Prioritized logic programming and its application to commonsense reasoning

Chiaki Sakama [a,1] and Katsumi Inoue [b,2]

[a] *Department of Computer and Communication Sciences, Wakayama University, Sakaedani, Wakayama 640 8510, Japan*

[b] *Department of Electrical and Electronics Engineering, Kobe University, Rokkodai, Nada-ku, Kobe 657 8501, Japan*

Abstract

Representing and reasoning with priorities are important in commonsense reasoning. This paper introduces a framework of *prioritized logic programming* (PLP), which has a mechanism of explicit representation of priority information in a program. When a program contains incomplete or indefinite information, PLP is useful for specifying preference to reduce non-determinism in logic programming. Moreover, PLP can realize various forms of commonsense reasoning in AI such as abduction, default reasoning, circumscription, and their prioritized variants. The proposed framework increases the expressive power of logic programming and exploits new applications in knowledge representation.

Keywords: prioritized logic programs, abduction, default reasoning, prioritized circumscription

1 Introduction

In commonsense reasoning a theory is usually assumed incomplete and may contain indefinite or conflicting knowledge. Under such circumstances, priority information is useful to select appropriate knowledge in an incomplete theory and guides us to intended conclusions. For representing and reasoning with priorities, several prioritized systems have been proposed in the field of *nonmonotonic reasoning* (NMR) in AI.

[1] Corresponding author. Tel/Fax: +81 73 457 8128;
e-mail: sakama@sys.wakayama-u.ac.jp
[2] E-mail: inoue@eedept.kobe-u.ac.jp

In *default logic* [48], conflicting default rules produce multiple extensions. Then more specific default rules are preferred to reduce anomalous extensions. Such preference knowledge is implicitly encoded in default rules [49,17], or explicitly specified as priorities between default rules [6,3,12,50]. On the other hand, *circumscription* [42] introduces preference over models. A minimal model which consists of minimal possible extensions of predicates is selected as a preferred model. Further preference between predicates is specified in *prioritized circumscription* [35]. In *abduction*, an observation has more than one explanation in general. To select preferred explanations from many candidates, the simplicity measure is usually adopted as well as other syntactic or semantic criteria [54,14].

Logic programming provides a powerful language for representing and reasoning with commonsense knowledge [4]. Various extensions of logic programming provide mechanisms of handling incomplete and conflicting knowledge in many ways. *Normal logic programs* [40] incorporate *negation as failure* into a program and realize default reasoning. *Disjunctive logic programs* [41] introduce disjunctive rules in a program, which enables us to reason with indefinite information. *Extended logic programs* [20] distinguish default and explicit negation to represent incomplete information in a program. *Abductive logic programs* [32] use hypothetical knowledge to realize abduction in logic programming.

In these extended frameworks, each language introduces different kinds of *non-determinism* as

- multiple minimal models in normal and disjunctive programs,
- multiple explanations in an abductive logic program,
- conflicting answer sets in an extended logic program.

To reduce such non-determinism in programming knowledge, it is useful to introduce a mechanism of explicit representation of priorities to specify the intended meaning of a program. The logic programming languages, however, provide a rather weak mechanism of specifying priorities in a program. When a logic program contains non-Horn clauses, it has multiple minimal models in general. Preference is then introduced to select intended minimal models of a program. However, such preference is defined at the semantic level, and a program itself does not have a mechanism of representing priorities at the syntactic level.[3] To reason with priorities in logic programming, several languages which incorporate priorities into programs emerged quite recently [7–10,13,21,53,56,57].

This paper studies representing and reasoning with priorities in logic programming. We first introduce a framework of *prioritized logic programming*

[3] Stratified negation [2,46] can express priorities between atoms in a restricted manner.

(PLP) which has a mechanism of explicit representation of priorities in a program. The declarative semantics of such programs is given by the *preferred answer sets*, which incorporate priorities into Gelfond and Lifschitz's answer set semantics [20]. Next, we demonstrate that various forms of commonsense reasoning in AI, such as abduction, default reasoning, circumscription, and their prioritized versions, are realized in PLP. We analyze the computational complexity of PLP, and show that the introduction of priorities increases the expressive power of logic programming.

This paper is an extended form of [53]. The rest of this paper is organized as follows. In Section 2, a framework of prioritized logic programming is introduced. Section 3 presents applications of PLP to commonsense reasoning in AI. Section 4 discusses the computational aspect of PLP. Section 5 presents comparisons with related work, and Section 6 concludes the paper.

2 Prioritized logic programs

2.1 General extended disjunctive programs

Logic programs we consider in this paper are *general extended disjunctive programs*. A general extended disjunctive program (GEDP) consists of *rules* of the form:

$$\begin{aligned} &L_1 \mid \cdots \mid L_k \mid not\, L_{k+1} \mid \cdots \mid not\, L_l \\ &\quad \leftarrow L_{l+1}, \ldots, L_m, not\, L_{m+1}, \ldots, not\, L_n \ \ (n \geq m \geq l \geq k \geq 0) \end{aligned} \tag{1}$$

where each L_i is a positive or negative literal. "|" represents a disjunction and *not* means *negation as failure* (NAF). The disjunction to the left of $\leftarrow$ is the *head* and the conjunction to the right of $\leftarrow$ is the *body* of the rule. A rule with the empty head is called an *integrity constraint*. A *ground rule* is a rule having no variable. A rule with variables stands for the set of its ground instances, i.e., the set of ground rules obtained by substituting variables with elements of the Herbrand universe of a program in every possible way.

Intuitively, the rule (1) is read as: if all $L_{l+1}, \ldots, L_m$ are believed and all $L_{m+1}, \ldots, L_n$ are disbelieved, then either some L_i $(1 \leq i \leq k)$ should be believed or some L_j $(k+1 \leq j \leq l)$ should be disbelieved. The class of GEDPs is introduced in [37,26] as a subclass of *minimal belief and negation as failure* (MBNF) [38]. GEDPs are a fairly general class of existing LP languages in the sense that it includes the so-called *normal*, *disjunctive* and *extended logic programs*. Moreover, it can also express the class of *abductive logic programs*, which will be discussed in the next section. A GEDP is called an *extended*

disjunctive program (EDP) if it contains no *not* in the head of any rule (i.e., $k = l$). An EDP is called a *normal disjunctive program* (NDP) if every L_i in the program is an atom; and an EDP is called an *extended logic program* (ELP) if it contains no disjunction ($l \leq 1$). We say that a set of ground literals S *satisfies* a ground rule of the form (1) if $\{L_{l+1}, \ldots, L_m\} \subseteq S$ and $\{L_{m+1}, \ldots, L_n\} \cap S = \emptyset$ imply either $\{L_1, \ldots, L_k\} \cap S \neq \emptyset$ or $\{L_{k+1}, \ldots, L_l\} \setminus S \neq \emptyset$. Also, S satisfies the conjunction $L_1, \ldots, L_m, \mathit{not}\, L_{m+1}, \ldots, \mathit{not}\, L_n$ if $\{ L_1, \ldots, L_m \} \subseteq S$ and $\{ L_{m+1}, \ldots, L_n \} \cap S = \emptyset$.

The semantics of GEDPs is given by the *answer sets*. The following definition is due to [26]. First, let P be a *not*-free GEDP (i.e., $k = l$ and $m = n$) and $S \subseteq \mathcal{L}_P$, where $\mathcal{L}_P$ is the set of all ground literals in the language of P. Then, S is an *answer set* of P if S is a minimal set satisfying the following two conditions:

(i) S satisfies every rule in P, i.e., for each ground rule

$$L_1 \mid \cdots \mid L_l \leftarrow L_{l+1}, \ldots, L_m \ \ (l \geq 1)$$

from P, $\{L_{l+1}, \ldots, L_m\} \subseteq S$ implies $L_i \in S$ for some i $(1 \leq i \leq l)$. In particular, for each ground integrity constraint $\leftarrow L_1, \ldots, L_m$ from P, $\{L_1, \ldots, L_m\} \not\subseteq S$ holds;

(ii) If S contains a pair of complementary literals L and $\neg L$, then $S = \mathcal{L}_P$.

Secondly, given any GEDP P and $S \subseteq \mathcal{L}_P$, consider the not-free GEDP P^S (called a *reduct*) obtained as follows: a rule

$$L_1 \mid \cdots \mid L_k \leftarrow L_{l+1}, \ldots, L_m$$

is in P^S if there is a ground rule of the form (1) from P such that

$$\{L_{k+1}, \ldots, L_l\} \subseteq S \ \text{ and } \{L_{m+1}, \ldots, L_n\} \cap S = \emptyset.$$

Then, S is an *answer set* of P if S is an answer set of P^S. Every answer set of a GEDP P satisfies every ground rule from P [26]. An answer set is *consistent* if it is not $\mathcal{L}_P$. The answer set $\mathcal{L}_P$ is said *contradictory*. A GEDP is *consistent* if it has a consistent answer set; otherwise, the program is *inconsistent*. An answer set S of a GEDP P is *minimal* if there is no other answer set S' of P s.t. $S' \subset S$. The set of all answer sets of P is written as $\mathcal{AS}_P$.

