Computing Preferred Answer Sets in Answer Set Programming

Toshiko Wakaki[1], Katsumi Inoue[2], Chiaki Sakama[3], and Katsumi Nitta[4]

[1] Shibaura Institute of Technology, Department of Electronic Information Systems, 307 Fukasaku, Minuma-ku, Saitama-City, Saitama 337–8570 Japan
`twakaki@sic.shibaura-it.ac.jp`
[2] Kobe University, Department of Electrical and Electronics Engineering, Rokkodai, Nada, Kobe 657–8501 Japan
`inoue@eedept.kobe-u.ac.jp`
[3] Wakayama University, Center for Information Science, 930 Sakaedani, Wakayama 640–8510, Japan
`sakama@sys.wakayama-u.ac.jp`
[4] Tokyo Institute of Technology, Department of Computational Intelligence and Systems Science, 4259 Nagatsuta, Midori-ku, Yokohama 226–8502, Japan
`nitta@dis.titech.ac.jp`

Abstract. A framework of *prioritized logic programs* (PLPs) is useful to represent explicit priorities between literals of logic programs. With its expressive power, PLPs theoretically enable us to realize various frameworks of nonmonotonic reasoning as well as preference abduction. However, its implementation issues have scarcely been studied and no sound procedure is known for computing preferred answer sets of PLPs. In this paper, we present a procedure to compute all preferred answer sets of a PLP in answer set programming. We show soundness and completeness theorems for the procedure. Finally we show that not only our procedure makes PLPs practically available but also it has the capability of representing dynamic preferences in addition to static ones.

1 Introduction

A framework of *prioritized logic programs* [18] (PLPs) introduces explicit representation of priorities between literals and negation-as-failure formulas to logic programs. A PLP is defined as a pair (P, Φ),where P is a (nonmonotonic) logic program and Φ is a set of priorities between literals and negation-as-failure formulas in the language. The semantics of PLP is given as *preferred answer sets* which are defined as the answer sets of P that are selected with respect to the priorities in Φ. It was shown that PLPs can realize various frameworks of nonmonotonic reasoning such as default reasoning [16], prioritized circumscription [13, 14] as well as preference abduction [12].

To realize prioritized reasoning in logic programming, there are several different frameworks and implementation techniques such as ordered logic programs [5], Logic Programs with Ordered Disjunctions [1], ordered default theories [4] and and preferred answer sets of extended logic programs [2]. As for the procedure of PLPs, on the other hand, Sakama and Inoue [18] provided a naive procedure for computing preferred answer sets of a PLP, but the procedure is applicable to a limited class of PLPs and it is turned unsound.

In this paper, we present a more efficient procedure to compute all preferred answer sets for a PLP in answer set programming (ASP) and show soundness and completeness theorems for the procedure. Our procedure is based on a *generate-and-test algorithm* and uses the technique of *meta-programming*. The basic idea of our approach is to translate a PLP (P, Φ) and any answer set S of a program P, into a single logic program $T[P, \Phi, S]$ whose answer sets represent answer sets of P preferable to S. More precisely, if $T[P, \Phi, S]$ is consistent, answer sets of P preferable to S can be obtained from the respective answer sets of $T[P, \Phi, S]$; otherwise we can conclude that such S is a "strictly" preferred answer set of (P, Φ). Thanks to our theorems, our procedure can compute all preferred answer sets of a given PLP, making use of preferences generated from such a translated logic program $T[P, \Phi, S]$ to decide whether an answer set S of P is preferred or not. Thus, our procedure can be easily implemented using answer set solvers (ASP solvers) such as *dlv* [6], *smodels* [15] and *MGTP* [10]. Moreover, we show that not only our procedure makes PLPs practically available but also our approach can accommodate dynamic preferences [3] in addition to the original static ones which significantly widen the class of PLPs and further increase their expressiveness.

The structure of the paper is as follows. In Section 2, we review some definitions and notation related to PLPs. In Section 3, we present two theorems and our sound and complete procedure of computing preferred answer sets. In Section 4, we give a brief discussion on applying our approach to a legal reasoning example. Section 5 provides some comparisons with related works and concluding remarks.

2 Preliminaries

We review some definitions and notations about PLP [17, 18].

2.1 General Extended Disjunctive Logic Programs

A *general extended disjunctive logic program* (GEDP) [11] is a set of rules of the form:

$$L_1 \mid \cdots \mid L_k \mid not\, L_{k+1} \mid not\, L_l \leftarrow L_{l+1}, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n, \qquad (1)$$

where $n \geq m \geq l \geq k \geq 0$, each L_i is a literal, that is either an atom A or its negation $\neg A$, and " |" represents a disjunction. The rule with the empty head is called an *integrity constraint*. A rule with variables stands for the set of its ground instances (i.e. ground rules).

The semantics of a GEDP P is given by the *answer sets* [7, 11] as follows.

Definition 1 Let Lit_P be a set of all ground literals in the language of P.

First, let P be a *not*-free GEDP (i.e., for each rule $k = l, m = n$). Then, $S \subseteq Lit_P$ is an *answer set* of P if S is a minimal set satisfying the conditions:

1. For each ground rule $L_1 \mid \cdots \mid L_l \leftarrow L_{l+1}, \ldots, L_m$ in P, if $\{L_{l+1}, \ldots, L_m\} \subseteq S$, then $L_i \in S$ for some i ($1 \leq i \leq l$); In particular, for each integrity constraint $\leftarrow L_1, \ldots, L_m$ in P, $\{L_1, \ldots, L_m\} \not\subseteq S$ holds;
2. If S contains a pair of complementary literals, then $S = Lit_P$.

Second, let P be any GEDP and $S \subseteq Lit_P$. The *reduct* of P by S is a *not*-free GEDP P^S obtained as follows:

A rule $L_1 \mid \cdots \mid L_l \leftarrow L_{l+1}, \ldots, L_m$ is in P^S
iff there is a ground rule of the form (1) from P such that
$$\{L_{k+1}, \ldots, L_l\} \subseteq S \text{ and } \{L_{m+1}, \ldots, L_n\} \cap S = \emptyset.$$

For P^S, its answer sets have already been defined. Then, S is an answer set of P if S is an answer set of P^S.

