H_{\prec_{ab}}) = \{&\neg(LP1 \prec_{ab} LS1) \leftarrow \ not\ (LP1 \prec_{ab} LS1),\\ &(LP1 \prec_{ab} LS1) \leftarrow \ not\ \neg(LP1 \prec_{ab} LS1),\\ &\neg(LS1 \prec_{ab} LP1) \leftarrow \ not\ (LS1 \prec_{ab} LP1),\\ &(LS1 \prec_{ab} LP1) \leftarrow \ not\ \neg(LS1 \prec_{ab} LP1)\}.\end{aligned}$$

as well as an abductive logic program is constructed from Π_{ab}, $\Gamma(H_{\prec_{ab}})$ and a goal $ab1 > ab2$ as follows:

$$\Pi_{ab} \cup \Gamma(H_{\prec_{ab}}) \cup \{\leftarrow \ not\ (ab1 > ab2)\}.$$

Therefore a credulous explanation E of $ab1 > ab2$ w.r.t. $\langle \Pi_{ab},\ H_{\prec_{ab}} \rangle$ can be obtained as follows:

$$E \stackrel{def}{=} S \cap H_{\prec_{ab}} = \{LS1 \prec_{ab} LP1\}$$

where S is an answer set of the above abductive logic program. Thus, there exists an answer set of $\Pi_{ab} \cup \mathcal{F}\ (E)$ in which $ab1 > ab2$ is true, which means that,

$$\Pi \cup \Gamma \cup \{LS1 \prec LP1 \leftarrow\} \models_{sm} ab1 > ab2. \qquad (7.2)$$

Therefore, according to Theorem 7.6, it becomes possible to infer a desired goal: $ab1 > ab2$ if a credulous explanation $\{LS1 \prec_{ab} LP1\}$ is assumed and added to Γ.

7.2.2 Finding Dynamic Preferences as Hypotheses for a Goal of Object Domain

Supposing a situation that a desired goal of the object domain is not derived due to the lack of dynamic preferences, we present a method of finding dynamic preferences as hypotheses to make it possible to derive the goal by using them. In this subsection, we show that such preferences as hypotheses can be found out as a skeptical explanation of the goal of the object domain w.r.t. the abductive framework which is constructed by integrating the meta-level abductive framework provided in the previous subsection with the object-level abductive framework provided in Chapter 6, i.e. the method of finding priorities of circumscription policy.

The following Theorem integrates Theorem 7.6 with Theorem 6.13. The reasoning shown in the following Theorem has the reverse direction of reasoning shown in Theorem 5.10.

Theorem 7.9 *A set M of preference information, a set A of clauses and tuples P, Z of n minimized predicates and variable predicates occurring in A are given. Let Π and Γ be normal logic programs constructed according to Definition 5.3, S_{pri} be a set of priority atoms derived from $\Pi \cup \Gamma$ according to Definition 5.6, $\mathcal{E}$ be an extended logic program constructed from A,P,Z according to Definition 5.9, Π_{ab} and $H_{\prec_{ab}}$ be sets defined in Definition 7.1 and D be a set defined in Definition 7.2. In this theorem, we suppose $D = \phi$. In addition, suppose that S_{pri} forms some k division $\left[P^1, \ldots, P^k\right]$ $(1 \leq k \leq n)$ of P such that $md(S_{pri}) = \left[P^1, \ldots, P^k\right]$ according to Definition 5.8 as well as for some ground literal G of the language of A whose predicate symbol is not equality, it holds that G is true in all answer sets of $\Pi \cup \Gamma \cup \mathcal{E}$ which means according to Theorem 5.10 as follows:*

$$Circum(\widetilde{\forall} A; P^1 > P^2 > \ldots > P^k; Z) \not\models G.$$

Then if E $(E \subset H_{\prec_{ab}})$ is a skeptical explanation of G w.r.t. an abductive framework $\langle \Pi_{ab} \cup \mathcal{E}, H_{\prec_{ab}} \rangle$, it holds that

$$Circum(\widetilde{\forall} A; P^{1'} > P^{2'} > \ldots > P^{k'}; Z) \models G$$

where $\left[P^{1'}, \ldots, P^{k'}\right]$ is a k' division of P which has a 1-1 correspondence such that $md(S'_{pri}) = \left[P^{1'}, \ldots, P^{k'}\right]$ to a set S'_{pri} $(S_{pri} \subset S'_{pri})$ of priority atoms derived from a stable model of $\Pi \cup \Gamma \cup \mathcal{F}(\{x \prec y | x \prec_{ab} y \in E\})$.

Theorem 7.9 means that even if such a desired goal G is not inferred from the given preferences M based on the method of reasoning dynamic preferences shown in Chapter 5, we can make it possible to derive G by using preferences as hypotheses E obtained as a skeptical explanation together with preferences as facts M.