The above definition of answer sets reduces to that of Gelfond and Lifschitz [20] in an EDP. Note that every answer set of any EDP is minimal [20,37], but the minimality of answer sets no longer holds for GEDPs. For example, suppose a program with the single rule

$$L \mid \mathit{not}\, L \leftarrow,$$

saying, L is true or not. Then, it has two answer sets $\{L\}$ and $\emptyset$.

2.2 Prioritized logic programs

Next we introduce a prioritization mechanism to a program. Given a GEDP P and the set of ground literals $\mathcal{L}_P$, we define $\mathcal{L}_P^* = \mathcal{L}_P \cup \{\, not\, L : L \in \mathcal{L}_P \,\}$. Then a pre-order relation $\preceq$, which is reflexive and transitive, is defined on $\mathcal{L}_P^*$.

Definition 2.1 (priorities) For any elements e_1 and e_2 from $\mathcal{L}_P^*$, if $e_1 \preceq e_2$ then we say that e_2 *has a higher priority than* e_1. $e_1 \prec e_2$ stands for $e_1 \preceq e_2$ and $e_2 \not\preceq e_1$. The statement $e_1 \preceq e_2$ is called a *priority*. A relation over elements including variables is defined as follows. For tuples $\mathbf{x}$ and $\mathbf{y}$ of variables, the statement $p_1(\mathbf{x}) \preceq p_2(\mathbf{y})$ stands for every priority $p_1(\mathbf{s}) \preceq p_2(\mathbf{t})$ for any instances $\mathbf{s}$ of $\mathbf{x}$ and $\mathbf{t}$ of $\mathbf{y}$.

Note that if there is a priority $e_1 \prec e_2$, e_1 and e_2 do not have common instances. For example, there is no priority like $p(x, a) \prec p(b, y)$ because $p(b, a) \not\prec p(b, a)$.

Given a set Φ of priorities, we define the closure Φ^* as the set of priorities which are reflexively or transitively derived using priorities in Φ.

Definition 2.2 (prioritized logic program) A *prioritized logic program* (PLP) is defined as a pair (P, Φ) where P is a GEDP and Φ is a set of priorities over $\mathcal{L}_P^*$. [4]

The declarative semantics of a PLP is defined using answer sets. In what follows, for any sets $S \subseteq \mathcal{L}_P$ and $T \subseteq \mathcal{L}_P$, and for any ground literal L, $L \in S \setminus T$ means $L \in S$ and $L \notin T$; and $not\, L \in S \setminus T$ means $L \notin S$ and $L \in T$.

Definition 2.3 (preference between answer sets) Given a PLP (P, Φ), the relation $\sqsubseteq$ is defined over the answer sets of P as follows. For any answer sets S_1, S_2, and S_3 of P,

(i) $S_1 \sqsubseteq S_1$.
(ii) $S_1 \sqsubseteq S_2$ if

$$\exists e_2 \in S_2 \setminus S_1 \,[\, \exists e_1 \in S_1 \setminus S_2 \text{ s.t. } (e_1 \preceq e_2) \in \Phi^* \\ \wedge \neg \exists e_3 \in S_1 \setminus S_2 \text{ s.t. } (e_2 \prec e_3) \in \Phi^* \,].$$

(iii) If $S_1 \sqsubseteq S_2$ and $S_2 \sqsubseteq S_3$, then $S_1 \sqsubseteq S_3$.

[4] We abuse the term PLP for representing both prioritized logic *programming* and prioritized logic *program*. For the latter case, it is used as a countable noun.

We say that S_2 is *preferable* to S_1 wrt Φ if $S_1 \sqsubseteq S_2$ holds. We write $S_1 \sqsubset S_2$ if $S_1 \sqsubseteq S_2$ and $S_2 \not\sqsubseteq S_1$.

By the definition, $S_1 \sqsubseteq S_2$ holds iff $S_2 \setminus S_1$ has an element e_2 whose priority is higher than some element e_1 in $S_1 \setminus S_2$, and $S_1 \setminus S_2$ does not have another element e_3 whose priority is strictly higher than e_2. In particular, the condition $(\neg\exists e_3 \in S_1 \setminus S_2 \text{ s.t. } (e_2 \prec e_3) \in \Phi^*)$ of (ii) is automatically satisfied if there is no priority chained over more than two different elements (i.e., $e_1 \preceq e_2 \preceq e_3$ implies either $e_1 = e_2$ or $e_2 = e_3$).

Example 2.1 Let (P, Φ) be the PLP such that

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

Then, $\{p, r\}$ and $\{q\}$ are two answer sets of P, and $\{q\} \sqsubseteq \{p, r\}$. Note that $\{p, r\} \not\sqsubseteq \{q\}$ by the presence of $q \preceq r$ in Φ.

Definition 2.4 (preferred answer set) Let (P, Φ) be a PLP. Then, an answer set S of P is called a *preferred answer set* (or *p-answer set*, for short) of (P, Φ) if $S \sqsubseteq S'$ implies $S' \sqsubseteq S$ (wrt Φ) for any answer set S' of P. The set of all p-answer sets of (P, Φ) is written as $\mathcal{PAS}_{(P,\Phi)}$.

Intuitively, the p-answer sets are answer sets including elements with the highest priorities wrt Φ. By the definition, (P, Φ) has a p-answer set if P has a finite number of answer sets.

A PLP and p-answer sets are useful when a program has multiple answer sets and a reasoner wants to filter them out according to her preference. For instance, indefinite information in a disjunctive logic program is reduced by the prioritization mechanism of PLP.

Example 2.2 Let P_0 be the program

$$
\begin{aligned}
& \textit{battery-dead} \mid \textit{ignition-damaged} \leftarrow \textit{turn-key}, \neg\textit{start}, \\
& \textit{turn-key} \leftarrow, \\
& \neg\textit{start} \leftarrow,
\end{aligned}
$$

where the first rule attributes the failure of starting a car to a battery or an ignition. Now a reasoner empirically knows that an ignition causes a problem less frequently than a battery. This situation is expressed by the priority

$$\Phi :\ \textit{ignition-damaged} \preceq \textit{battery-dead}\,.$$

Then, the p-answer set of (P_0, Φ) becomes $S = \{\, \mathit{turn\text{-}key},\ \neg \mathit{start},\ \mathit{battery\text{-}dead} \,\}$.

Note that the above situation is also expressed using negation as failure. Suppose the program P_1 which is obtained from P_0 by rewriting the first rule with

$$\mathit{battery\text{-}dead} \leftarrow \mathit{turn\text{-}key},\ \neg \mathit{start},\ \mathit{not}\ \mathit{ignition\text{-}damaged}.$$

Then, S becomes the answer set of the program P_1. However, such a trick is not useful in dynamically changing situations. Suppose that the reasoner later finds that the car-radio works and there is the integrity constraint

$$IC: \quad \leftarrow \mathit{battery\text{-}dead},\ \mathit{radio\text{-}work},$$

saying that a radio does not work with a dead battery. Let

$$P_2 = P_1 \cup \{\, \mathit{radio\text{-}work} \leftarrow \,\} \cup \{IC\}\,.$$

Then it is impossible to get the alternative solution *ignition-damaged* from P_2. By contrast, using PLP the p-answer set of

$$P_3 = P_0 \cup \{\, \mathit{radio\text{-}work} \leftarrow \,\} \cup \{IC\}$$

becomes $\{\, \mathit{turn\text{-}key},\ \neg \mathit{start},\ \mathit{radio\text{-}work},\ \mathit{ignition\text{-}damaged} \,\}$, as intended.

Thus PLP can naturally specify prioritized knowledge, and can select appropriate answer sets according to the change of situations. Note that any knowledge which is irrelevant to preference is not affected by the selection of p-answer sets. For example, consider the program P_4 which is obtained from P_0 by replacing the first disjunctive rule with

$$\mathit{battery\text{-}dead} \mid \mathit{ignition\text{-}damaged} \mid \mathit{cold\text{-}morning} \leftarrow \mathit{turn\text{-}key},\ \neg \mathit{start},$$

where *cold-morning* has no priority over the other two disjuncts. Then, (P_4, Φ) has the p-answer set $\{\, \mathit{turn\text{-}key},\ \neg \mathit{start},\ \mathit{cold\text{-}morning} \,\}$ in addition to S.

2.3 Properties of PLP

The p-answer sets of PLPs extend the answer sets of GEDPs.

Proposition 2.1 *(relation between answer sets and p-answer sets) Let (P, Φ) be a PLP. Then, $\mathcal{PAS}_{(P,\Phi)} \subseteq \mathcal{AS}_P$. In particular, $\mathcal{PAS}_{(P,\emptyset)} = \mathcal{AS}_P$.*

Thus, the answer sets of a program are characterized as a special case of the p-answer sets of a PLP with empty priorities. It is also clear that if a program P has the unique answer set, it also becomes the unique p-answer set of (P, Φ) for any Φ.

The above proposition presents that introducing priorities reduces the number of possible solutions in general. However, such reduction is not necessarily monotonic, i.e., increasing priorities in a PLP does not always decrease the number of p-answer sets.

Proposition 2.2 *(nonmonotonicity) Let* (P, Φ_1) *and* (P, Φ_2) *be two PLPs. Then,* $\Phi_1 \subseteq \Phi_2$ *does not imply* $\mathcal{PAS}_{(P,\Phi_2)} \subseteq \mathcal{PAS}_{(P,\Phi_1)}$.