An answer set is *consistent* if it is not Lit_P. The answer set Lit_P is said *contradictory*. A GEDP is *consistent* if it has a consistent answer set; otherwise, the program is *inconsistent*.

2.2 Prioritized Logic Programs

Given a GEDP P and the set of ground literals Lit_P, let $\mathcal{L}^*_P$ be the set defined as $Lit_P \cup \{notL \mid L \in Lit_P\}$. A prioritized logic program (PLP) is defined as follows.

Definition 2 *(Priorities between literals)*
A reflexive and transitive relation $\preceq$ is defined on $\mathcal{L}^*_P$. For any element e_1 and e_2 from $\mathcal{L}^*_P$, $e_1 \preceq e_2$ is called a *priority*, and we say e_2 *has a higher priority than* e_1. We write $e_1 \prec e_2$ if $e_1 \preceq e_2$ and $e_2 \not\preceq e_1$, and say e_2 *has a strictly higher priority than* e_1. A nonground priority stands for the set of its ground instances. That is, for tuples $\mathbf{x}$ and $\mathbf{y}$ of variables, $e_1(\mathbf{x}) \preceq e_2(\mathbf{y})$ stands for any priority $e_1(\mathbf{t}) \preceq e_2(\mathbf{s})$ for any instance $\mathbf{t}$ of $\mathbf{x}$ and instance $\mathbf{s}$ of $\mathbf{y}$.

Definition 3 *(Prioritized Logic Programs: PLPs)*
A *prioritized logic program* (PLP) is defined as a pair (P, Φ), where P is a GEDP and Φ is a set of priorities on $\mathcal{L}^*_P$.

The declarative semantics of PLP is given by preferred answer sets as follows.

Definition 4 *(Preferences between answer sets)*
Given a PLP (P, Φ), the preference relation $\sqsubseteq$ on answer sets of P is defined as follows: Let S_1 and S_2 be two answer sets of P. Then, S_2 is *preferable* to S_1 with respect to Φ, written as $S_1 \sqsubseteq S_2$, if for some element $e_2 \in S_2 \setminus S_1$,
(i) there is an element $e_1 \in S_1 \setminus S_2$ such that $e_1 \preceq e_2$, and (ii) there is no element $e_3 \in S_1 \setminus S_2$ such that $e_2 \prec e_3$.

Besides, the relation $\sqsubseteq$ on answer sets is also defined as reflexive and transitive. We write $S_1 \sqsubset S_2$ if $S_1 \sqsubseteq S_2$ and $S_2 \not\sqsubseteq S_1$. Hereafter, each $S_1 \sqsubseteq S_2$ is called *preference.*

Definition 5 *(Preferred answer sets)*
An answer set S of P is called a *preferred answer set* (or *p-answer set*, for short) of P (with respect to Φ) if $S \sqsubseteq S'$ implies $S' \sqsubseteq S$ for any answer set S' of P.

Example 1. Let (P, Φ) be the PLP such that

$$P: \quad p \leftarrow not\ q, \qquad\qquad \Phi: \quad p \preceq not\ q,$$
$$q \leftarrow not\ p,$$

where P has two answer sets $\{p\}$ and $\{q\}$. Of these, $\{q\}$ becomes the unique preferred answer set of (P, Φ).

Generally, NAF formulas in Φ are eliminated without changing the meaning of a PLP [18]. In the above example, a PLP (P, Φ) is transformed to the semantically equivalent (P', Φ') such that $P' = P \cup \{p' \leftarrow not\ p\}$, $\Phi' = \Phi \cup \{p \preceq p'\}$. As a result, (P', Φ') has the unique p-answer set $\{p',\ q\}$ which coincides with $\{q\}$ wrt literals from Lit_p.

Then, without loss of generality in the following sections we consider a PLP (P, Φ) in which Φ contains no NAF formula.

3 Computing Preferred Answer Sets

In this section, we introduce a sound and complete procedure for computing preferred answer sets of PLPs. In this section, we consider a ground PLP, i.e., a PLP (P, Φ) such that P is a ground GEDP and Φ is a set of priorities over ground literals.

3.1 Translation for Preference Generation

As is mentioned in the introduction, our procedure of computing preferred answer sets of a PLP is regarded as a *generate-and-test algorithm*, which constructs a logic program $T[P, \Phi, S]$ translated from both a given PLP (P, Φ) and any answer set S of a program P in order to generate preferences between answer sets of P in answer set programming.

First, we encode a given answer set S and another answer set S' of P in the single answer set of a program $T[P, \Phi, S]$, which is used for judging whether S' is preferable to S. To this end, we use *renaming of literals* such that each literal $L \in Lit_P$ in S is renamed by the newly introduced literal L^* respectively. This technique symbolically enables us to embed one answer set $S \subseteq Lit_P$ as a set S^* of renamed literals L^*, together with another one $S' \subseteq Lit_P$ in the same answer set E of $T[P, \Phi, S]$.

Second, to compare a literal $c \in S$ and another literal $d \in S'$ according to (i) and (ii) of Definition 4, we use *meta-programming* techniques. That is, to compare literals, a new ground term L_t is introduced for every literal $L \in Lit_P$. Note that both a literal $L \in Lit_P$ and its renamed literal L^* mentioned above are expressed using the same ground term L_t introduced for the corresponding literal L.

Third, we provide predicate symbols m_1 and m_2 such that, for the term c_t which corresponds to some literal $c \in Lit_P$ as well as its renamed literal c^*, $m_1(c_t)$ means $c \in S$ for a given answer set S, while $m_2(c_t)$ means $c \in S'$ for any answer set S' of P.

In the following, we define two sets, Lit^*_P and $\mathcal{C}$ where Lit^*_P is a set of renamed literals L^*s and $\mathcal{C}$ is a set of newly introduced ground terms L_ts mentioned above. Due to the restriction of answer set programming, we suppose that Lit_P is finite, and ground terms L_ts in $\mathcal{C}$ are individual constants which have no function symbols.

