Compiling Prioritized Circumscription into Extended Logic Programs

Toshiko Wakaki
Dept. of Computational Intelligence
and Systems Science
Tokyo Institute of Technology
Nagatsuda, Midori-ku, Yokohama, Japan

Ken Satoh
Division of Electronics and
Information Engineering
Hokkaido University
N13W8, Kita-ku, Sapporo 060, Japan

Abstract

We propose a 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 paper, we show theoretically that any circumscription whose theory contains both the domain closure axiom and the uniqueness of names axioms can always be compiled into an extended logic program II, 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 II, which can be computed by running II under the existing logic programming interpreter.

1 Introduction

Circumscription [McCarthy, 1980; Lifschitz, 1985] was proposed to formalize the commonsense reasoning under incomplete information.

So far, many studies have been proposed to explore the approach of the use of logic programming for the automation of circumscription based on the relationship between the semantics of circumscription and the semantics of logic programs.

Gelfond and Lifschitz [1988a] was the first to consider a computational method for some restricted class of prioritized circumscription which compiles circumscriptive theories into a stratified logic program. Though their method is computationally efficient, the applicable class is too limited.

So we proposed the 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, our method expands the applicable class of circumscription by making use of the 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 probtem,* 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 paper, we propose 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, 1987], 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, any circumscription whose theory contains both the domain closure axiom and the uniqueness of names axioms can always be compiled into an extended logic program II, 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 II. This can be computed by running II 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.

The structure of the paper is as follows. In section 2, we give preliminaries related to circumscription and extended logic programs. In section 3, 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. We finish section 4 with comparing our method with related researches.

2 Preliminaries

We review parallel circumscription and prioritized circumscription as well as the syntax and semantics of extended logic programs (ELP) [Gelfond and Lifschitz, 1991] in a slightly extended form.

Let $T(P,Z)$ be a sentence, P be a tuple of minimized predicates and Z be a tuple of variable predicates. Q denotes the rest of the predicates occurring in T, called the fixed predicates. Then parallel circumscription of P in $T(P,Z)$ with variable Z is denoted by $Circum(T;P;Z)$. If P is broken into parts $P^1,\ldots,P^k$, then the circumscription assigning a higher priority to the members of P^i than to the members of P^j for $i<j$ is denoted by the following prioritized circumscription:

$$Circum(T;P^1>P^2>\ldots>P^k;Z).$$

Prioritized circumscription can be represented by using parallel circumscription as follows:

$$Circum(T;P^1>\ldots>P^k;Z) \equiv \bigwedge_{i=1}^{k} Circum(T;P^i;P^{i+1},\ldots,P^k,Z) \quad (2.1)$$

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

A extended logic program is a set of rules of the form:

$$L \leftarrow L_1,\ldots,L_m,notL_{m+1},\ldots,notL_n, \quad (2.2)$$

or of the form:

$$\leftarrow L_1,\ldots,L_m,notL_{m+1},\ldots,notL_n, \quad (2.3)$$

where $n \geq m \geq 0$, and L and L_i's are literals. Each rule of the form (2.3) is called an *integrity constraint*.

The semantics of an extended logic program is an extension of the *stable model semantics*[Gelfond and Lifschitz, 1988b] and is given by the *answer sets* [Gelfond and Lifschitz, 1991]. 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, $S = \alpha(\Pi^S)$.

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 $Th(\Sigma)$ [Inoue, 1992].

3 Translation from Circumscription into Extended Logic Programs

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.

In the following Definition 3.1 and Definition 3.2, we restrict a first order theory T to the one which is function-free and contains both the *domain closure axiom* (DCA) and *uniqueness of names axioms* (UNA). We consider T as a set of clauses:

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

We suppose 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 3.1
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,
$$\neg p(t) \leftarrow not\ p(t).$$
2. For any fixed predicate q from Q and any t,
$$\neg q(t) \leftarrow not\ q(t),$$
$$q(t) \leftarrow not\ \neg q(t).$$
3. For any clause $C \stackrel{def}{=} \ell_1 \vee \ell_2 \vee \ldots \vee \ell_n$ in $\mu Th(\Sigma)$ and any contrapositive of C such as $\neg\ell_{i_2} \wedge \ldots \wedge \neg\ell_{i_n} \supset \ell_{i_1}$ and $\neg\ell_{i_1} \wedge \neg\ell_{i_2} \wedge \ldots \wedge \neg\ell_{i_n} \supset false$,
$$\ell_{i_1} \leftarrow \neg\ell_{i_2},\ldots,\neg\ell_{i_n},$$

$$\leftarrow \neg \ell_{i_1}, \neg \ell_{i_2}, \ldots, \neg \ell_{i_n}.$$

where $\ell_{i_j} \in \{\ell_1, \ldots, \ell_n\}$ (for $j = 1, \ldots, n$).
As a special case, if C is a unit literal ℓ,

$$\ell \leftarrow, \qquad \leftarrow \neg \ell.$$

Definition 3.2

Given prioritized circumscription as follows:

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

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

1. According to (2.1), a given prioritized circumscription is represented by the conjunction of k parallel circumscriptions. So, let every ith $(1 \leq i \leq k)$ parallel circumscription,

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

   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:

   $$Circum(\tilde{\forall} Ti; Pi^i; Pi^{i+1}, \ldots, Pi^k, Zi), \qquad (3.1)$$

   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 (3.1) according to Definition 3.1.
   (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 in [Etherington, 1987].

Theorem 3.3 *Assume that T is a first order theory including DCA and UNA. Then, those formulas true in every default extension of the default theory:*

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

art precisely those theorems of Circum(T; P; Z)` where P,Q and Z are tuples of minimized, fixed and variable predicate symbols respectively.

Proof:(sketch) According to Theorem 1 in [de Kleer and Konolige, 1989], circumscription including fixed predicates q_i can be logically equivalently transformed into circumscription without fixed predicates where q_i and $\neg q_i$ are added to circumscribed predicates. As a result, this theorem is proved by applying Etherington's result [Etherington, 1987] about correspondence between circumscription without fixed predicates and Reiter's default theory [Reiter, 1980] to the transformed circumscription.

Based on Theorem 3.3 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 3.4 *For any ground literal G of the language of T whose predicate symbol is not equality, it holds that,*

$$Circum(\tilde{\forall} T; P; Z) \models G$$
iff *G is true in all answer sets of* ELP Π^α,

where ELP Π^α *is constructed from* $Circum(\tilde{\forall} T; P; Z)$ *by using definition 3.1.*

Theorem 3.5 *For any ground literal G of the language of T whose predicate symbol is not equality, it holds that,*

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

where ELP Π^β *is constructed from*

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

by using Definition 3.2.

Remarks. Theorem 3.4 and Theorem 3.5 can be easily generalized for a ground formula F instead of for a ground literal G.

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 (11,-4) in which A is a set of predicate symbols called abducible predicates. Their result is given as follows:

Suppose that a normal logic program 13 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 $(\Pi, \{\})$.

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 3.1 and Definition 3.2 respectively are finite, we can make use of their result in a slightly

extended form, and Theorem 3.4 and Theorem 3.5 can be said in other words by the following corollaries.

Corollary 3.6 *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* $(\Pi^\alpha, \{\})$,

where ELP Π^α *is constructed from* $Circum(\widetilde{\forall}T; P; Z)$ *by using Definition 3.1.*

Corollary 3.7 *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,*

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

iff *derive*$(not\ G, \{\})$ *fails under the abductive framework* $(\Pi, \{\})$.

where Π *consists of all rules of* ELP Π^β *in Definition 3.2 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 3.8
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 $x = Nixon$:

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

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, $\mu Th(\Sigma)$ is obtained as follows:

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

According to Definition 3.1, let us construct ELP Π^α from circumscription (1).

1. For minimized predicates ab_1, ab_2 and a constant 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 constant 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 $\mu Th(\Sigma)$,
$$ab_1(N) \leftarrow \neg pac(N),$$
$$pac(N) \leftarrow \neg ab_1(N),$$
$$\leftarrow \neg pac(N), \neg ab_1(N),$$
$$ab_2(N) \leftarrow pac(N),$$
$$\neg pac(N) \leftarrow \neg ab_2(N),$$
$$\leftarrow pac(N), \neg ab_2(N),$$
$$ab_1(N) \leftarrow \neg\ ab_2(N),$$
$$ab_2(N) \leftarrow \neg\ ab_1(N),$$
$$\leftarrow \neg ab_1(N), \neg ab_2(N),$$
$$rep(N) \leftarrow, \qquad \leftarrow \neg rep(N),$$
$$quak(N) \leftarrow, \qquad \leftarrow \neg quak(N),$$

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 3.9 Let us compile the following prioritized circumscription:

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

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

$$\neg q \supset p, \quad \neg p \supset q, \quad \neg p \wedge \neg q \supset false.$$

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

$$\neg r \supset q, \quad \neg q \supset r, \quad \neg q \wedge \neg r \supset false.$$

According to (2.1), it holds that

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

Then according to Definition 3.2, predicate symbols occurring each parallel circumscription are renamed as follows:

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

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

$(\Pi^\alpha)_1$:
$$\neg p1 \leftarrow not\ p1,$$
$$p1 \leftarrow \neg\ q1, \qquad q1 \leftarrow \neg\ p1,$$
$$\leftarrow \neg p1, \neg q1,$$
$$q1 \leftarrow \neg\ r1, \qquad r1 \leftarrow \neg\ q1,$$
$$\leftarrow \neg q1, \neg r1,$$

$(\Pi^\alpha)_2$:
$$\neg q2 \leftarrow not\ q2,$$
$$\neg p2 \leftarrow not\ p2,$$
$$p2 \leftarrow not\ \neg p2,$$
$$p2 \leftarrow \neg\ q2, \qquad q2 \leftarrow \neg\ p2,$$
$$\leftarrow \neg\ p2, \neg\ q2,$$
$$q2 \leftarrow \neg\ r2, \qquad r2 \leftarrow \neg\ q2,$$
$$\leftarrow \neg\ q2, \neg\ r2,$$

$$\Pi_\gamma: \quad p \leftarrow p1, \quad p \leftarrow p2,$$
$$q \leftarrow q1, \quad q \leftarrow q2,$$
$$r \leftarrow r1, \quad r \leftarrow r2,$$
$$\neg p \leftarrow \neg p1, \quad \neg p \leftarrow \neg p2,$$
$$\neg q \leftarrow \neg q1, \quad \neg q \leftarrow \neg q2,$$
$$\neg r \leftarrow \neg r1, \quad \neg r \leftarrow \neg r2.$$

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 3.5, 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 ¬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 ¬, which enables us to compute prioritized circumscription as is shown in Theorem 3.5. 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 3.10 [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 models of parallel circumscription and the answer sets of the **translated EDP Π^α.**

Definition 3.11
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 3.2.**
2. **EDP 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 disjunctive program (EDP) which is constructed from the ith renamed parallel circumscription (3.1) according to Definition 3.10.**
   (b) **Π_γ is the same set given by Definition 3.2.**

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

Theorem 3.12 *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,*

$$Circum(\widetilde{\forall} T; P^1 > P^2 > \ldots > P^k; Z) \models F$$
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 3.11.

Example 3.13
Consider prioritized circumscription given in Example 3.9. We apply Theorem 3.12 instead of Theorem 3.5 to it. Then according to Definition 3.11, 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: $Circum(\{p1 \vee q1, q1 \vee r1\}; p1; q1, r1)$,

$$(\Pi^\alpha)_1: \quad q1 \leftarrow not\ p1,$$
$$q1 | r1,$$
$$\neg p1 \leftarrow not\ p1,$$
$$q1 | \neg q1 \leftarrow, \qquad r1 | \neg r1 \leftarrow .$$

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

$$(\Pi^\alpha)_2: \quad p2 \leftarrow not\ q2,$$
$$r2 \leftarrow not\ q2,$$
$$p2 | q2 \leftarrow,$$
$$\neg q2 \leftarrow not\ q2,$$
$$r2 | \neg r2 \leftarrow, \qquad p2 | \neg p2 \leftarrow .$$

Π_γ is obtained as the same one shown in Example 3.9. 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 3.9.

4 Related Works and Conclusion

In this paper, 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 II, 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.

- Gelfond and Lifschitz's method [1988a] as well as our previous method [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 paper.
- 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 ¬ is available or not. Theorem 3.12 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.
- 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 method proposed in this paper.

References

[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, 1987] David W. Etherington. Relating Default Logic and Circumscription. *Proc. IJCAI-87,* pages 489-494, 1987.

[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] Matthew L. Ginsberg, A Circumscriptive **Theorem Prover,** ***Artificial Intelligence 39,*** **pages** 209-230, 1989.

[Inoue, 1992] Inoue, K. Linear Resolution for Consequence **Finding,** ***Artificial Intelligence 56,*** **pages 301-**353, 1992.

[inoue and Sakama, 1994] Inoue, K. and Sakama, C. On Positive Occurrences of Negation as Failure. *Proc. KR'94,* pages 293-304, 1994.

[Lifschitz, 1985] Lifschitz, V. Computing Circumscription. *Proc. IJCAI-85,* pages 121-127, 1985.

[McCarthy, 1980] McCarthy, J. Circumscription - a Form of Non-monotonic Reasoning. *Artificial Intelligence 13,* pages 27-39, 1980.

[Przymusinski, 1987] Przymusinski, T. On the Declarative Semantics of Deductive Databases and Logic **Programs, in** ***Foundations of Deductive Databases*** *and Logic Programming* (J. Minker, Ed.), Morgan Kaufmann, pages 193-216, 1987.

[Reiter, 1980] Reiter, R. A Logic for default reasoning. ***Artificial Intelligence 13,*** **pages 81-132, 1980.**

[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 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.

[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.**