It should be noted that a skeptical explanation E of G w.r.t. $\langle \Pi_{ab} \cup \mathcal{E}, H_{\prec_{ab}} \rangle$ can be computed by means of the ordinary abductive logic programming addressed in Section 6.2 based on the abductive framework $\langle T \cup \mathcal{E}, H \rangle$ defined by Lemma 7.10 as follows.

Lemma 7.10 *W.r.t. Theorem 7.9, E is a skeptical explanation of G w.r.t. $\langle \Pi_{ab} \cup \mathcal{E}, H_{\prec_{ab}} \rangle$ if and only if E is a skeptical explanation of G w.r.t. $\langle T \cup \mathcal{E}, H \rangle$ where T and H are given in Lemma 7.4.*

We explain Theorem 7.9 by using an example used in Section 7.1 as follows.

Example 7.11 Consider an example in Section 7.1 again where it holds that $D = \phi$ and $S_{pri} = \phi$. As is discussed in Section 7.1, the following results of parallel circumscription are derived from the given M and A,

$$Circum(\widetilde{\forall} A; ab1, ab2; Z) \not\models perfected.$$

Therefore a desired result $perfected$ is not derived. So, according to Theorem 7.9, in order to find preferences as hypotheses which make it possible to infer

$perfected$, let us compute a skeptical explanation E of $perfected$ w.r.t. an abductive framework $\langle \Pi_{ab} \cup \mathcal{E},\ H_{\prec_{ab}} \rangle$. $H_{\prec_{ab}}$ and $\Gamma(H_{\prec_{ab}})$ are already shown in the previous Example 7.8, and a set of $\mathcal{E}$ is explained in Example 5.11. In the following, we show how a set of such skeptical explanations are computed by using bottom-up procedure provided in Section 6.2.

Step 1: From $perfected$ as a goal and the abductive framework: $\langle \Pi_{ab} \cup \mathcal{E},\ H_{\prec_{ab}} \rangle$, an extended logic program is constructed as follows:

$$\Pi_{ab} \cup \mathcal{E} \cup \Gamma(H_{\prec_{ab}}) \cup \{\leftarrow not\ perfected\},$$

which has two answer sets. From each answer set $S1$, a credulous explanation $E1 = S1 \cap H_{\prec_{ab}}$ of $perfected$ w.r.t. the above abductive framework is computed. Then a set $C1$ which consists of all of $E1$ is obtained as follows:

$$C1 = \{\{LS1 \prec_{ab} LP1\}, \phi\}.$$

Step 2: From $perfected$ as a goal and the abductive framework: $\langle \Pi_{ab} \cup \mathcal{E},\ H_{\prec_{ab}} \rangle$, an extended logic program is constructed as follows:

$$\Pi_{ab} \cup \mathcal{E} \cup \{G' \leftarrow not\ perfected\ \} \cup \Gamma(H_{\prec_{ab}}) \cup \{\leftarrow not\ G'\},$$

which has also two answer sets. From each answer set $S2$, $E2 = S2 \cap H_{\prec_{ab}}$ is computed. Then a set $C2$ which consists of all of $E2$ is obtained as follows: $C2 = \{\{LP1 \prec_{ab} LS1\}, \phi\}$.

Step 3: $\phi \in C1$ and $\phi \in C2$, then $\phi \notin C3$. Thus $C3 = \{\{LS1 \prec_{ab} LP1\}\}$.

Therefore $\{LS1 \prec_{ab} LP1\}$ is obtained as only one skeptical explanation w.r.t. $\langle \Pi_{ab} \cup \mathcal{E},\ H_{\prec_{ab}} \rangle$. According to Definition 5.6 and the result (7.2) obtained in Example 7.8, S'_{pri} is computed from a stable model of $\Pi \cup \Gamma \cup \mathcal{F}(\{LS1 \prec LP1\})$ as follows:

$$S'_{pri} = \{ab1 > ab2\}$$

such that $md(S'_{pri}) = [[ab1], [ab2]]$, which denotes prioritized circumscription as follows: $Circum(\widetilde{\forall} A; ab1 > ab2; perfected)$.

On the other hand, let us assume $\{LS1 \prec_{ab} LP1\}$ which we just find in the above computation, and apply our method of reasoning dynamic preferences, i.e. Theorem 5.10. Thus $\Pi \cup \Gamma \cup \mathcal{F}(\{LS1 \prec LP1\}) \cup \mathcal{E}$ has only one answer set in which $perfected$ is true, which leads to

$$Circum(\widetilde{\forall} A; ab1 > ab2; perfected) \models perfected$$

based on Theorem 5.10.

7.3 Examples

Example 7.12 Consider an example in Section 7.1, but here let Γ given in Example 7.8 be slightly changed into Γ_3 as follows.

$$\Gamma_3 \stackrel{def}{=} \Gamma \cup \{\prec (LP1, LS1, Lex1) \leftarrow,\ \prec (LS1, LP1, Lex2) \leftarrow\ \}$$

In this case, both the UCC and the SMA are the object-level rules, both $LP1$ and $LS1$ are one-level meta rules (i.e. preferences) than the object-level ones, and both $Lex1$ and $Lex2$ are two-level meta rules than the object-level ones.