Definition 6 Lit^*_P and $\mathcal{C}$ are defined as follows.

$$Lit^*_P \overset{def}{=} \{L^* | L \ \in Lit_P\}$$
$$\mathcal{C} \ \overset{def}{=} \{L_t | L \in Lit_P\}$$

Next we define $T[P, \Phi, S]$ which is a *meta-program* constructed using a PLP (P, Φ) and an answer set S.

Definition 7 Given a PLP (P, Φ) and an answer set S of P, $T[P, \Phi, S]$ is the GEDP defined as:

$$T[P, \Phi, S] \overset{def}{=} P \cup \Gamma \cup \Pi,$$

where Γ is the set of *domain dependent* rules constructed from Φ and S as follows,

1. $L^* \leftarrow,$ for each $L \in S$,
   where each $L^* \in Lit^*_P$ is a renamed literal corresponding to $L \in S$ respectively,
2. $\preceq (a_t, b_t) \leftarrow,$ for any $a \preceq b \in \Phi$
   where $a_t, b_t \in \mathcal{C}$ are respective ground terms expressing literals $a, b \in Lit_P$,
3. $m_1(L_t) \leftarrow L^*,$ $m_2(L_t) \leftarrow L,$
   for every $L \in Lit_P$, its renamed literal $L^* \in Lit^*_P$ and a ground term $L_t \in \mathcal{C}$ expressing a literal L,

and Π is the set of *domain independent* rules as follows:

4. $\preceq (x, x) \leftarrow,$
5. $\preceq (x, z) \leftarrow \preceq (x, y), \ \preceq (y, z),$

6. $\prec(x,y) \leftarrow \preceq(x,y),\ not \preceq(y,x),$
7. $gr_1(x,y) \leftarrow m_1(x),\ \preceq(x,y),\ m_2(y), not\ m_2(x),\ not\ m_1(y),$
8. $gr_2(y,z) \leftarrow m_2(y),\ \prec(y,z),\ m_1(z), not\ m_1(y),\ not\ m_2(z),$
9. $attacked(y) \leftarrow gr_2(y,z),$
10. $defeated(x) \leftarrow gr_1(x,y),\ not\ attacked(y),$
11. $better \leftarrow defeated(x),$
12. $\leftarrow not\ better.$

Remark 1. $\preceq$ and $\prec$ in the above rules are predicate symbols.

Remark 2. The rule 2 enables us to express rules of P and priorities in Φ in the same logic program. The rule 4 and the rule 5 represent the reflexive and transitive laws of $\preceq$ respectively. The rule 5 and the rule 6 calculate the closure of the priority relation $\preceq$ and that of the strict priority relation $\prec$ respectively. Rules $7 \sim 10$ compute preference between S and another answer set of P according to (i) and (ii) of Definition 4. Thus, the rule 10 means that $defeated(x)$ is true if there exists some answer set S' of P such that, for an element $x \in S \setminus S'$, there exists some element $y \in S' \setminus S$ which has a higher priority than a element x and any element $z \in S \setminus S'$ does not have a strictly higher priority than y. Rule 11 means that *better* is true if $defeated(x)$ is true for such $x \in S \setminus S'$.

Example 2. Let (P, Φ) be the PLP such that

$$\begin{aligned} P\colon\quad & p \mid q \leftarrow, \\ & q \mid r \leftarrow . \\ \Phi\colon\quad & p \preceq q,\ q \preceq r. \end{aligned}$$

P has two answer sets $S_1 = \{p, r\}$ and $S_2 = \{q\}$. With respect to the answer set S_1, $T[P, \Phi, S_1] = P \cup \Gamma_1 \cup \Pi$, is constructed with the following Γ_1,

$$\begin{aligned} \Gamma_1 :\ & p^* \leftarrow, \quad r^* \leftarrow, \preceq(p_t, q_t) \leftarrow, \quad \preceq(q_t, r_t) \leftarrow, \\ & m_1(p_t) \leftarrow p^*, \quad m_1(q_t) \leftarrow q^*, \quad m_1(r_t) \leftarrow r^*, \\ & m_1(np_t) \leftarrow \neg p^*, \quad m_1(nq_t) \leftarrow \neg q^*, \quad m_1(nr_t) \leftarrow \neg r^*, \\ & m_2(p_t) \leftarrow p, \quad m_2(q_t) \leftarrow q, \quad m_2(r_t) \leftarrow r, \\ & m_2(np_t) \leftarrow \neg p, \quad m_2(nq_t) \leftarrow \neg q, \quad m_2(nr_t) \leftarrow \neg r. \end{aligned}$$

where $Lit_P = \{p, q, r, \neg p, \neg q, \neg r\}$, $Lit^*_P = \{p^*, q^*, r^*, \neg p^*, \neg q^*, \neg r^*\}$,
$\mathcal{C} = \{p_t, q_t, r_t, np_t, nq_t, nr_t\}$.

Now, we define two kinds of preferred answer sets and show two theorems with respect to $T[P, \Phi, S]$, which guarantee the *soundness* and *completeness* of our procedure to compute all preferred answer sets of (P, Φ).

Definition 8 *(tie-preferred, and strictly preferred answer set)*
A preferred answer set S of a PLP $(P, \varPhi)$ is called *tie-preferred* if there is another preferred answer set S' of $(P, \varPhi)$ such that $S \sqsubseteq S'$ and $S' \sqsubseteq S$. S is called *strictly preferred* if $S \not\sqsubseteq S'$ for any preferred answer set S'.

Theorem 1. *(soundness/completeness of the procedure) Let $T[P, \varPhi, S]$ be a GEDP constructed from a PLP $(P, \varPhi)$ and an answer set S of P. Then it holds that, if $T[P, \varPhi, S]$ is consistent, $S' \overset{def}{=} E \cap Lit_P$ is another answer set of P such that $S \sqsubseteq S'$ for any answer set E of $T[P, \varPhi, S]$. Conversely, if there is another answer set S' of P such that $S \sqsubseteq S'$, then $T[P, \varPhi, S]$ is consistent.*

Proof: *See Appendix.*

Theorem 2. *Let $T[P, \varPhi, S]$ be a GEDP constructed from a PLP $(P, \varPhi)$ and an answer set S of P. Then it holds that, $T[P, \varPhi, S]$ is inconsistent if and only if S is a strictly preferred answer set of $(P, \varPhi)$.*

Proof: *See Appendix.*

Example 3. Consider the PLP $(P, \varPhi)$ in Example 2. According to Theorem 2, we can conclude that $S_1 = \{p, r\}$ is a strictly preferred answer set of $(P, \varPhi)$ since $T[P, \varPhi, S_1]$ is inconsistent. By contrast, for $S_2 = \{q\}$,

$$T[P, \varPhi, S_2] = P \cup \varGamma_2 \cup \varPi$$
$$where\ \varGamma_2 = \varGamma_1 \setminus \{p^* \leftarrow, r^* \leftarrow\} \cup \{q^* \leftarrow\},$$

becomes consistent and has only one answer set E such that $E \cap Lit_P = \{p, r\}$, i.e. S_1. Thus we can obtain preference such that $S_2 \sqsubseteq S_1$ according to Theorem 1.