Example 2.3 Let P be the program

$$
\begin{aligned}
&p \mid q \leftarrow, \\
&q \mid r \leftarrow, \\
&\leftarrow q,\, r,
\end{aligned}
$$

and $\Phi_1 = \emptyset$, $\Phi_2 = \{\, p \preceq q \,\}$, and $\Phi_3 = \{\, p \preceq q,\ q \preceq r \,\}$. Then (P, Φ_1) has the p-answer sets $\{p, r\}$ and $\{q\}$; (P, Φ_2) has $\{q\}$; and (P, Φ_3) has $\{p, r\}$.

As an example of the above program, consider the following situation. There are three different medicines p, q, and r. A patient has to take either p or q, and either q or r. Also, it is known that taking q and r together causes side effects (hence they should not be taken together). With the empty priorities Φ_1, there are two possibilities of taking $\{p, r\}$ or $\{q\}$. If it is known that the medicine q is more effective than p, she prefers taking $\{q\}$ under the priority Φ_2. Later, the medicine r is known as the best one as in Φ_3, then $\{p, r\}$ is the best choice.

In the above example, $\{q\}$ is selected as far as Φ_2 is concerned, while the selection is changed when more information Φ_3 is available. Thus, p-answer sets characterize the situation in which previous beliefs may possibly be rebutted according to the change of priorities.

In PLPs priority relations are defined over elements from $\mathcal{L}^*_P$, but they are used to express priorities over more general forms of knowledge.

▷ **Priorities between conjunctive knowledge** :
Suppose that a priority relation exists between conjunctions of elements:

$$(e_1, \ldots, e_m) \preceq (e'_1, \ldots, e'_n)$$
(or sets of elements $\{\, e_1, \ldots, e_m \,\} \preceq \{\, e'_1, \ldots, e'_n \,\}$).

Then it is expressed in a PLP (P, Φ) by introducing the rules

$$e_0 \leftarrow e_1, \ldots, e_m \ \text{ and } \ e'_0 \leftarrow e'_1, \ldots, e'_n$$

to P with the newly introduced atoms e_0 and e'_0, and the priority $e_0 \preceq e'_0$ in Φ.

▷ **Priorities between disjunctive knowledge** :
Suppose that a priority relation exists between disjunctions of elements:

$$(e_1 \mid \cdots \mid e_m) \preceq (e'_1 \mid \cdots \mid e'_n).$$

Then it is expressed in a PLP (P, Φ) by introducing the rules

$$e_0 \leftarrow e_i \ (\text{for } i = 1, \ldots, m) \ \text{ and } \ e'_0 \leftarrow e'_j \ (\text{for } j = 1, \ldots, n)$$

to P with the newly introduced atoms e_0 and e'_0, and the priority $e_0 \preceq e'_0$ in Φ.

▷ **Priorities with preconditions** :
Suppose that a priority relation holds under some condition Γ:

$$(e_1 \preceq e_2) \leftarrow \Gamma.$$

Then it is expressed in a PLP (P, Φ) by introducing the rules

$$e'_1 \leftarrow e_1, \Gamma \ \text{ and } \ e'_2 \leftarrow e_2, \Gamma$$

to P with the newly introduced atoms e'_1 and e'_2, and the priority $e'_1 \preceq e'_2$ in Φ.

▷ **Priorities between rules** :
Suppose that a priority relation exists between (conflicting) rules in P:

$$(H_1 \leftarrow B_1) \preceq (H_2 \leftarrow B_2)\,.$$

Then it is expressed in a PLP (P, Φ) by introducing the rules

$$r_1 \leftarrow B_1 \ \text{ and } r_2 \leftarrow B_2$$

to P with the newly introduced atoms r_1 and r_2, and the priority $r_1 \preceq r_2$ in Φ.

We illustrate the above third and fourth cases using examples.

Example 2.4 A person drinks tea or coffee ($tea \mid coffee \leftarrow$), but she prefers coffee to tea when sleepy ($(tea \preceq coffee) \leftarrow sleepy$). Such a conditional priority can be encoded in a PLP as follows. Assume that ($sleepy \leftarrow$) holds. Then, the (P, Φ) with

P : $tea \mid coffee \leftarrow$,
$tea' \leftarrow tea,\ sleepy$,
$coffee' \leftarrow coffee,\ sleepy$,
$sleepy \leftarrow$.
Φ : $tea' \preceq coffee'$.

has the p-answer set $\{\, sleepy,\ coffee,\ coffee' \,\}$. Next, if it turns that no coffee is available, then the PLP $(P \cup \{\, \neg coffee \leftarrow \,\}, \Phi)$ has the p-answer set $\{\, sleepy,\ tea,\ tea',\ \neg coffee \,\}$. Thus, PLP chooses an appropriate answer set according to the change of situations.

Example 2.5 Let P be the program

$innocent \leftarrow not\ guilty$,
$guilty \leftarrow not\ innocent$.

If one is presumed innocent unless proven otherwise, the first rule is preferred to the second one. The situation is expressed in the PLP (P, Φ) as

P : $innocent \leftarrow not\ guilty$,
$guilty \leftarrow not\ innocent$,
$r_{innocent} \leftarrow not\ guilty$,
$r_{guilty} \leftarrow not\ innocent$.
Φ : $r_{guilty} \preceq r_{innocent}$.

Then, (P, Φ) has the p-answer set $\{\, innocent,\ r_{innocent} \,\}$, which corresponds to the solution by the first rule.

As shown above, priorities between rules are expressed in terms of priorities between atoms. However, this transformation does not work well when a program is inconsistent.

Example 2.6 Let P be the program

$flies \leftarrow bird$,
$\neg flies \leftarrow penguin$,
$bird \leftarrow penguin$,
$penguin \leftarrow$,

which has the contradictory answer set $\mathcal{L}_P$. If the second more specific rule is preferred to the first more general one, introducing the rules

$r_{flies} \leftarrow bird$,

$r_{\neg flies} \leftarrow penguin$

and the priority $r_{flies} \preceq r_{\neg flies}$ is of no use. In fact, the transformed program also has the answer set $\mathcal{L}_P$.

In the above example, the first rule is usually regarded as a defeasible *default* rule. Specifying priorities between conflicting default rules will be discussed in Section 3.2.2.

3 Commonsense Reasoning in PLP

In this section, we present applications of PLP to commonsense reasoning in AI.

3.1 Abduction

Abduction is inference to explanations and is realized by *abductive logic programming*. We first review the framework of abductive logic programming in terms of GEDPs.

Definition 3.1 [26] (abductive logic program) Let P be a GEDP and $\mathcal{A}$ a set of literals called *abducibles*. Then, an *abductive logic program* (ALP) is represented as a GEDP

$$\Pi = P \cup \{\, A \mid not\, A \leftarrow :\ A \in \mathcal{A} \,\}. \tag{2}$$

The set $\mathcal{A}$ is identified with the set of ground instances from $\mathcal{A}$, and any instance of an element from $\mathcal{A}$ is also called an abducible. Let Π be an ALP and O a ground literal which represents an *observation*.[5] Then, a set $E \subseteq \mathcal{A}$ is an *explanation*[6] of O in Π if there is a consistent answer set S of Π such that $E = S \cap \mathcal{A}$ and $O \in S$.

E is an explanation of O in Π iff S is a consistent answer set of $\Pi \cup \{ \leftarrow not\, O \}$ such that $E = S \cap \mathcal{A}$ [26].

In the above definition, additional disjunctive rules in (2) mean that "an abducible A is assumed or not". Then, with the constraint $\leftarrow not\, O$ asserting

[5] Without loss of generality an observation is assumed to be a (non-abducible) ground literal [29].

[6] Explanations considered here are *credulous* or *brave* explanations [15].

"O should hold", an answer set of $\Pi \cup \{ \leftarrow not\, O \}$ contains abducibles which constitute an explanation of O.

Example 3.1 Let Π be the program

$$
\begin{array}{l}
\mathit{wet\text{-}shoes} \leftarrow \mathit{wet\text{-}grass}, \\
\mathit{wet\text{-}grass} \leftarrow \mathit{rained}, \\
\mathit{wet\text{-}grass} \leftarrow \mathit{sprinkler\text{-}on}, \\
\mathit{rained} \mid not\ \mathit{rained} \leftarrow, \\
\mathit{sprinkler\text{-}on} \mid not\ \mathit{sprinkler\text{-}on} \leftarrow,
\end{array}
$$

where *rained* and *sprinkler-on* are abducibles. Then, given the observation $O = \mathit{wet\text{-}shoes}$, the program $\Pi \cup \{ \leftarrow not\, O \}$ has three answer sets

$$
\begin{array}{l}
\{ \mathit{wet\text{-}shoes}, \mathit{wet\text{-}grass}, \mathit{rained} \}, \\
\{ \mathit{wet\text{-}shoes}, \mathit{wet\text{-}grass}, \mathit{sprinkler\text{-}on} \}, \\
\{ \mathit{wet\text{-}shoes}, \mathit{wet\text{-}grass}, \mathit{rained}, \mathit{sprinkler\text{-}on} \},
\end{array}
$$

which imply that $\{ \mathit{rained} \}$, $\{ \mathit{sprinkler\text{-}on} \}$, $\{ \mathit{rained}, \mathit{sprinkler\text{-}on} \}$ are the possible explanations of O.