Let N_3 be a a stable model of $\Pi \cup \Gamma_3$. Then according to Definition 5.6, the following result is derived:

$$S_{pri} \stackrel{def}{=} N_3 \cap \{ab1 > ab2,\ ab2 > ab1\} = \phi$$

$$i.e. \qquad \Pi \cup \Gamma_3 \not\models_{sm} ab1 > ab2, \quad \Pi \cup \Gamma_3 \not\models_{sm} ab2 > ab1.$$

On the other hand, since based on Theorem 5.10, $\Pi \cup \Gamma_3 \cup \mathcal{E}$ has two answer sets where $perfected$ is true in one answer set, and $\neg perfected$ is true in another one, the following result is derived:

$$Circum(\widetilde{\forall} A; ab1, ab2; perfected) \not\models perfected.$$

First let $ab1 > ab2$ be a desired goal. In a similar way shown in Example 7.8, an abductive framework $\langle \Pi_{ab},\ H_{\prec_{ab}} \rangle$ is constructed according to Theorem 7.6, where according to Definition 7.1,

$$\Pi_{ab} \stackrel{def}{=} \Pi \cup \Gamma_3 \cup I_c \cup \Pi' \cup \Gamma' \cup \cup \{x \prec y \leftarrow\ x \prec_{ab} y\}$$

and $H_{\prec_{ab}}$ is obtained as a set of ground literals of $\prec_{ab}$ about a pair of $Lex1, Lex2$, a pair of $LS1, Lex1$ and a pair of $LP1, Lex2$. Then based on Theorem 7.6, we can obtain seven credulous explanations of $ab1 > ab2$ w.r.t. the abductive framework $\langle \Pi_{ab},\ H_{\prec_{ab}} \rangle$ mentioned above as follows:

(1) $\{Lex2 \prec_{ab} Lex1\}$

(2) $\{LS1 \prec_{ab} Lex1\}$

(3) $\{Lex2 \prec_{ab} LP1\}$

(4) $\{LS1 \prec_{ab} Lex1, Lex2 \prec_{ab} Lex1\}$

(5) $\{Lex2 \prec_{ab} LP1, lex2 \prec_{ab} Lex1\}$

(6) $\{Lex2 \prec_{ab} LP1, LS1 \prec_{ab} Lex1\}$

(7) $\{Lex2 \prec_{ab} Lex1, Lex2 \prec_{ab} LP1, LS1 \prec_{ab} Lex1\}$

(1)$\sim$(3) are minimal explanations, each of (4)$\sim$(6) is an union of two minimal explanations which are different each other, and (7) is an union of all minimal explanations. W.r.t. (1), $Lex2 \prec_{ab} Lex1$ expresses a meta-preference as hypothesis between equi-level preferences such as $Lex1$ and $Lex2$. W.r.t. (2) and (3), they express meta-preferences as as hypotheses between different-level preferences such as a pair of $LS1$ and $Lex1$ and a pair of $LP1$ and $Lex2$.

Besides let E be one of credulous explanations: (1)$\sim$(7). Then as is addressed in Theorem 7.6, it hold that for such any E,

$$\Pi \cup \Gamma_3 \cup \{x \prec y | x \prec_{ab} y \in E\} \cup \mathcal{E} \models_{ab} ab1 > ab2$$

such that $md(ab1 > ab2) = [[ab1], [ab2]]$.

On the other hand, let $perfected$ be a desired goal. In a similar way shown in Example 7.8, an abductive framework $\langle \Pi_{ab} \cup \mathcal{E}, H_{\prec_{ab}} \rangle$ is constructed according to Theorem 7.9 where Π_{ab} and $H_{\prec_{ab}}$ are mentioned above and $\mathcal{E}$ is the same one shown in Example 7.8. Then based on Theorem 7.9, seven skeptical explanations (1)$\sim$(7) of $perfected$ w.r.t. the abductive framework $\langle \Pi_{ab} \cup \mathcal{E}, H_{\prec_{ab}} \rangle$, are computed by using the bottom-up or top-down procedures provided in section 6.2.

Besides let E be one of skeptical explanations: (1)$\sim$(7). Then as is addressed in Theorem 7.9, it holds that for such any E, $\Pi \cup \Gamma_3 \cup \mathcal{F}(\{x \prec y | x \prec_{ab} y \in E\}) \cup \mathcal{E}$ has unique answer set in which $perfected$ is true, which leads to

$$Circum(\widetilde{\forall} A; ab1 > ab2; perfected) \models perfected.$$

based on Theorem 5.10.