Example 4. Let $(P, \varPhi)$ be the PLP such that

$$\begin{aligned}
P:\quad & p \leftarrow not\ q,\\
& q \leftarrow not\ p,\\
& r \leftarrow p, \quad \neg s \leftarrow q.\\
\varPhi:\quad & p \preceq q,\ \neg s \preceq r. \qquad \text{where } Lit_P = \{p, q, r, s, \neg p, \neg q, \neg r, \neg s\}
\end{aligned}$$

P has two answer sets $S_1 = \{p, r\}$ and $S_2 = \{q, \neg s\}$.
With respect to $S_1 = \{p, r\}$, $T[P, \varPhi, S_1] = P \cup \varGamma_1 \cup \varPi$ has rules of $\varGamma_1$ as follows:

$$\begin{aligned}
\varGamma_1:\ & p^* \leftarrow, \quad r^* \leftarrow, \quad \preceq(p_t, q_t) \leftarrow, \quad \preceq(ns_t, r_t) \leftarrow,\\
& m_1(p_t) \leftarrow p^*, \quad m_1(q_t) \leftarrow q^*, \quad m_1(r_t) \leftarrow r^*, \quad m_1(s_t) \leftarrow s^*,\\
& m_1(np_t) \leftarrow \neg p^*, \quad m_1(nq_t) \leftarrow \neg q^*, \quad m_1(nr_t) \leftarrow \neg r^*, \quad m_1(ns_t) \leftarrow \neg s^*,\\
& m_2(p_t) \leftarrow p, \quad m_2(q_t) \leftarrow q, \quad m_2(r_t) \leftarrow r, \quad m_2(s_t) \leftarrow s,\\
& m_1(np_t) \leftarrow \neg p, \quad m_1(nq_t) \leftarrow \neg q, \quad m_1(nr_t) \leftarrow \neg r, \quad m_1(ns_t) \leftarrow \neg s.
\end{aligned}$$

where $Lit^*_P = \{p^*, q^*, r^*, s^*, \neg p^*, \neg q^*, \neg r^*, \neg s^*\}, \mathcal{C} = \{p_t, q_t, r_t, s_t, np_t, nq_t, nr_t, ns_t\}$.
In this case, $T[P, \varPhi, S_1]$ is consistent and has only one answer set E_1 such that $E_1 \cap Lit_P = \{q, \neg s\}$, i.e. S_2. Similarly, for S_2, $T[P, \varPhi, S_2]$ is consistent and has only

one answer set E_2 such that $E_2 \cap Lit_P = \{p, r\}$, i.e. S_1. As a result, preferences such that $S_1 \sqsubseteq S_2$ and $S_2 \sqsubseteq S_1$ are obtained according to Theorem 1. Thus according to Definition 8, we can decide that both S_1 and S_2 are tie-preferred answer sets.

3.2 A Procedure of Computing Preferred Answer Sets

We introduce a procedure *CompPAS* which computes all preferred answer sets of a PLP (P, Φ) based on Theorem 1 as follows. The procedure *CompPAS* uses a translated program $T[P, \Phi, S]$ to generate preferences with respect to a given answer set S. In addition, it uses a program Ψ shown in Table.1, to find all preferred answer sets of (P, Φ) from preferences generated by $T[P, \Phi, S]$.

Moreover, every answer set S of P is assigned a newly introduced individual constant s called an *answer set ID* respectively. Now, the procedure is as follows.

Procedure 1 *CompPAS(P, Φ, Δ)*

Input: a PLP (P, Φ)
Output: the set Δ of all preferred answer sets of (P, Φ)

In the following, AS is the set of answer sets of P, S and S' are answer sets of P, Ω is the set of answer set IDs, Δ is a set of preferred answer sets and Σ is a set of preferences which is initially an empty set ϕ.

1. Compute the set AS of all answer sets of P.
2. If Φ is an empty set $\emptyset$,
   (a) then $\Delta := AS$, return Δ.
   (b) otherwise,
      i. let Ω be a set of answer set IDs such that $|\Omega| = |AS|$ [1],
      ii. for each answer set $S \in AS$, assign the corresponding answer set ID.
3. If $T[P, \Phi, S]$ is consistent for any answer set $S \in AS$ whose answer set ID is s, do from (a) to (c) for each answer set E of $T[P, \Phi, S]$.
   (a) put $S' := E \cap Lit_P$,
   (b) find the answer set ID $s' \in \Omega$ for S' where $S' \in AS$ by Theorem 1,
   (c) put $\Sigma := \Sigma \cup \{\sqsubseteq (s, s') \leftarrow\}$.
4. Compute an answer set U of the following logic program,

$$\Psi \cup \Sigma \cup \{as(s) \leftarrow |\, s \in \Omega\}$$

   where Ψ is a set of rules shown in Table 1.
5. Return Δ which is given by
   $\Delta = \{S \in AS \mid S$ is an answer set whose answer set ID s satisfies
   p-as(s) $\in U\}$.

[1] For any set A, $|A|$ denotes the cardinality of A.

Table 1. A set Ψ of rules

$\sqsubseteq (x,x) \leftarrow as(x),$
$\sqsubseteq (x,z) \leftarrow \sqsubseteq (x,y),\ \sqsubseteq (y,z),$
$\sqsubset (x,y) \leftarrow \sqsubseteq (x,y),\ not \sqsubseteq (y,x),$
$worse(x) \leftarrow \sqsubset (x,y),$
$p\text{-}as(x) \leftarrow as(x),\ not\ worse(x).$

Remark 3. $\sqsubseteq$ and $\sqsubset$ are predicate symbols denoting preference relations defined in Definition 4 .