3.1.1 Minimal abduction

In abduction, selecting *best explanations* from many candidate explanations is particularly important. In this respect, *minimal* explanations are usually preferred as simplest hypotheses to explain an observation. An explanation E is *minimal* if no $E' \subset E$ is an explanation. Such minimal abduction is expressed in PLP as follows.

Definition 3.2 (minimal abduction) Given an ALP Π and an observation O, *minimal abduction* is defined as a PLP (Π, Φ_{MA}) where

$$\Phi_{MA} = \{ A \preceq not\, A \,:\, A \in \mathcal{A} \}.$$

In Φ_{MA}, the priority $A \preceq not\, A$ is read as "A is less likely to happen". This priority condition has the effect of eliminating an abducible A in each p-answer set whenever possible. An answer set S is called $\mathcal{A}$-*minimal* if there is no answer set S' such that $S' \cap \mathcal{A} \subset S \cap \mathcal{A}$. Then the following results hold.

Lemma 3.1 *[26] (minimal explanation vs. $\mathcal{A}$-minimal answer set) Let Π be an ALP and O an observation. Then, O has a minimal explanation E in Π iff $\Pi \cup \{ \leftarrow not\, O \}$ has a consistent $\mathcal{A}$-minimal answer set S such that $E = S \cap \mathcal{A}$.*

Theorem 3.2 *(minimal abduction in PLP) Let* (Π, Φ_{MA}) *be a PLP representing minimal abduction. Then, an observation* O *has a minimal explanation* E *in* Π *iff* $(\Pi \cup \{ \leftarrow \mathit{not}\, O \}, \Phi_{MA})$ *has a consistent p-answer set* S *such that* $E = S \cap \mathcal{A}$.

Proof. By Lemma 3.1, it is enough to show that S is a consistent $\mathcal{A}$-minimal answer set of $\Pi \cup \{ \leftarrow \mathit{not}\, O \}$ iff S is a consistent p-answer set of $(\Pi \cup \{ \leftarrow \mathit{not}\, O \}, \Phi_{MA})$.
Put $\Pi' = \Pi \cup \{ \leftarrow \mathit{not}\, O \}$ and let S be a consistent answer set of Π'. Then,
S is a consistent $\mathcal{A}$-minimal answer set of Π'
iff for any consistent answer set T of Π', $\exists A \in (S \setminus T) \cap \mathcal{A}$ implies $\exists A' \in (T \setminus S) \cap \mathcal{A}$, because otherwise $T \cap \mathcal{A} \subset S \cap \mathcal{A}$
iff for any consistent answer set T of Π', $\exists A \in \mathcal{A}$ s.t. $(A \in S \setminus T$ and $\mathit{not}\, A \in T \setminus S)$ implies $\exists A' \in \mathcal{A}$ s.t. $(A' \in T \setminus S$ and $\mathit{not}\, A' \in S \setminus T)$
iff for any consistent answer set T of Π', $S \sqsubseteq T$ implies $T \sqsubseteq S$ wrt Φ_{MA}
iff S is a consistent p-answer set of (Π', Φ_{MA}). □

Example 3.2 In Example 3.1, let $\Phi_{MA} = \{ \mathit{sprinkler\text{-}on} \preceq \mathit{not\ sprinkler\text{-}on}, \mathit{rained} \preceq \mathit{not\ rained} \}$. Then, $(\Pi \cup \{ \leftarrow \mathit{not}\, O \}, \Phi_{MA})$ has two p-answer sets $\{ \mathit{wet\text{-}shoes}, \mathit{wet\text{-}grass}, \mathit{rained} \}$ and $\{ \mathit{wet\text{-}shoes}, \mathit{wet\text{-}grass}, \mathit{sprinkler\text{-}on} \}$, which imply the minimal explanations $\{ \mathit{rained} \}$ and $\{ \mathit{sprinkler\text{-}on} \}$, respectively.

3.1.2 Prioritized abduction

Although minimal abduction reduces the number of possible explanations, it is not strong enough to select intended explanations. In fact, an abductive logic program generally has multiple minimal explanations. To specify further priorities between minimal explanations, we apply the priority relation $\preceq$ to abducibles and apply the relation $\sqsubseteq$ to explanations.

Definition 3.3 (priority over abducibles) For any abducibles A_1 and A_2 from $\mathcal{A}$, if $A_1 \preceq A_2$ we say that A_2 has a *higher priority than* A_1. Let $\Phi_{\mathcal{A}}$ be a set of priorities over abducibles. For two sets $E \subseteq \mathcal{A}$ and $F \subseteq \mathcal{A}$, $E \sqsubseteq F$ is defined as in Definition 2.3 wrt the priorities in $\Phi_{\mathcal{A}}$.

Definition 3.4 (preferred minimal explanation) Let Π be an ALP and $\Phi_{\mathcal{A}}$ a set of priorities over abducibles. Given an observation O, a minimal explanation E of O is called a *preferred (minimal) explanation* if $E \sqsubseteq F$ implies $F \sqsubseteq E$ (wrt $\Phi_{\mathcal{A}}$) for any minimal explanation F of O.

By the definition, a minimal explanation is preferred if it contains an abducible with a relatively higher priority than those in any other explanation. In particular, if an ALP has the unique minimal explanation, it is always the

preferred explanation.

Definition 3.5 (prioritized minimal abduction) Let (Π, Φ_{MA}) be a PLP representing minimal abduction. Given a set $\Phi_{\mathcal{A}}$ of priorities over abducibles, *prioritized minimal abduction* is defined as a PLP (Π, Φ_{PMA}) where

$$\Phi_{PMA} = \Phi_{MA} \cup \{\, not\, A_i \preceq not\, A_j \ :\ (A_j \preceq A_i) \in \Phi_{\mathcal{A}} \,\}.$$

In the definition, the additional priority $not\ A_i \preceq not\ A_j$ is read "an abducible A_j is less likely to happen than A_i". Introducing this priority to Φ_{MA}, any p-answer set S satisfying '$not\, A_j$' is preferred. Thus, preferred minimal explanations are computed by prioritized minimal abduction.

Theorem 3.3 *(preferred minimal explanation vs. prioritized minimal abduction) Let Π be an ALP, $\Phi_{\mathcal{A}}$ a set of priorities over abducibles, and O an observation. Then, E is a preferred minimal explanation of O iff $(\Pi \cup \{\leftarrow\ not\, O\}, \Phi_{PMA})$ has a consistent p-answer set S such that $E = S \cap \mathcal{A}$.*

Proof. Put $\Pi' = \Pi \cup \{\leftarrow not\, O\}$. Then, E is a preferred minimal explanation of O
iff E is a minimal explanation of O and for any minimal explanation F of O, $E \sqsubseteq F$ implies $F \sqsubseteq E$ (wrt $\Phi_{\mathcal{A}}$)
iff S is a consistent p-answer set of (Π', Φ_{MA}) with $E = S \cap \mathcal{A}$ (Theorem 3.2), and for any consistent p-answer set T of (Π', Φ_{MA}) with $F = T \cap \mathcal{A}$, $S \cap \mathcal{A} \sqsubseteq T \cap \mathcal{A}$ implies $T \cap \mathcal{A} \sqsubseteq S \cap \mathcal{A}$ (wrt $\Phi_{\mathcal{A}}$), hence $S \sqsubseteq T$ implies $T \sqsubseteq S$ (wrt Φ_{PMA})
iff S is a consistent p-answer set of (Π', Φ_{PMA}) with $E = S \cap \mathcal{A}$. □

Example 3.3 In Example 3.2 suppose that a reasoner does not use the sprinkler, hence a good reason exists to prefer '*not sprinkler-on*' to '*not rained*'. The situation is represented using the prioritized minimal abduction (Π, Φ_{PMA}) where Φ_{PMA} contains the priority

$$not\ rained \preceq not\ sprinkler\text{-}on\,,$$

together with the priorities in Φ_{MA}. Then, the PLP $(\Pi \cup \{\leftarrow not\, O\}, \Phi_{PMA})$ has the unique p-answer set $\{\, wet\text{-}shoes, wet\text{-}grass, rained \,\}$, which implies the preferred minimal explanation $\{\, rained \,\}$.

3.2.1 Knowledge system

Default reasoning is a form of reasoning with incomplete information. Poole [44] proposed a simple framework for default reasoning, which is reformulated by Inoue [28] in the context of logic programming as follows.

A *knowledge system* is defined as a pair $K = (P, \Delta)$ where P and Δ are EDPs representing *facts* and *defaults*, respectively.[7] A fact or default containing no variable is called *ground*. Given $K = (P, \Delta)$, an *extension base* is defined as a consistent answer set of $P \cup D$ where D is a maximal subset of the ground instances of elements from Δ.