7.4 Summary

In legal argumentation, it often occurs that due to the lack of dynamic preferences, a disputer cannot infer his desiring conclusion to defeat logically his opponent in the argumentation. As a result, dynamic preferences as hypotheses are required to be found out so that he can derive the desired conclusion by assuming the hypotheses and win the argumentation.

However, so far, no method has been proposed to find such dynamic preferences as hypotheses. As we have already shown the method to reason dynamic preferences in the framework of circumscription in Chapter 5, the method presented in this chapter has the reverse direction of the reasoning in Chapter 5 and enables us to find dynamic preferences as hypotheses as explanations of the desired goal w.r.t. the abductive framework constructed from the reasoning model presented in Chapter 5. In our method presented here, dynamic preferences as hypotheses are computed by means of abductive logic programming so that it can give a way to support the legal argumentation by a computer.

In case that a desired conclusion is given by a goal of the object domain, our method, however, has the same problem of the computational complexity as is discussed in Section 6.4.

In this chapter, we suppose $D = \phi$, which just corresponds to the condition such that the Lifschitz's circumscriptive theory [Lifschitz, 1989] never becomes *contradiction*. How to find dynamic preferences as hypotheses to derive a desired goal in case of $D \neq \phi$ is our future problem.

7.5 Proofs

Firstly we prepare the definition of a stratified logic program with integrity constraints. Next, we show the proofs of Theorems 7.6 and 7.9.

Definition 7.13 (A stratified logic program with integrity constraints)
A normal logic program Π with integrity constraints is called a *stratified logic program with integrity constraints* if it is possible to decompose the set P of all predicates of the program Π into disjoint sets

$$P^1 ; \ldots ; \ P^k$$

called *strata*, so that for every rule of the form where A_i, B_j, C are atoms:

$$C \leftarrow A_1, \ldots, A_m, notB_1, \ldots, notB_n, \tag{7.1}$$

or of the form called an integrity constraint where $\bot$ is a ground atom which has a special predicate symbol not occurring in any of A_i, B_j, C:

$$\bot \leftarrow A_1, \ldots, A_m, notB_1, \ldots, notB_n, \tag{7.2}$$

if each predicate of C and $\bot$ belongs to some stratum, say P^ℓ $(1 \leq \ell \leq k)$, we have that:

(a) all predicates of A_i belong to $P_1, \ldots, P_\ell$;

(b) all predicates of B_j belong to $P_1, \ldots, P_{\ell-1}$.

Any particular decomposition $P^1; \ldots ; P^k$ of P satisfying the above conditions is called a *stratification* of Π. For $i < j$, we call that a stratum P^i is lower than a stratum P^j.

Remarks: As the form of an integrity constraints, the following which has the empty head is often used instead of (7.2)

$$\leftarrow A_1, \ldots, A_m, notB_1, \ldots, notB_n, \tag{7.3}$$

Definition 7.14 *Let Π be a stratified logic program with integrity constraints which has a stratification P^1 ;...; P^k, and M_k be a unique perfect model (i.e. an iterated fixed point model) of Π. A canonical model for a stratified logic program Π with integrity constraints is M_k iff $\bot \notin M_k$. We say that Π is consistent if Π has a unique canonical model.*

Theorem 7.15 *If Π is a consistent stratified logic program with integrity constraints, then its unique stable model is identical to its canonical model (i.e. a perfect model, or an iterated fixed point model).*

Proof of Theorem 7.6

Let E be a (credulous) explanation of $p_1 > p_2$ w.r.t. $\langle \Pi_{ab}, \ H_{\prec_{ab}} \rangle$. Then $p_1 > p_2$ is true in some answer set (stable model) M of $\Pi_{ab} \cup \mathcal{F}(E)$ where $E \subset H_{\prec_{ab}}$.
According to Definition 7.1, it holds that,

$$\begin{aligned} &\Pi \cup \Gamma \cup \{\leftarrow x >^{'} y, \ not \ x > y\} \cup \Pi^{'} \cup \Gamma^{'} \\ &\qquad \cup \{x \prec y \leftarrow \ x \prec_{ab} y\} \cup \{x \prec_{ab} y \leftarrow | x \prec_{ab} y \in E\} \models_{sm} p_1 > p_2, \end{aligned}$$

which is equivalently transformed as follows.

$$\Pi \cup \Gamma \cup \{x \prec y \leftarrow | x \prec_{ab} y \in E\} \cup \{\leftarrow x >^{'} y, \ not \ x > y\} \cup \Pi^{'} \cup \Gamma^{'} \models_{sm} p_1 > p_2. \ (1)$$

W.r.t. $p_1 > p_2$, since it is supposed that $\Pi \cup \Gamma \not\models_{sm} p_1 > p_2$ which means that $\Pi' \cup \Gamma' \not\models_{sm} p_1 >' p_2$, it holds that,