Remark 4. *as(s)* and *p-as(s)* represent that there exists some answer set of P whose answer set ID is s, and there exists some preferred answer set of (P, Φ) whose answer set ID is s, respectively.

Example 5. Let (P, Φ) be the PLP such that

$$\begin{aligned} P: \quad & p \leftarrow not\ q, not\ r, \\ & q \leftarrow not\ p, not\ r, \\ & r \leftarrow not\ p, not\ q, \\ & s \leftarrow q. \\ \Phi: \quad & p \preceq q,\ q \preceq p,\ s \preceq r. \end{aligned}$$

Here Lit_P, Lit^*_P and $\mathcal{C}$ are the same as those of Example 4. Preferred answer sets of this (P, Φ) can be computed using Procedure 1 as follows. P has three answer sets such as $S_1 = \{p\}$, $S_2 = \{q, s\}$, and $S_3 = \{r\}$, whose answer set IDs are s_1, s_2 and s_3, respectively. Then, in step 3, since $T[P, \Phi, S]$ is inconsistent only for $S = S_3$,

$$\Sigma = \{\sqsubseteq(s_1, s_2) \leftarrow,\ \sqsubseteq(s_2, s_1) \leftarrow,\ \sqsubseteq(s_2, s_3) \leftarrow\}$$

is obtained from preferences which are $S_1 \sqsubseteq S_2$ generated by $T[P, \Phi, S_1]$ as well as $S_2 \sqsubseteq S_1$, $S_2 \sqsubseteq S_3$ by $T[P, \Phi, S_2]$. In step 5, $\Delta = \{\{r\}\}$ is returned, since *p-as*$(s_3) \in U$, *p-as*$(s_1) \notin U$, *p-as*$(s_2) \notin U$ for an answer set U of a program: $\Psi \cup \Sigma \cup \{as(s_1) \leftarrow, as(s_2) \leftarrow, as(s_3) \leftarrow\}$. Thus, using the procedure, we can obtain the result that only $\{r\}$ is a preferred answer set of (P, Φ). [2]

Example 6. Let us compute the tie-preferred answer sets of the PLP in Example 4 using Procedure 1. Suppose that in step 2 of the procedure, s_1 and s_2 are assigned to answers sets $S_1 = \{p, r\}$ and $S_2 = \{q, \neg s\}$ as their IDs respectively.

In step 3, Σ is obtained as $\{\sqsubseteq (s_1, s_2) \leftarrow,\ \sqsubseteq (s_2, s_1) \leftarrow\}$ wrt preferences $S_1 \sqsubseteq S_2$ and $S_2 \sqsubseteq S_1$ shown in Example 4. Thus, in step 4, we obtain $p\text{-}as(s_1) \in U$ and $p\text{-}as(s_2) \in U$ for the answer set U of $\Psi \cup \Sigma \cup \{as(s_1) \leftarrow,\ as(s_2) \leftarrow\}$.

Therefore, we can decide both $\{p, r\}$ and $\{q, \neg s\}$ as preferred answer sets of (P, Φ) using our procedure.

[2] Sakama and Inoue's procedure for selecting p-answer sets [18] computes both $\{p\}$ and $\{r\}$ as preferred answer sets of this (P, Φ). So, their procedure is not sound.

4 Dynamic Preferences and Application to Legal Reasoning

Our approach can accommodate *dynamic preferences* [3] by slightly extending the framework of PLPs though the original PLPs are limited to the static ones.

4.1 Expressing Dynamic Preferences

In this section, we show that our procedure enables to express a *priority with preconditions* keeping both Theorem 1 and Theorem 2, which significantly increases the expressiveness of PLPs to accommodate dynamic preferences.

Definition 9 Let a PLP be a pair $(P, \tilde{\Phi})$ where P is a GEDP and $\tilde{\Phi}$ is a *stratified logic program* whose rules have the form:

$$A \leftarrow B_1, \ldots, B_m, not\, C_{m+1}, \ldots, not\, C_n, \qquad (n \geq m \geq 0)$$

where A, B_i, and C_j $(1 \leq i \leq m,\ m+1 \leq j \leq n)$ are atoms. In $\tilde{\Phi}$, there is at least one rule whose head atom A is $\preceq (a_t, b_t)$ where $a_t, b_t \in \mathcal{C}$ are ground terms expressing respective literals $a, b \in Lit_P$. Besides, any predicate symbol occurring in $\tilde{\Phi}$ is newly introduced except $\preceq$. Each rule containing variables stands for the set of all its ground instances such that any variable is replaced by any ground term from $\mathcal{C}$. The *stratification* [8] of $\tilde{\Phi}$ is $P^1; \ldots; P^k$ such that, for every predicate p from P^i $(1 \leq i \leq k)$, (a) all predicates that occur in the definition of p belong to $P^1; \ldots; P^i$, and (b) all predicates that occur in the definition of p under *not* belong to $P^1; \ldots; P^{i-1}$. For $i < j$, we call that a stratum P^i is higher than a stratum P^j. In particular, $\preceq$ should belong to the *stratum* P^k, whose predicates have the lowest priority.

Definition 10 Given a PLP $(P, \tilde{\Phi})$ and an answer set S of P, $T[P, \tilde{\Phi}, S]$ is the GEDP defined as $T[P, \tilde{\Phi}, S] \overset{def}{=} P \cup \tilde{\Gamma} \cup \Pi$, where $\tilde{\Gamma}$ is is the same as Γ in Definition 7 except that each rule $\preceq (a_t, b_t) \leftarrow$ in the item 2 is replaced by the set of rules $\tilde{\Phi}$.

Theorem 3. *Let $T[P, \tilde{\Phi}, S]$ be a GEDP constructed from a PLP $(P, \tilde{\Phi})$ and an answer set S of P. Then, both Theorem1 and Theorem2 hold if $T[P, \Phi, S]$ is replaced by $T[P, \tilde{\Phi}, S]$ in these theorems.*

4.2 Application to Legal Reasoning

We discuss an application of our procedure to a legal reasoning example with dynamic preferences. It is shown that a predicate expressing the conflict between higher meta-level priorities belongs to a higher stratum in $\tilde{\Phi}$ as follows.