Example 3.4 Let $K_1 = (P_1, \Delta_1)$ be the knowledge system such that

$$
\begin{aligned}
P_1 : & \neg flies(x) \leftarrow penguin(x), \\
& bird(x) \leftarrow penguin(x), \\
& bird(polly) \leftarrow, \;\; penguin(tweety) \leftarrow . \\
\Delta_1 : & \; flies(x) \leftarrow bird(x).
\end{aligned}
$$

Then K_1 has the unique extension base $S = \{\, bird(polly),\, penguin(tweety),\, bird(tweety),\, flies(polly),\, \neg flies(tweety) \,\}$. Note that the default rule in Δ_1 is applied for $x = polly$ but not for $x = tweety$, since $P_1 \cup \{flies(tweety)\}$ is inconsistent.

In abduction, minimal hypotheses are preferred to explain an observation. By contrast, in default reasoning hypotheses are assumed as many as possible unless they cause contradiction.

To formulate default reasoning in PLP, we define the PLP expression of a knowledge system.

Definition 3.6 (knowledge system in PLP) Given a knowledge system $K = (P, \Delta)$, its PLP expression (Π, Φ_{KS}) is defined as follows.

(i) Any rule in P is included in Π.
(ii) Any rule $Head \leftarrow Body$ in Δ is transformed to the rules

$$Head \leftarrow \delta(\mathbf{x}),\, Body, \tag{3}$$
$$\delta(\mathbf{x}) \mid not\, \delta(\mathbf{x}) \leftarrow \tag{4}$$

[7] [28] introduces K with ELPs P and Δ. Gelfond [19] introduces a similar system with EDPs.

in Π, where $\mathbf{x}$ represents variables appearing in the rule, and $\delta(\mathbf{x})$ is a newly introduced atom uniquely associated with each rule from Δ.

(iii) For any $\delta(\mathbf{x})$ introduced above, the priority $not\,\delta(\mathbf{x}) \preceq \delta(\mathbf{x})$ is in Φ_{KS}.

In the above transformation, the rule (4) says that the corresponding default rule (3) is effective or not, and priorities in Φ_{KS} express that default rules *normally* hold. In this way, PLP can represent a knowledge system in a single program Π together with priorities Φ_{KS}.

Let $\mathcal{D}$ be the set of ground instances of any atom $\delta(\mathbf{x})$ in Π. An answer set S is called $\mathcal{D}$-*maximal* if there is no answer set S' such that $S \cap \mathcal{D} \subset S' \cap \mathcal{D}$. Let $\mathcal{L}_K$ be the set of all ground literals in the language of K. Then the following results hold.

Lemma 3.4 *(extension base vs. $\mathcal{D}$-maximal answer set) Let $K = (P, \Delta)$ be a knowledge system and Π the transformed program as above. If S is a consistent $\mathcal{D}$-maximal answer set of Π, there is an extension base T of K such that $T = S \cap \mathcal{L}_K$. Conversely, if T is an extension base of K, there is a consistent $\mathcal{D}$-maximal answer set S of Π such that $S \cap \mathcal{L}_K = T$.*

Proof. If S is a consistent $\mathcal{D}$-maximal answer set of Π, S is a consistent answer set of Π and for any consistent answer set S' of Π, $\delta_1 \in S' \setminus S$ implies $\delta_2 \in S \setminus S'$ for some $\delta_2 \in \mathcal{D}$, because otherwise $S \cap \mathcal{D} \subset S' \cap \mathcal{D}$. Then, it holds that $T = S \cap \mathcal{L}_K$ is a consistent answer set of $P \cup D$ with some $D \subseteq \Delta$, and for any consistent answer set T' of $P \cup D'$ with $D' \subseteq \Delta$, $d' \in D' \setminus D$ implies $d \in D \setminus D'$ for some ground defaults d and d'. Hence, T is a consistent answer set of $P \cup D$ where D is a maximal subset of the ground instances of elements from Δ. The converse is shown in a similar manner. □

Theorem 3.5 *(extension base vs. p-answer set) Let $K = (P, \Delta)$ be a knowledge system and (Π, Φ_{KS}) its PLP expression. If S is a consistent p-answer set of (Π, Φ_{KS}), there is an extension base T of K such that $T = S \cap \mathcal{L}_K$. Conversely, if T is an extension base of K, there is a consistent p-answer set S of (Π, Φ_{KS}) such that $S \cap \mathcal{L}_K = T$.*

Proof. By Lemma 3.4, it is enough to show that S is a $\mathcal{D}$-maximal consistent answer set of Π iff S is a consistent p-answer set of (Π, Φ_{KS}).
S is a $\mathcal{D}$-maximal consistent answer set of Π
iff S is a consistent answer set of Π and for any consistent answer set S' of Π, $\delta_1 \in S' \setminus S$ and $(not\,\delta_1) \in S \setminus S'$ imply $\delta_2 \in S \setminus S'$ and $(not\,\delta_2) \in S' \setminus S$ for any $\delta_1, \delta_2 \in \mathcal{D}$. As $(not\,\delta_1 \preceq \delta_1) \in \Phi_{KS}$ and $(not\,\delta_2 \preceq \delta_2) \in \Phi_{KS}$, $\delta_1 \in S' \setminus S$ and $(not\,\delta_1) \in S \setminus S'$ iff $S \sqsubseteq S'$; and $\delta_2 \in S \setminus S'$ and $(not\,\delta_2) \in S' \setminus S$ iff $S' \sqsubseteq S$.
Thus, S is a $\mathcal{D}$-maximal consistent answer set of Π
iff S is a consistent answer set of Π and for any consistent answer set S' of Π, $S \sqsubseteq S'$ implies $S' \sqsubseteq S$ wrt Φ_{KS}
iff S is a consistent p-answer set of (Π, Φ_{KS}). □

Example 3.5 The knowledge system K_1 of Example 3.4 is expressed in the PLP (Π, Φ_{KS}) as

$$
\begin{aligned}
\Pi : \; & \neg flies(x) \leftarrow penguin(x), \\
& bird(x) \leftarrow penguin(x), \\
& bird(polly) \leftarrow, \;\; penguin(tweety) \leftarrow, \\
& flies(x) \leftarrow \delta(x),\, bird(x), \\
& \delta(x) \mid not\, \delta(x) \leftarrow . \\
\Phi_{KS} : \; & not\, \delta(x) \preceq \delta(x) .
\end{aligned}
$$

Then (Π, Φ_{KS}) has the unique p-answer set $\{\, bird(polly),\, penguin(tweety),\, bird(tweety),\, flies(polly),\, \neg flies(tweety),\, \delta(polly) \,\}$, which corresponds to the extension base S of K_1.

3.2.2 Prioritized default reasoning

A default theory generally has multiple extensions and priorities are used for selecting an intended one. In this section, we introduce priorities to default reasoning in PLP.

Example 3.6 Let $K_2 = (P_2, \Delta_2)$ be the knowledge system such that

$$
\begin{aligned}
P_2 : \; & bird(x) \leftarrow penguin(x), \\
& bird(polly) \leftarrow, \;\; penguin(tweety) \leftarrow . \\
\Delta_2 : \; & flies(x) \leftarrow bird(x), \\
& \neg flies(x) \leftarrow penguin(x).
\end{aligned}
$$

Compared with Example 3.4, the first rule in P_1 is placed at Δ_2 as a default rule. Then K_2 has another extension base $S' = \{\, bird(polly),\, penguin(tweety),\, bird(tweety),\, flies(polly),\, flies(tweety) \,\}$, in addition to $S = \{\, bird(polly),\, penguin(tweety),\, bird(tweety),\, flies(polly),\, \neg flies(tweety) \,\}$.

In Example 3.6 we want to prefer S to S' as in Example 3.4, because S is produced by the default rule $\neg flies(x) \leftarrow penguin(x)$ which presents an exception of the rule $flies(x) \leftarrow bird(x)$. To select the intended extension base, we need a mechanism of specifying priorities between defaults.

To this end, we combine the technique of prioritization over rules presented in Section 2.3 with the PLP (Π, Φ_{KS}) in Section 3.2.1. For each default rule $Head \leftarrow Body$ in Δ, its *named rule* is defined as $r(\mathbf{x}) = (Head \leftarrow Body)$ where $r(\mathbf{x})$ is an atom representing the *(default) name*, and $\mathbf{x}$ represents variables appearing in the rule. A default rule $Head \leftarrow Body$ is identified with its

default name. The set of ground instances of default names of all defaults in Δ is denoted by $N(\Delta)$.

Definition 3.7 (generating default) Let $K = (P, \Delta)$ be a knowledge system. A ground default r from Δ is called *generating* in an extension base S if S satisfies the body of r. The set of all default names such that the corresponding defaults from Δ are generating in S is denoted by $GD(S)$.

We introduce the priority relation $\preceq$ over default names.

Definition 3.8 (priorities between default rules) For any default names r_i and r_j from $N(\Delta)$, if $r_j \preceq r_i$ we say that a default rule r_i has a *higher priority than* a default rule r_j.

Intuitively, $r_j \preceq r_i$ means that the default r_i has the precedence over the default r_j in the generation of an extension base. Using the priority, we select an extension base which is generated by default rules with relatively higher priorities.

Definition 3.9 (preferred extension base) Let $K = (P, \Delta)$ be a knowledge system and Φ_D a set of priorities over default names. For any extension bases S and T of K, $GD(T) \sqsubseteq GD(S)$ is defined as in Definition 2.3 wrt the priorities in Φ_D. An extension base S is called a *preferred extension base* if $GD(S) \sqsubseteq GD(T)$ implies $GD(T) \sqsubseteq GD(S)$ (wrt Φ_D) for any extension base T of K.