$\Pi \cup \Gamma \cup \{x \prec y \leftarrow | x \prec_{ab} y \in E\} \cup \{\leftarrow x >' y,\ not\ x > y\} \cup \Pi' \cup \Gamma' \models_{sm} p_1 > p_2$
iff $\Pi \cup \Gamma \cup \{x \prec y \leftarrow | x \prec_{ab} y \in E\} \models_{sm} p_1 > p_2$

On the other hand, w.r.t. a goal $q_1 > q_2$ such that $\Pi \cup \Gamma \models_{sm} q_1 > q_2$, let us suppose that

$$\Pi \cup \Gamma \cup \mathcal{F}(\{x \prec y \mid x \prec_{ab} y \in E\}) \not\models_{sm} q_1 > q_2. \tag{2}$$

Now, $\Pi \cup \Gamma \models_{sm} q_1 > q_2$ means as follows:

$$\Pi' \cup \Gamma' \models q_1 >' q_2 \tag{3}$$

Since both of $\Pi' \cup \Gamma'$ and $\Pi \cup \Gamma \cup \mathcal{F}(\{x \prec y \mid x \prec_{ab} y \in E\})$ are stratified logic programs, each of them has a unique perfect model. Since a set of predicate symbols of $\Pi' \cup \Gamma'$ and that of $\Pi \cup \Gamma \cup \mathcal{F}(\{x \prec y \mid x \prec_{ab} y \in E\})$ are disjoint each other, it is obvious that a stratified logic program with integrity constraints such that $\Pi \cup \Gamma \cup \{x \prec y \leftarrow | x \prec_{ab} y \in E\} \cup \{\leftarrow x >' y,\ not\ x > y\} \cup \Pi' \cup \Gamma'$ has no canonical model due to (2) and (3), which contradicts to (1). Thus it holds that $\Pi \cup \Gamma \cup \mathcal{F}(\{x \prec y \mid x \prec_{ab} y \in E\}) \models_{sm} q_1 > q_2$ for any $q_1 > q_2$ such that $\Pi \cup \Gamma \models_{sm} q_1 > q_2$.

□

Proof of Theorem 7.9

Let E ($E \subset H_{\prec_{ab}}$) be a skeptical explanation of G w.r.t. an abductive framework $\langle \Pi_{ab} \cup \mathcal{E}, H_{\prec_{ab}} \rangle$. Then according to Definition 6.2, G is true in all answer sets of Π_{ALP} which is defined as follows.

$\Pi_{ALP} \stackrel{def}{=} \Pi \cup \Gamma \cup \{x \prec y \leftarrow | x \prec_{ab} y \in E\} \cup \{\leftarrow x >' y,\ not\ x > y\} \cup \Pi' \cup \Gamma' \cup \mathcal{E}$

Let τ be a subset of Π_{ALP} as follows.

$$\tau \stackrel{def}{=} \Pi \cup \Gamma \cup \{x \prec y \leftarrow | x \prec_{ab} y \in E\} \cup \{\leftarrow x >' y,\ not\ x > y\} \cup \Pi' \cup \Gamma'$$

Since τ is a stratified logic program with integrity constraints, it has either a unique canonical (perfect) model or no model.

So, suppose that τ has no model. Then according to the definition of $\mathcal{E}$ (see Definition 5.9), the body of every rule of $\mathcal{E}$ becomes false when the answer set of Π_{ALP} is evaluated. Therefore, there exists no answer set of Π_{ALP}, which contradicts that G is true in all answer sets of Π_{ALP}. Thus τ has a unique canonical (perfect) model. So, let S'_{pri} be $M \cap Pri$ where M is a unique canonical model of τ.
Now suppose again that there exists no k' division I'_P of P such that $md(S'_{pri}) = \left[P^{1'}, \ldots, P^{k'}\right] = I'_P$. Then the body of every rule of $\mathcal{E}$ becomes false when Π_{ALP} is evaluated. Therefore in such a case, the answer sets of Π_{ALP} become coincident with the stable models of τ where the literal G never occur, which contradicts again that G is true in all answer sets of Π_{ALP}.

Thus there must exist some k' division I'_P of P such that

$$md(S'_{pri}) = I'_P = \left[P^{1'}, \ldots, P^{k'}\right],$$

where we can easily show $S_{pri} \subset S'_{pri}$ due to an integrity constraint in I_C.

Therefore

G is true in all answer sets of Π_{ALP}

iff G is true in all answer sets of $\mathcal{E}(I'_P) \cup S'_{pri}$ for some k' division I'_P of P.

iff $Circum(\widetilde{\forall} A; P^{1'} > P^{2'} > \ldots > P^{k'}; Z) \models G$.

□

Chapter 8

Conclusion

In this thesis, we have presented several methods to realize a computer-aided intelligent system which can treat various human commonsense reasoning. Every method presented in this thesis concerns *circumscription* which is one of the most powerful formalisms of nonmonotonic reasoning (along with *abduction* in some methods) as well as *logic programming*. The subjects handled in this thesis are divided into two, that is, one is to explore methods of computing circumscription, and another is to explore their applications for the practical use in the real world.