The legal problem [9] is as follows. The domain knowledge is about *the person's ship* and *laws of the Uniform Commercial Code (UCC)* and *the Ship Mortgage Act (SMA)*, which are expressed by a set P of the following rules,

$$\begin{array}{lll} P: & \mathit{perfected} \leftarrow \mathit{posses}, \mathit{not}\ \mathit{ab}1, & (\mathit{UCC}) \\ & \neg \mathit{perfected} \leftarrow \mathit{ship}, \neg \mathit{filstate}, \mathit{not}\ \mathit{ab}2, & (\mathit{SMA}) \\ & \mathit{posses} \leftarrow, \quad \mathit{ship} \leftarrow, \quad \neg \mathit{filstate} \leftarrow, & \\ & \mathit{ab}1 | \mathit{not}\ \mathit{ab}1 \leftarrow, \quad \mathit{ab}2 | \mathit{not}\ \mathit{ab}2 \leftarrow, \quad \leftarrow \mathit{ab}1, \mathit{ab}2, & \\ & \mathit{ucc} \leftarrow \mathit{not}\ \mathit{ab}1, \quad \mathit{sma} \leftarrow \mathit{not}\ \mathit{ab}2. & \end{array}$$

Since the two laws are in conflict with one another, they lead to two answer sets S_1 and S_2 of P as follows.

$$S_1 = \{\mathit{perfected}, \mathit{posses}, \mathit{ship}, \neg \mathit{filstate}, \mathit{ab}2, \mathit{ucc}\}.$$
$$S_2 = \{\neg \mathit{perfected}, \mathit{posses}, \mathit{ship}, \neg \mathit{filstate}, \mathit{ab}1, \mathit{sma}\}.$$

Now, there are two well-known legal principles for resolving such conflict between laws as follows.
The principle of Lex Posterior gives precedence newer laws, and the principle of Lex Superior gives precedence to laws supported by the higher authority. In our case, UCC is newer than the SMA, and the SMA has higher authority since it is a federal law.

The above knowledge is described as a set $\tilde{\Phi}_1$ which consists of priorities with preconditions. Then, we can represent it as the following stratified logic program, which corresponds to the extended rule 2 of Γ in Definition 7 as follows.

$$\begin{array}{lll} \tilde{\Phi}_1: & \mathit{moreRecent}(\mathit{ucc}_t, \mathit{sma}_t) \leftarrow, & \\ & \mathit{fed}(\mathit{sma}_t) \leftarrow, \quad \mathit{state}(\mathit{ucc}_t) \leftarrow, & \\ & \mathit{lp}(Y,X) \leftarrow \mathit{moreRecent}(X,Y), & \\ & \mathit{ls}(Y,X) \leftarrow \mathit{fed}(X), \mathit{state}(Y), & \\ & \preceq (Y,X) \leftarrow \mathit{lp}(Y,X), \mathit{not}\ \mathit{conf}_1(Y,X), & (\mathit{LP}) \\ & \preceq (Y,X) \leftarrow \mathit{ls}(Y,X), \mathit{not}\ \mathit{conf}_1(Y,X), & (\mathit{LS}) \end{array}$$

where conf_1 is a predicate symbol denoting *conflict* between the legal principles with respect to laws X and Y. In this case of $(P, \tilde{\Phi}_1)$, $T[P, \tilde{\Phi}_1, S_1]$ has only one answer set E_1 such that $E_1 \cap \mathit{Lit}_P = S_2$, and $T[P, \tilde{\Phi}_1, S_2]$ has only one answer set E_2 such that $E_2 \cap \mathit{Lit}_P = S_1$. These lead to $S_1 \sqsubseteq S_2$ and $S_2 \sqsubseteq S_1$. As a result, we obtain two *tie-preferred* answer sets S_1 and S_2 of this PLP due to the conflict between two legal principles, i.e. Lex Posterior and Lex Superior.

Next, suppose we have a new preference information such that Lex Superior has a higher priority than Lex Posterior as follows.

$$\mathit{LexPosterior}(X,Y) \preceq \mathit{LexSuperior}(U,V).$$

In our framework, such an additional *meta-priority* can be expressed by a tie-breaking rule (2) as follows.

$$\mathit{conf}_1(X,Y) \leftarrow \mathit{lp}(X,Y), \mathit{ls}(Y,X), \mathit{not}\ \mathit{conf}_2(Y,X), \qquad (2)$$

where conf_2 denotes the conflict of one level higher priorities than that of conf_1. Let $\tilde{\Phi}_2$ be $\tilde{\Phi}_1 \cup \{\text{rule (2)}\}$. It should be noted that conf_2 belongs to the one level higher stratum than that of conf_1 in the stratification of $\tilde{\Phi}_2$. Then, this case is expressed by a PLP $(P, \tilde{\Phi}_2)$ where $T[P, \tilde{\Phi}_2, S_2]$ is inconsistent. Thus we obtain the result that S_2 is a *strictly preferred* answer set of $(P, \tilde{\Phi}_2)$, but S_1 is not preferred in similar way. Therefore, $\neg perfected$ is determined.

5 Related Works and Conclusion

In this paper, we present a sound and complete procedure to compute all preferred answer sets of a given PLP based on answer set programming. Moreover, we introduce the capability of representing not only static preference but also dynamic one by slightly extending the framework of PLPs. With respect to complexity, our procedure calls the ASP solver polynomial order times in step 3. We are now going to implement our procedure by using *dlv* and C++.

In the following, we compare our approach with related works in the aspects of the methodology, preference representation and complexity.