Definition 3.10 (prioritized knowledge system) Let (Π, Φ_{KS}) be a PLP representing a knowledge system $K = (P, \Delta)$. Given a set Φ_D of priorities over default names, a *prioritized knowledge system* is defined as a PLP (Π_R, Φ_{PKS}) such that

$$\Pi_R = \Pi \cup R \text{ where } R = \{\, r \leftarrow \delta,\, Body \mid (Head \leftarrow \delta,\, Body) \in \Pi \text{ and } r = (Head \leftarrow Body) \in \Delta \,\},$$
$$\Phi_{PKS} = \Phi_{KS} \cup \Phi_D\,.$$

In the definition, R introduces rules which imply default names (cf. Section 2.3) and Φ_D introduces priorities over defaults. We show that the PLP (Π_R, Φ_{PKS}) realizes prioritized default reasoning.

Lemma 3.6 *(prioritized knowledge system vs. knowledge system) Let (Π_R, Φ_{PKS}) be a prioritized knowledge system. If S is a p-answer set of (Π_R, Φ_{PKS}), $S \setminus N(\Delta)$ is a p-answer set of (Π, Φ_{KS}).*

Proof. Priorities in Φ_D within Φ_{PKS} do not relate to any priority in Φ_{KS}, and the priorities in Φ_D filter the p-answer sets of (Π, Φ_{KS}) using default names

derived by the rules in R. Thus, if S is a p-answer set of (Π_R, Φ_{PKS}), removing default names from S makes $S \setminus N(\Delta)$ a p-answer set of (Π, Φ_{KS}). □

Theorem 3.7 *(preferred extension base vs. prioritized knowledge system) Let $K = (P, \Delta)$ be a knowledge system, Φ_D a set of priorities over default names, and (Π_R, Φ_{PKS}) a prioritized knowledge system. If S is a p-answer set of (Π_R, Φ_{PKS}), there is a preferred extension base S' of K (wrt Φ_D) such that $S' = S \cap \mathcal{L}_K$. Conversely, if S' is a preferred extension base of K (wrt Φ_D), there is a p-answer set S of (Π_R, Φ_{PKS}) such that $S \cap \mathcal{L}_K = S'$.*

Proof. Let S be a p-answer set of (Π_R, Φ_{PKS}). As $S \setminus N(\Delta)$ is a p-answer set of (Π, Φ_{KS}) (Lemma 3.6), $S' = S \cap \mathcal{L}_K$ is an extension base of $K = (P, \Delta)$ (Theorem 3.5). Let T be any answer set of Π_R such that $T \setminus N(\Delta)$ is a p-answer set of (Π, Φ_{KS}). Then, $T' = T \cap \mathcal{L}_K$ is also an extension base of $K = (P, \Delta)$. As S is a p-answer set of (Π_R, Φ_{PKS}), $S \sqsubseteq T$ implies $T \sqsubseteq S$ wrt Φ_{PKS}. Since $GD(S') \subseteq S$ and $GD(T') \subseteq T$, $GD(S') \sqsubseteq GD(T')$ implies $GD(T') \sqsubseteq GD(S')$ wrt Φ_D. Hence, S' is a preferred extension base of K. The converse is shown in a similar manner. □

Example 3.7 The knowledge system K_2 of Example 3.6 is expressed in the PLP (Π_R, Φ_{PKS}) as

$$
\begin{aligned}
\Pi_R : \; & bird(x) \leftarrow penguin(x), \\
& bird(polly) \leftarrow, \;\; penguin(tweety) \leftarrow, \\
& flies(x) \leftarrow \delta_1(x),\, bird(x), \\
& \neg flies(x) \leftarrow \delta_2(x),\, penguin(x), \\
& r_1(x) \leftarrow \delta_1(x),\, bird(x), \\
& r_2(x) \leftarrow \delta_2(x),\, penguin(x), \\
& \delta_1(x) \mid not\, \delta_1(x) \leftarrow, \\
& \delta_2(x) \mid not\, \delta_2(x) \leftarrow . \\
\Phi_{PKS} : \; & not\, \delta_1(x) \preceq \delta_1(x)\,, \;\; not\, \delta_2(x) \preceq \delta_2(x)\,, \;\; r_1(x) \preceq r_2(x)\,.
\end{aligned}
$$

Then, (Π_R, Φ_{PKS}) has the unique p-answer set

$$
\begin{aligned}
\{\, & bird(polly),\, penguin(tweety),\, bird(tweety),\, flies(polly),\, \neg flies(tweety), \\
& \delta_1(polly),\, \delta_2(polly),\, \delta_2(tweety),\, r_1(polly),\, r_2(tweety)\,\},
\end{aligned}
$$

which corresponds to the intended extension base.

3.3.1 Parallel circumscription

In this section we consider realizing *circumscription* in PLP. We first review the framework of circumscription from [39].

Given a first-order theory T, let P and Z be disjoint tuples of predicates from T. Then *(parallel) circumscription* of P in T with *variable* Z is defined as the second-order formula

$$Circ(T; P; Z) = T(P, Z) \wedge \neg \exists P'\, Z'(T(P', Z') \wedge P' < P)$$

where $T(P, Z)$ is a theory containing predicate constants P, Z, and P', Z' are tuples of predicate variables that have the same arities as those predicates in P, Z. The set of all predicates other than P, Z from T is denoted by Q. The predicates in Q are called the *fixed* predicates.

For a structure M, let $|M|$ be its universe and $M[\![C]\!]$ the interpretations of all individual, function, and predicate constants C in the language. For any two structures M_1 and M_2, $M_1 \ll M_2$ iff

(i) $|M_1| = |M_2|$,
(ii) $M_1[\![Q]\!] = M_2[\![Q]\!]$,
(iii) $M_1[\![P]\!] \subset M_2[\![P]\!]$.

A model M of T is a model of $Circ(T; P; Z)$ iff there is no model N of T such that $N \ll M$.

To realize circumscription in the context of logic programming, we assume a first-order theory T as a set of *clauses* of the form:

$$A_1 \vee \cdots \vee A_l \vee \neg B_1 \vee \cdots \vee \neg B_m \tag{5}$$

where each A_i $(1 \leq i \leq l; l \geq 0)$ and B_j $(1 \leq j \leq m; m \geq 0)$ are atoms. Also, we consider the *Herbrand model* of T, which has the effect of introducing both the *domain closure assumption* and the *unique name assumption* into T [5,40]. Now the PLP expression of circumscription is defined as follows.

Definition 3.11 (circumscription in PLP) Given a circumscription $Circ(T; P; Z)$, its PLP expression (Π, Φ_{CIRC}) is defined as follows.

(i) For any clause (5) in T, Π has the rule

$$A_1 \mid \cdots \mid A_l \leftarrow B_1, \ldots, B_m.$$

(ii) For any fixed or variable predicate λ in T, Π has the rule

$$\lambda(\mathbf{x}) \mid not\, \lambda(\mathbf{x}) \leftarrow .$$

(iii) Priorities are given as

$$\begin{aligned}\Phi_{CIRC}=&\{\, p_i(\mathbf{x}) \preceq not\, p_i(\mathbf{x}) \;:\; p_i \in P_i \;(i=1,\ldots,k)\,\}\\ &\cup\{\, q(\mathbf{x}) \preceq not\, q(\mathbf{x}),\;\; not\, q(\mathbf{x}) \preceq q(\mathbf{x}) \;:\; q \in Q\,\}.\end{aligned}$$

Here, $\mathbf{x}$ is a tuple of variables in each predicate.

In the transformation, minimizing extensions of predicates from P is expressed by the priority $p_i(\mathbf{x}) \preceq not\, p_i(\mathbf{x})$ in Φ_{CIRC}. On the other hand, each atom with a fixed or variable predicate is either true or not, and it is expressed by the second disjunctive rule. In this case, extensions of variable predicates can be varied, while those of fixed predicates are not affected by priorities over minimized predicates. This situation is expressed by the symmetric priorities $q(\mathbf{x}) \preceq not\, q(\mathbf{x})$ and $not\, q(\mathbf{x}) \preceq q(\mathbf{x})$ in Φ_{CIRC}.

With this setting, circumscription is expressed in terms of PLP. In the following, p is also used to represent an atom with a minimized predicate from P, and q an atom with a fixed predicate. Also, P, Z, Q are used to represent the sets of atoms with the corresponding predicates.