The contributions of this thesis are the following.

1. We present two methods of computing circumscription by means of compiling circumscription into a logic program as follows.

   (a) The first one is an extension of Gelfond and Lifschitz's method [1988] in a sense that our method extends the applicable class of their method. With keeping the efficiency of their method, some class of prioritized circumscription can be compiled into a stratified logic program by our method.

   (b) The second one, i.e. Wakaki and Satoh's method, is based on our original idea, which compiles a wide class of parallel circumscription as well as prioritized circumscription into extended logic programs (ELP). In addition, our approach can give an extension of Sakama and Inoue's method [1995] which compiles parallel circumscription into EDP, so that it may become applicable to prioritized circumscription.

2. We present a method of reasoning dynamic preferences in the framework of circumscription which can be computed by logic programming based on Wakaki and Satoh's method.

3. We provide the bottom-up and top-down abductive procedures to compute skeptical explanations.

4. We present methods to find priorities as well as dynamic preferences as hypotheses.

(a) We present a method of finding priorities of circumscription policy to derive a desired goal as a skeptical explanation in abduction.

(b) We propose a method of finding dynamic preferences as hypotheses to derive a desired goal for a disputer in legal argumentation.

W.r.t. the contribution 1, Wakaki and Satoh's method enables us to compile the largest class of prioritized circumscription into logic programs of all methods proposed so far. This method is so powerful and applicable that all methods addressed in the contributions 2 and 4 are based on this method. We do not know any research or approach which succeeds to apply circumscription to fairly practical problems of commonsense reasoning considered in this thesis.

W.r.t. the contribution 2, Brewka proposed a method to treat dynamic preferences and compute them by logic programming in the framework of a semi-normal default theory, but no one does in the framework of circumscription. When the commonsense reasoning is considered by circumscription in the object level and it occurs that dynamic preferences should be reasoned, our method presented in the contribution 2 becomes useful.

W.r.t. the contribution 3, any method to compute a skeptical explanation has not been proposed as well as few applications that need skeptical explanations have been reported so far. Also w.r.t. the contribution 4, any method of finding priorities as well as dynamic preferences as hypotheses has not been proposed so far.

As a future research, we think that the following are needed.

1. **Research for the efficient implementation:**

   In order to use Wakaki and Satoh's method for the practical purposes, we have to implement the compilation process from circumscription into extended logic programs. However, the characteristic clauses should be computed in the compilation process. So, as is already addressed the problem of computational complexity in section 4.3, we must investigate about how to compute the characteristic clauses efficiently as a future technical problem.

2. **Uniform mathematical approach to treat dynamic preferences:**

   In order to reason dynamic preferences addressed in Section 5 as well as find them as hypotheses addressed in Section 7, we prepare the meta-level inference rules from preferences to priorities expressed by a normal logic program with assuming that the object-level nonmonotonic reasoning is formalized by the framework of circumscription whose form is given by a second-order predicate logic. However, it is impossible to describe the reasoning of dynamic preferences by the predicate logic since such preferences should be expressed by the meta-level statements, meta-meta level statements, meta-meta-meta level statements and so no. Thus it is desirable to handle the meta-level reasoning of dynamic preferences as well as the object-level nonmonotonic

reasoning based on some uniform mathematical foundation. This problem is left as our future work.

3. **Learning default rules and preferences:**

   In daily life, we learn default rules and preferences empirically. As is addressed in [Satoh, 1992], when we see one thousand birds in which 999 birds fly, but as only one penguin does not fly, we consider that "a bird normally flies" as a default, i.e. commonsense knowledge. So, in order to realize a computer-aided intelligent system which can treat various human commonsense reasoning, we had better to support such mechanisms to learn or acquire default rules and preferences as human does. The techniques of machine learning and inductive logic programming may be helpful for this purpose.

Bibliography

[Apt,et. 1987] Apt, K.R., Blair, H.A. and Walker, A., Towards a Theory of Declarative Knowledge, in: J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, pages 89-148, 1987.

[Baker and Ginsberg, 1989] Baker, A. B. and Ginsberg, M. L., A Theorem Prover for Prioritized Circumscription, *Proc. of IJCAI-89*, pages 463-467,1 989.

[Brewka, 1994] Brewka, G., Reasoning about priorities in default logic, *Proc. AAAI-94*, pages 940-945, 1994.

[Brewka, 1996] Brewka, G., Well-founded Semantics for Extended Logic Programs with Dynamic Preferences, *Journal of Artificial Intelligence Research 4*, pages 19-36, 1996.

[Delgrande, 1997] Delgrande, J. P. and Schaub T. H., Compiling Reasoning with and about Preferences into Default Logic, *Proc. IJCAI-97*, pages 168-174, 1997.