(1) Sakama and Inoue's naive procedure

Sakama and Inoue [18] firstly provided the procedure for selecting preferred answer sets of a PLP. Their procedure computes all answer sets and then finds preferred answer sets by means of comparing all answer sets pairwise with respect to the "strict" priority relation $\prec$ obtained from Φ. According to their Theorem 4.1 in [18], they claim that their procedure is sound, and completeness of the procedure also holds if preferred answer sets of a given PLP are *cycle-free.* But Example 5 of this paper shows that their procedure is not sound. To fix the problem, it is necessary to use the pre-order priority relation $\preceq$ which is reflexive and transitive instead of using irreflexive "strict" priority relation $\prec$ from Φ. With this consideration, their procedure can compute the correct preferred answer set of Example 5. However, even if their procedure is corrected to become sound in this way, there still exists such a problem in the corrected procedure that it cannot find any tie-preferred answer set but only compute strictly preferred answer sets.

By contrast, without comparing all answer sets pairwise, our procedure $CompPAS$ can obtain all strictly preferred answer sets immediately by checking the inconsistency of $T[P, \Phi, S]$ for each answer set of P in step 3. In addition, our procedure can find any strictly preferred answer set as well as any tie-preferred answer of a given PLP in step 4 and step 5 of $CompPAS$ based on Theorem 1. Furthermore, in our approach, both P and Φ are represented in the same logic program $T[P, \Phi, S]$, it enables us to handle dynamic preferences in PLP though it is originally limited to the static ones.

(2) Delgrande and Schaub's approach

Delgrande and Schaub proposed the framework of a *ordered logic program* [5]. It is represented by an extended logic program in which rules are named by terms and preferences among rules are given by a set of preference atoms representing preference relations over a set of rule names. According to their methodology, since each preference between rules is specified as a strict partial order, preferred answer sets of an ordered logic program Π can be obtained as answer sets which are $<_X$-preserving, of the standard extended logic program $\tau(\Pi)$ translated from Π. This feature is a little similar to our case that the strictly preferred answer set of PLP can be decided directly when our translated program $T[P, \Phi, S]$ is inconsistent. However, in our case, tie-preferred answer sets can be also found after such translation, since priorities are represented by a pre-order relation $\preceq$. The complexity of our approach (see Lemma 4.4 in [18]) lies in the one level higher of the polynomial hierarchy than theirs. With respect to dynamic preferences, their framework can treat them as well as ours.

(3) Brewka, Niemelä and Syrjänen's approach

Brewka et al.[1] proposed *logic programs with ordered disjunction*(*LPOD*s). According to their approach, an ordered disjunction appears in the head of a rule which enables to represent knowledge above preference. Their implementation is based on two normal (non-disjunctive) logic programs, a generator and a tester in order to compute preferred answer sets efficiently. Our procedure also consists of a generator which generates answer sets in step 1, and a tester which checks whether each answer set is preferred or not in step 4 using the preferences generated in step 3.

References

1. G. Brewka, I. Niemelä and T. Syrjänen: Implementing Ordered Disjunction Using Answer Set Solvers for Normal Programs, *Proc. 8th European Conference on Logics in Artificial Intelligence (JELIA'02), LNAI 2424*, Springer (2002) 445-455.
2. G. Brewka and T. Eiter: Preferred Answer Sets for Extended Logic Programs, *Artificial Intelligence* **109** (1999) 297-356.
3. G. Brewka: Well-founded Semantics for Extended Logic Programs with Dynamic Preferences, *Journal of Artificial Intelligence Research 4* (1996) 19-36.
4. J. P. Delgrande and T. Schaub T: Expressing Preferences in Default Logic, *Artificial Intelligence* **123** (2000) 41-87.
5. J. P. Delgrande, T. Schaub and H. Tompits: A Framework for Compiling Preferences in Logic Programs, Theory and Practice of Logic Programming 3(2) (2003) 129-187.
6. T. Eiter, N. Leone, C. Mateis, G. Pfeifer, F. Scarcello: A deductive system for nonmonotonic reasoning, *Proc. LPNMR-97, LNCS 1265*, Springer (1997) 364-375.
7. M. Gelfond and V. Lifschitz: Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing 9* (1991) 365-385.
8. Gelfond, M. and Lifschitz, V.: Compiling Circumscriptive Theories into Logic Programs, *Proc. AAA1-88*, 455-459. Extended version in: *Proc. 2nd Int. Workshop on Nonmonotonic Reasoning*, LNAI 346 (1988) 74-99.
9. T. F. Gorden: The Pleadings Game: An Artifical Intelligence Model of Procedural Justice. *Ph.D. thesis, TU Darmstadt* (1993).
10. K. Inoue, M. Koshimura and R. Hasegawa: Embedding negation as failure into a model generation theorem prover, *In: Deepak Kapur, editor, Proceedings of the Eleventh International Conference on Automated Deduction, LNAI 607*, Springer (1992) 400-415.
11. K. Inoue and C. Sakama: On Positive Occurrences of Negation as Failure. *Proc. KR'94* (1994) 293-304.
12. K. Inoue and C. Sakama: Abducing Priorities to Derive Intended Conclusions *Proc. Sixteenth International Joint Conference on Artificial Intelligence* (1999) 44-49.
13. V. Lifschitz: Computing Circumscription. *Proc. IJCAI-85* (1985) 121-127.

14. J. McCarthy: Applications of Circumscription to Formalizing Commonsense Knowledge. *Artificial Intelligence* **28** (1986) 89-116.
15. I. Niemelä and P. Simons: Smodels: An implementation of the stable model and well-founded semantics for normal logic programs. *Proc. the Fourth International Conference on Logic Programming and Nonmonotonic Reasoning*, Springer-Verlag, (1997) 420-429.
16. D. Poole: A Logical framework for default reasoning, *Artificial Intelligence* **36** (1988) 27-47.
17. C. Sakama and K. Inoue: Representing Priorities in Logic Programs. *Proc. Joint International Conference and Symposium on Logic Programming* (1996) 82-96.
18. C. Sakama and K. Inoue: Prioritized logic programming and its application to commonsense reasoning, *Artificial Intelligence 123* (2000) 185-222.