Theorem 3.8 *(circumscription vs. p-answer set) Let $Circ(T;P;Z)$ be a circumscription and (Π, Φ_{CIRC}) its PLP expression. Then, M is an Herbrand model of $Circ(T;P;Z)$ iff M is a p-answer set of (Π, Φ_{CIRC}).*

Proof. M is a model of $Circ(T;P;Z)$ iff there is no model N of T such that $N \ll M$. For any two models M and N such that $M \cap Q = N \cap Q$, $N \ll M$
iff $\exists p \in P\; (p \in M \setminus N) \wedge \neg\exists p' \in P\; (p' \in N \setminus M)$
iff $\exists p \in P\; (not\, p \in N \setminus M \wedge p \in M \setminus N) \wedge \neg\exists p' \in P\; (not\, p' \in M \setminus N \wedge p' \in N \setminus M)$
iff $M \sqsubseteq N$ and $N \not\sqsubseteq M$ (wrt Φ_{CIRC}).
Hence, for any M and N such that $M \cap Q = N \cap Q$, $N \ll M$ iff $M \sqsubseteq N$ and $N \not\sqsubseteq M$. Therefore, $N \not\ll M$ iff ($M \sqsubseteq N$ implies $N \sqsubseteq M$).
On the other hand, for any M and N such that $M \cap Q \neq N \cap Q$, if $q \in (M \setminus N) \cap Q$ then $M \sqsubseteq N$ by $q \preceq not\, q$ in Φ_{CIRC}. In this case, $N \sqsubseteq M$ also holds by $not\, q \preceq q$. Thus, $M \sqsubseteq N$ iff $N \sqsubseteq M$. As $M \cap Q \neq N \cap Q$, $N \not\ll M$ and $M \not\ll N$ hold.
Therefore, for any M and N, $N \not\ll M$ iff ($M \sqsubseteq N$ implies $N \sqsubseteq M$) $(*)$.
Let $M \cap (Q \cup Z) = \Gamma$. If M is an Herbrand model of $Circ(T;P;Z)$, then M is a minimal model of $T \cup \Gamma$. In this case, M is a minimal model of
$T \cup \{\, (\lambda \mid not\, \lambda \leftarrow) \in \Pi\,\}^M$ iff M is a minimal model of Π^M
iff M is an answer set of Π.
Conversely, if M is an answer set of Π, M is an Herbrand model of T. Thus, the statement $(*)$ holds for answer sets M and N of Π. Hence, M is an Herbrand

model of $Circ(T;P;Z)$ iff M is a p-answer set of (Π, Φ_{CIRC}). □

Example 3.8 [39] Let T be the first-order theory [8]

$$
\begin{array}{l}
block(x) \wedge \neg ab(x) \supset ontable(x), \\
\neg ontable(b_1), \\
block(b_1), \ \ block(b_2),
\end{array}
$$

where $P = \{ ab \}$, $Z = \{ ontable \}$ and $Q = \{ block \}$. $Circ(T;P;Z)$ is expressed in the PLP (Π, Φ_{CIRC}) as

$$
\begin{array}{rl}
\Pi : & ontable(x) \mid ab(x) \leftarrow block(x), \\
& \leftarrow ontable(b_1), \\
& block(b_1) \leftarrow, \ \ block(b_2) \leftarrow, \\
& ontable(x) \mid not\ ontable(x) \leftarrow, \\
& block(x) \mid not\ block(x) \leftarrow . \\
\Phi_{CIRC} : & ab(x) \preceq not\ ab(x), \\
& block(x) \preceq not\ block(x), \ \ not\ block(x) \preceq block(x).
\end{array}
$$

Then, (Π, Φ_{CIRC}) has the p-answer set

$$\{ block(b_1),\ block(b_2),\ ab(b_1),\ ontable(b_2) \},$$

which correspond to the Herbrand model of $Circ(T;P;Z)$.

3.3.2 Prioritized circumscription

Next we consider realizing *prioritized circumscription* [35] in PLP.

Let P be a tuple of predicates from a first-order theory T, which is split into disjoint parts $P_1, \ldots, P_k$. Then *prioritized circumscription* $Circ(T; P_1 > \cdots > P_k; Z)$ minimizes extensions of P_i with a priority higher than those of P_j $(i < j)$ with Z varied. The set Q of all predicates other than P and Z from T are fixed as before. For any two structures M_1 and M_2, $M_1 \ll M_2$ iff

(i) $|M_1| = |M_2|$,
(ii) $M_1[\![Q]\!] = M_2[\![Q]\!]$,
(iii) for every $j = 1, \ldots, k$, if $M_1[\![P_1, \ldots, P_{j-1}]\!] = M_2[\![P_1, \ldots, P_{j-1}]\!]$ then $M_1[\![P_j]\!] \subset M_2[\![P_j]\!]$,

[8] The unique name assumption holds under the Herbrand interpretation, hence $b_1 \neq b_2$ is omitted in T.

where $M[\![P_1, \ldots, P_k]\!] = M[\![P_1 \cup \cdots \cup P_k]\!]$. A model M of T is a model of $Circ(T; P_1 > \cdots > P_k; Z)$ iff there is no model N of T such that $N \ll M$ [36].

Given a set T of clauses, the PLP expression of prioritized circumscription is defined as follows.

Definition 3.12 (prioritized circumscription in PLP) Given a prioritized circumscription $Circ(T; P_1 > \cdots > P_k; Z)$, its PLP expression (Π, Φ_{PCIRC}) is defined as follows.

(i) For any clause (5) in T, Π has the rule

$$A_1 \mid \cdots \mid A_l \leftarrow B_1, \ldots, B_m.$$

(ii) For any fixed or variable predicate λ in T, Π has the rule

$$\lambda(\mathbf{x}) \mid not\, \lambda(\mathbf{x}) \leftarrow .$$

(iii) Priorities are given as

$$\begin{aligned}\Phi_{PCIRC} = &\{\, p_i(\mathbf{x}) \preceq not\, p_i(\mathbf{x}) \,:\, p_i \in P_i \ (i = 1, \ldots, k) \,\} \\ &\cup \{\, not\, p_{i+1}(\mathbf{x}) \preceq not\, p_i(\mathbf{y}) \,:\, p_i \in P_i,\ p_{i+1} \in P_{i+1} \ (i = 1, \ldots, k-1) \,\} \\ &\cup \{\, q(\mathbf{x}) \preceq not\, q(\mathbf{x}),\ \ not\, q(\mathbf{x}) \preceq q(\mathbf{x}) \,:\, q \in Q \,\}.\end{aligned}$$

Here, $\mathbf{x}$ and $\mathbf{y}$ are tuples of variables in each predicate.

The transformation is the same as the case of parallel circumscription with the only difference that the predicate hierarchy $P_1 > \cdots > P_k$ is expressed in Φ_{PCIRC} as $not\, p_{i+1}(\mathbf{x}) \preceq not\, p_i(\mathbf{y})$, which means that extensions from p_i is minimized at a higher priority than those from p_{i+1}.

With this setting, prioritized circumscription is characterized by the p-answer sets of (Π, Φ_{PCIRC}). In the following, p_i is also used to represent an atom with a minimized predicate from P_i.

Theorem 3.9 *(prioritized circumscription vs. p-answer set) Let* $Circ(T; P_1 > \cdots > P_k; Z)$ *be a prioritized circumscription and* (Π, Φ_{PCIRC}) *its PLP expression. Then,* M *is an Herbrand model of* $Circ(T; P_1 > \cdots > P_k; Z)$ *iff* M *is a p-answer set of* (Π, Φ_{PCIRC}).

Proof. First, any model M of $Circ(T; P_1 > \cdots > P_k; Z)$ is a model of $Circ(T; P_1, \ldots, P_k; Z)$. Then, M is an Herbrand model of $Circ(T; P_1 > \cdots > P_k; Z)$ iff there is no Herbrand model N of $Circ(T; P_1, \ldots, P_k; Z)$ such that $N \ll M$. For any Herbrand model M and N of $Circ(T; P_1, \ldots, P_k; Z)$ such that $M \cap Q = N \cap Q$, $N \ll M$ iff

$$\begin{aligned}&\exists i\ (1 \le i \le k)\ \ \exists p_i \in P_i\, (p_i \in M \setminus N) \ \wedge\ \neg \exists p_i' \in P_i\, (p_i' \in N \setminus M) \\ &\qquad \wedge\, \forall p_j \in P_j\, (j < i)\, (p_j \in M \Leftrightarrow p_j \in N). \quad (*)\end{aligned}$$

Since M is minimal wrt the extensions of P, $(*)$ implies

$$\exists k\,(i<k)\,\exists p_k \in P_k\,(p_k \in N \setminus M).$$

Hence, $(*)$ iff

$$\begin{aligned}
&\exists i\,(1 \le i \le k)\ \ \exists p_i \in P_i\,(p_i \in M \setminus N)\\
&\qquad \wedge \exists k\,(i<k)\,\exists p_k \in P_k\,(p_k \in N \setminus M)\\
&\qquad \wedge \neg\exists p'_i \in P_i\,(p'_i \in N \setminus M)\\
&\qquad \wedge \forall p_j \in P_j\,(j<i)\,(p_j \in M \Leftrightarrow p_j \in N)\\
\text{iff}\quad &\exists i\,(1 \le i \le k)\ \ \exists p_i \in P_i\,(not\ p_i \in N \setminus M)\\
&\qquad \wedge \exists k\,(i<k)\,\exists p_k \in P_k\,(not\ p_k \in M \setminus N)\\
&\qquad \wedge \neg\exists p'_i \in P_i\,(not\ p'_i \in M \setminus N)\\
&\qquad \wedge \neg\exists p_j \in P_j\,(j<i)\,(not\ p_j \in M \setminus N)\\
&\qquad \wedge \neg\exists p'_j \in P_j\,(j<i)\,(not\ p'_j \in N \setminus M)\\
\text{iff}\quad &\exists i\,(1 \le i \le k)\ \ \exists p_i \in P_i\,(not\ p_i \in N \setminus M)\\
&\qquad \wedge \exists k\,(i<k)\,\exists p_k \in P_k\,(not\ p_k \in M \setminus N)\\
&\qquad \wedge \neg\exists p_j \in P_j\,(j \le i)\,(not\ p_j \in M \setminus N)\\
&\qquad \wedge \neg\exists p'_j \in P_j\,(j<i)\,(not\ p'_j \in N \setminus M).\ (\dagger)
\end{aligned}$$