[Dung, 1991] Dung, P. M., Negation as Hypothesis: an abductive foundation for logic programming, *Proc. 8th International Conference on Logic Programming*, pages3-17, 1991.

[de Kleer and Konolige, 1989] de Kleer, J. and Konolige, K., Eliminating the Fixed Predicates from a Circumscription, *Artificial Intelligence* **39**, pages 391-398, 1989.

[Etherington, 1987a] Etherington, D. W., Relating Default Logic and Circumscription, *Proc. IJCAI-87*, pages 489-494, 1987.

[Etherington, 1987b] Etherington, D. W., A Semantics for Default Logic, *Proc. IJCAI-87*, pages 495-498, 1987.

[Etherington, 1988] Etherington, D. W., Reasoning with Incomplete Information, *Research Notes in Artificial Intelligence*, Pitman, London, 1988.

[Gelfond and Lifschitz, 1988a] Gelfond, M. and Lifschitz, V., Compiling Circumscriptive Theories into Logic Programs, *Proc. AAA1-88*, pages 455-459. Extended version in: *Proc. 2nd Int. Workshop on Nonmonotonic Reasoning*, LNAI 346, pages 74-99, 1988.

[Gelfond and Lifschitz, 1988b] Gelfond, M. and Lifschitz, V. The Stable Model Semantics for Logic Programming, *Proc. LP-88*, pages 1070-1080, 1988.

[Gelfond and Lifschitz, 1991] Gelfond, M. and Lifschitz, V. Classical Negation in Logic Programs and Disjunctive Databases, *New Generation Computing 9*, pages 365-385, 1991.

[Ginsberg, 1989] Ginsberg, M. L., A Circumscriptive Theorem Prover, *Artificial Intelligence* **39**, pages 209-230, 1989.

[Gorden, 1993] Gorden, T. F., The Pleadings Game: An Artifical Intelligence Model of Procedural Justice. *Ph.D. thesis, TU Darmstadt*, 1993.

[Hanks and McDermott, 1987] Hanks, S. and McDermott, D., Nonmonotonic logic and temporal projection, *Artificial Intelligence* **33**, pages 379-412, 1987.

[Inoue, 1992a] Inoue, K., Linear Resolution for Consequence Finding, *Artificial Intelligence* **56**, pages 301-353, 1992.

[Inoue, 1992b] Inoue, K., Studies on Abductive and Nonmonotonic Reasoning, *Ph.D. Dissertation, Kyoto University*, Kyoto, Japan, 1992.

[Inoue, 1994] Inoue, K., Hypothetical Reasoning in Logic Programs, *Journal of Logic Programming* 18(3), pages 191-227, 1994.

[Inoue and Heft, 1990] Inoue, K. and Heft, N., On Theorem Provers for Circumscription, *Proc. 8th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pages 212-219, 1990.

[Inoue and Sakama, 1994] Inoue, K. and Sakama, C. On Positive Occurrences of Negation as Failure, *Proc. KR-94*, pages 293-304, 1994.

[Inoue and Sakama, 1998] Inoue, K. and Sakama, C. Negation as Failure in the Head, *Journal of Logic Programming* 35(1), pages 39-78, 1998.

[Junker, 1997] Junker, U., A Cumulative-Model Semantics for Dynamic Preferences on Assumptions, *Proc. IJCAI-97*, pages 162-167, 1997.

[Kakas and Mancarella, 1990] Kakas, A. C. and Mancarella, P., Generalized Stable Models: A Semantics for Abduction, *Proc. the Ninth European Conference on Artificial Intelligence*, pages 385-391, 1990.

[Eshghi and Kowalski, 1989] Eshghi, K. and Kowalski, R. A., Abduction Compared with Negation by Failure, *Proc. ICLP-89*, pages 234-254, 1989.

[Lifschitz, 1985] Lifschitz, V., Computing Circumscription, *Proc. IJCAI-85*, pages 121-127, 1985.

[Lifschitz, 1987a] Lifschitz, V., On the Declarative Semantics of Logic Programs with Negation, in: J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, pages 177-192, 1987.

[Lifschitz, 1987b] Lifschitz, V., Circumscriptive Theories: A Logic-Based Framework for Knowledge Representation (Preliminary Report), *Proc. AAAI-87*, pages 364-368, 1985.

[Lifschitz, 1989] Lifschitz, V., Circumscriptive Theories: A Logic-Based Framework for Knowledge Representation, in: R. Thomason (ed.), *Philosophical Logic and Artificial Intelligence*, Kluwer, pages 109-159, 1989.

[McCarthy, 1980] McCarthy, J., Circumscription - a Form of Non-monotonic Reasoning, *Artificial Intelligence* **13**, pages 27-39, 1980.

[McCarthy, 1986] McCarthy, J. Applications of Circumscription to Formalizing Commonsense Knowledge. *Artificial Intelligence* **28**, pages 89-116, 1986.