Appendix: Proofs of Theorems

Proof of Theorem 1

Proof: ($\Longrightarrow$) Since $T[P, \Phi, S] = P \cup \Gamma \cup \Pi$ is consistent, it holds that $better \in E$ for any answer set E of $T[P, \Phi, S]$, and E is also an answer set of $T[P, \Phi, S] \setminus \{\leftarrow$ *not better*$\}$. Now, it is easily shown that E should be an augmented answer set of P which not only includes an answer set of P but also has ground head literals of the rules from $\Gamma \cup \Pi \setminus \{\leftarrow$ *not better*$\}$. Thus $S' = E \cap Lit_P$ should be an answer set of P. According to the rule 1, it is obvious that $S^* = E \cap Lit^*_P$ is a renamed answer set of a given answer set S such that each $L^* \in S^*$ is a renamed literal wrt $L \in S$. Then, according to rule 3,

$$m_1(c_t) \in E \quad \text{iff} \quad c^* \in S^* \quad (\text{i.e. } c \in S)$$
$$m_2(d_t) \in E \quad \text{iff} \quad d \in S' \stackrel{def}{=} E \cap Lit_P$$

where $c_t \in \mathcal{C}$ wrt c and $d_t \in \mathcal{C}$ wrt d. Let Φ^* be a closure of Φ. Due to rules 2 , 4 and 5, it holds that,

$$\preceq (a_t, b_t) \in E \quad \text{iff} \quad a \preceq b \in \Phi^*$$

In addition, let Ψ^* be a closure of strict priorities defined as follows:

$$\Psi^* \stackrel{def}{=} \{a \prec b | a \preceq b \in \Phi^* \wedge b \preceq a \notin \Phi^*\}$$

Then, according to rule 6, it holds that,

$$\prec (a_t, b_t) \in E \quad \text{iff} \quad a \prec b \in \Psi^*.$$

Now, according to rules 7 $\sim$ 11, it holds that,

$better \in E$ for any answer set E of $T[P, \Phi, S]$
iff $defeated(c_t) \in E$ for $\exists c_t$
iff $gr_1(c_t, d_t) \in E \wedge attacked(d_t) \notin E$ for $\exists c_t \exists d_t$
iff for $\exists c_t$ s.t. $m_1(c_t) \in E \wedge m_2(c_t) \notin E$ and
$\quad \exists d_t$ s.t. $m_2(d_t) \in E \wedge m_1(d_t) \notin E$,
$\quad \preceq (c_t, d_t) \in E \wedge \prec (d_t, e_t) \notin E$
$\quad\quad$ for $\forall e_t$ s.t. $m_1(e_t) \in E \wedge m_2(e_t) \notin E$
iff for $\exists c \in S \setminus S'$, $\exists d \in S' \setminus S$ such that $c \preceq d \in \Phi^*$,
$\quad$ there is no $e \in S \setminus S'$ such that $d \prec e \in \Psi^*$
iff $S \sqsubseteq S'$ where $S' \stackrel{def}{=} E \cap Lit_P$ and $S \neq S'$

($\Longleftarrow$) Suppose that there is another answer set S' of P such that $S \sqsubseteq S'$. Then, $S \sqsubseteq S'$ should be derived in a way of either case 1. or 2. as follows.

1. $S \sqsubseteq S'$ is directly decided only using priorities in Φ^*.
2. $S \sqsubseteq S'$ is not directly decided using priorities in Φ^*, but is indirectly decided via the transitive law as follows.
   For some other answer set U of P, the following (a) and (b) should be satisfied:
   (a) $S \sqsubseteq U$ is directly decided using priorities in Φ^*.
   (b) $U \sqsubseteq S'$ is decided inductively in a way of either case 1. or case 2..

   Then $S \sqsubseteq S'$ is transitively derived from $S \sqsubseteq U$ and $U \sqsubseteq S'$ according to (a) and (b).

Thus, in the case of either 1. or 2., there exists some answer set V of P which is directly decided to be preferable to S due to priorities in Φ^*. Then, according to Definition 4, there exists some element $d \in V \setminus S$ such that
(i) there is an element $c \in S \setminus V$ such that $c \preceq d$, and
(ii) there is no element $e \in S \setminus V$ such that $d \prec e$.
So, due to the existence of such an element $d \in V \setminus S$, it is easily shown that for such another answer set V of P, $\{L \leftarrow |L \in V\} \cup \Gamma \cup \Pi$ becomes consistent. Therefore, $T[P, \Phi, S] \overset{def}{=} P \cup \Gamma \cup \Pi$ also becomes consistent. □

Proof of Theorem 2
Proof: ($\Longrightarrow$) The contrapositive is proved. That is, in the following, we prove that if S is not a strictly preferred answer set of (P, Φ); $T[P, \Phi, S]$ is consistent.

Suppose that S is not a strictly preferred answer set of (P, Φ). Then S is either (i) a tie-preferred answer set, or (ii) not a preferred answer set. In case of (i), there exists some preferred answer set S' of (P, Φ) such that $S \sqsubseteq S'$ and $S' \sqsubseteq S$ where $S' {\neq} S$ according to Definition 8. In case of (ii), since S is not a preferred answer set of (P, Φ), there should exist some answer set S' of P such that $S \sqsubseteq S'$ and $S' {\not\sqsubseteq} S$ according to Definition 5. Thus in both cases, there exists another preferred answer set S' of P such that $S \sqsubseteq S'$. As a result, we can conclude that $T[P, \Phi, S]$ is consistent according to the proof of $\Leftarrow$ part of Theorem 1.

($\Longleftarrow$) Suppose that S is a strictly preferred answer set of (P, Φ). In the following, assuming that $T[P, \Phi, S]$ is consistent, we show that the contradiction is derived.

Since $T[P, \Phi, S]$ is consistent according to the assumption, $S' = E \cap Lit_P$ should be another answer set of P such that,

$$S \sqsubseteq S' \tag{3}$$

for any answer set E of $T[P, \Phi, S]$ according to the proof of $\Rightarrow$ part of Theorem 1. Then $S \sqsubseteq S'$ for a preferred answer set S of (P, Φ) leads to $S' \sqsubseteq S$ due to Definitions 5. Thus $S' \overset{def}{=} E \cap Lit_P$ should be also another preferred answer set of (P, Φ). On the other hand, since S is a strictly preferred answer set of (P, Φ), it holds that,

$$S {\not\sqsubseteq} S'' \tag{4}$$

for any preferred answer set S'' such that $S \neq S''$ according to Definition 8, which contradicts (3). $\square$