[Peirce, 1932] Peirce, C. S., Elements of Logic, In *Collected papers of Charles Sanders Peirce*, Vol.2, (C. Hartshorn et al, eds.), Harvard University Press, MA, 1932.

[Poole, 1988] Poole, D., A Logical framework for default reasoning, *Artificial Intelligence* **36**, pages 27-47, 1988.

[Prakken and Sartor, 1995] Prakken, H. and Sartor, G., On the relation between legal language and legal argument: Assumptions, applicability and dynamic priorities, *Proc. Int. Conference in AI and Law*, 1995.

[Przymusinski, 1987] Przymusinski, T., On the Declarative Semantics of Deductive Databases and Logic Programs, in: J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, pages 193-216, 1987.

[Przymusinski, 1988] Przymusinski, T., On the Declarative and Procedural Semantics of Logic Programs, Preprint, University of Texas at EL Paso, 1988.

[Przymusinski, 1989] Przymusinski, T., An algorithm to compute circumscription, *Artificial Intelligence* **38**, pages 49-73, 1989.

[Reiter, 1978] Reiter, R., On closed world data bases, in: H. Gallaire and J. Minker (ed.), *Logic and Data Bases*, Plenum Press, pages 55-76, 1978.

[Reiter, 1980] Reiter, R., A Logic for default reasoning, *Artificial Intelligence* **13**, pages 81-132, 1980.

[Reiter, 1984] Reiter, R., Towards a logical reconstruction of relational database theory, In: *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages*, pages 191-238, Springer-Verlag, New York, 1984.

[Reiter and Criscuolo, 1981] Reiter, R. and Criscuolo, G., On Interacting Defaults, *Proc. IJCAI-81*, 1981.

[Sakama and Inoue, 1995] Sakama, C. and Inoue, K., Embedding Circumscriptive Theories in General Disjunctive Programs, *Proc. 3rd International Conference on Logic Programming and Nonmonotonic Reasoning*, 1995, LNAI 928, pages 344-357, Springer.

[Sakama and Inoue, 1996] Sakama, C. and Inoue, K., Representing Priorities in Logic Programs, *Proc. Joint International Conference and Symposium on Logic Programming*, pages 82-96, 1996.

[Satoh, 1988] Satoh, K., Approaches for Yale Shooting Problem, *Journal of Japanese Society for Artificial Intelligence*, Vol. 3, No. 2, pages 132-138, 1988 (in Japanese).

[Satoh, 1990] Satoh, K., Formalizing Soft Constraints by Interpretation Ordering, *Proc. of ECAI-90*, pages 585-590, 1990.

[Satoh, 1992] Satoh, K., A Logical Formalization of Preference-based Reasoning by Interpretation Ordering, *Ph.D. Dissertation, University of Tokyo*, Tokyo, Japan, 1992.

[Satoh and Iwayama, 1991] Satoh, K. and Iwayama, N., Computing Abduction by Using the TMS, *Proc. 8th International Conference on Logic Programming*, pages 505-518, 1991.

[Satoh and Iwayama, 1992] Satoh, K. and Iwayama, N., A Query Evaluation Method for Abductive Logic Programming, *Proc. Joint International Conference and Symposium on Logic Programming*, pages 671-685, 1992.

[Van Gelder, 1988] Van Gelder, A., Negation as Failure Using Tight Derivations for General logic programs, in: J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, pages 149-176, 1987.

[Wakaki and Satoh, 1995] Wakaki, T and Satoh, K., Computing Prioritized Circumscription by Logic Programming, *Proc. 12th International Conference on Logic Programming*, pages 283-297, 1995.

[Wakaki and Satoh, 1996] Wakaki, T and Satoh, K., Computing Prioritized Circumscription via Compilation into Logic Programs, *Journal of Japanese Society for Artificial Intelligence*, Vol.11, No.5, pp.786–794, 1996 (in Japanese).

[Wakaki and Satoh, 1997] Wakaki, T and Satoh, K., Compiling Prioritized Circumscription into Extended Logic Programs, *Proc. IJCAI-97*, pages 182-187, 1997.

[Wakaki, Satoh and Nitta, 1997] Wakaki, T, Satoh, K. and Nitta, K., Reasoning about Dynamic Preferences in Circumscriptive Theory by Logic Programming, *Int. Journal on Advanced Computational Intelligence*, Vol.1, No.2, pages 121-129, 1997.

[Wakaki et al., 1998] Wakaki, T., Satoh, K., Nitta, K. and Sakurai, S., Finding Priorities of Circumscription Policy as a Skeptical Explanation in Abduction, *Journal of IEICE Transactions on Information and Systems*, Vol.E-81D, No.10, pages 1111-1119, 1998.

[Wakaki and Satoh, 1999] Wakaki, T and Satoh, K., Computing Prioritized Circumscription by Compiling into Extended Logic Programs based on the Semantic Relationship, to appear in *Journal of Japanese Society for Artificial Intelligence*, 1999 (in Japanese).