Here,

$$\begin{aligned}
&\exists i\,(1 \le i \le k)\ \ \exists p_i \in P_i\,(not\ p_i \in N \setminus M)\\
&\qquad \wedge \exists k\,(i<k)\,\exists p_k \in P_k\,(not\ p_k \in M \setminus N)\\
&\qquad \wedge \neg\exists p_j \in P_j\,(j \le i)\,(not\ p_j \in M \setminus N)\quad (\ddagger)
\end{aligned}$$

implies $M \sqsubseteq N$ and $N \not\sqsubseteq M$ (wrt Φ_{PCIRC}). Therefore, $(\dagger)$ implies $M \sqsubseteq N$ and $N \not\sqsubseteq M$. Conversely, $M \sqsubseteq N$ and $N \not\sqsubseteq M$ imply $(\ddagger)$. In this case, there is a minimal i which satisfies $(\ddagger)$. Consider the minimal i' which satisfies the first conjunct $\exists p_{i'} \in P_{i'}\,(not\ p_{i'} \in N \setminus M)$ of $(\ddagger)$. Then, the second conjunct $\exists k\,(i'<k)\,\exists p_k \in P_k\,(not\ p_k \in M \setminus N)$ is also satisfied. If the third conjunct is not satisfied, i.e., $\exists p_j \in P_j\,(j \le i')\,(not\ p_j \in M \setminus N)$, then $M \sqsubseteq N$ implies $N \sqsubseteq M$, which contradicts the assumption. Hence, $\neg\exists p_j \in P_j\,(j \le i')\,(not\ p_j \in M \setminus N)$ also holds. Since i' is a minimal one satisfying $\exists p_{i'} \in P_{i'}\,(not\ p_{i'} \in N \setminus M)$, it holds that $\neg\exists p'_j \in P_j\,(j<i')\,(not\ p'_j \in N \setminus M)$. Then, by putting $i = i'$, $(\ddagger)$ implies $(\dagger)$, thus $M \sqsubseteq N$ and $N \not\sqsubseteq M$ imply $(\dagger)$. Hence, for any M and N such that $M \cap Q = N \cap Q$, $N \ll M$ iff $M \sqsubseteq N$ and $N \not\sqsubseteq M$, thereby $N \not\ll M$ iff $(M \sqsubseteq N$ implies $N \sqsubseteq M)$.

On the other hand, for any M and N such that $M \cap Q \neq N \cap Q$, $M \sqsubseteq N$ iff $N \sqsubseteq M$ by the same argument as in Theorem 3.8. Therefore, for any M and N, $N \not\ll M$ iff $(M \sqsubseteq N$ implies $N \sqsubseteq M)$. Since Herbrand models M and N of $Circ(T; P_1, \ldots, P_k; Z)$ are (p-)answer sets of Π by Theorem 3.8, M is an Herbrand model of $Circ(T; P_1 > \cdots > P_k; Z)$ iff M is a p-answer set of (Π, Φ_{PCIRC}). □

Example 3.9 [39] Let T be the first-order theory

$$\begin{aligned}
&block(x) \wedge \neg ab_1(x) \supset ontable(x),\\
&heavy_block(x) \wedge \neg ab_2(x) \supset \neg ontable(x),\\
&heavy_block(x) \supset block(x),
\end{aligned}$$

$$heavy\_block(b_1),\ \ block(b_2),\ \ \neg heavy\_block(b_2),$$

where $P_1 = \{\, ab_2 \,\}$ and $P_2 = \{\, ab_1 \,\}$ with $P_1 > P_2$, and $Z = \{\, ontable \,\}$ and $Q = \{\, block,\ heavy\_block \,\}$. $Circ(T; P_1 > P_2; Z)$ is expressed in the PLP (Π, Φ_{PCIRC}) as

$$\begin{aligned}
\Pi : \ & ontable(x) \mid ab_1(x) \leftarrow block(x), \\
& ab_2(x) \leftarrow ontable(x),\ heavy_block(x), \\
& block(x) \leftarrow heavy_block(x), \\
& heavy_block(b_1) \leftarrow,\ \ block(b_2) \leftarrow, \\
& \leftarrow heavy_block(b_2), \\
& ontable(x) \mid not\ ontable(x) \leftarrow, \\
& block(x) \mid not\ block(x) \leftarrow, \\
& heavy_block(x) \mid not\ heavy_block(x) \leftarrow . \\
\Phi_{PCIRC} : \ & ab_1(x) \preceq not\ ab_1(x),\ \ ab_2(x) \preceq not\ ab_2(x), \\
& not\ ab_1(x) \preceq not\ ab_2(x), \\
& block(x) \preceq not\ block(x),\ \ not\ block(x) \preceq block(x), \\
& heavy_block(x) \preceq not\ heavy_block(x), \\
& not\ heavy_block(x) \preceq heavy_block(x).
\end{aligned}$$

Then, (Π, Φ_{PCIRC}) has the p-answer set

$$\{\, heavy\_block(b_1),\ block(b_1),\ block(b_2),\ ontable(b_2),\ ab_1(b_1) \,\},$$

which corresponds to the Herbrand model of $Circ(T; P_1 > P_2; Z)$.

3.3.3 Connection to the perfect model semantics

It is known that prioritized circumscription is also characterized by the *perfect model semantics* [46] of a stratified disjunctive program in the absence of fixed and variable predicates. In this section, we address the semantical relationship between perfect models and p-answer sets.

As presented in Section 2.1, normal disjunctive programs are defined as a subset of GEDPs. An NDP consists of rules of the form

$$A_1 \mid \cdots \mid A_l \leftarrow A_{l+1}, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n \ \ (n \geq m \geq l \geq 0) \quad (6)$$

where each A_i is an atom. An NDP is called a *positive disjunctive program* if each rule contains no NAF (i.e., $m = n$). An NDP Π is *stratified* [46] if it is possible to decompose the set P of all predicates of Π into the disjoint sets

$P_1, \ldots, P_k$ (called *strata*), such that for every rule (6) in Π,

(i) predicates of the atoms A_h $(h = 1, \ldots, l)$ belong to the same stratum P_s;
(ii) predicates of the atoms A_i $(i = l+1, \ldots, m)$ belong to $\bigcup\{P_t : t \leq s\}$;
(iii) predicates of the atoms A_j $(j = m+1, \ldots, n)$ belong to $\bigcup\{P_t : t < s\}$.

Any decomposition $\{P_1, \ldots, P_k\}$ satisfying the above conditions is called a *stratification* of Π.

Let $pred(A)$ be the predicate of an atom A. An atom A has a *higher priority than* an atom B (written $B < A$) iff $pred(A) \in P_i$ and $pred(B) \in P_j$ with $i < j$. Given two distinct models M and N, M is *preferable* to N $(M \ll N)$ iff for any atom $A \in M \setminus N$ there is an atom $B \in N \setminus M$ such that $A < B$. A model M is *perfect* if there is no model preferable to M.

In a stratified program the existence of integrity constraints causes some problems. Syntactically, an integrity constraint $\leftarrow p$ has the same effect as the non-stratified rule $q \leftarrow p, \mathit{not}\, q$ where q is a new atom appearing nowhere in a program. Semantically, a perfect model may not be *supported* [2,4] [9] in the presence of integrity constraints.

Example 3.10 Let $\Pi = \{\, q \leftarrow \mathit{not}\, p, \;\; \leftarrow \;\; q \,\}$ with the priority $q < p$. Then Π has the perfect model $\{p\}$ which is not supported.

Note that the above program has no answer set. Thus, perfect models provide an intuitive meaning when a stratified program contains no integrity constraints. With this reason, we assume no integrity constraints in stratified programs hereafter in this subsection.

In a stratified program Π, the perfect models coincide with the answer sets [47], hence they also coincide with the p-answer sets of $(\Pi, \emptyset)$. In what follows, we present yet another characterization of perfect models of a stratified NDP in terms of p-answer sets of a PLP.

Given an NDP Π, we define the corresponding first-order theory $T(\Pi)$ such that any rule (6) in Π is transformed to the clause

$$A_1 \vee \cdots \vee A_l \vee \neg A_{l+1} \vee \cdots \vee \neg A_m \vee A_{m+1} \vee \cdots \vee A_n \qquad (7)$$

in $T(\Pi)$. We write $Circ(T; P_1 > \cdots > P_k; Z)$ with $Z = \emptyset$ simply as $Circ(T; P_1 > \cdots > P_k)$.

[9] A model M of an NDP P is *supported* [4] if for any atom $A \in M$ there is a ground rule of the form (6) from P s.t. $\{A_1, \ldots, A_l\} \cap M = A$, $\{A_{l+1}, \ldots, A_m\} \subseteq M$, and $\{A_{m+1}, \ldots, A_n\} \cap M = \emptyset$.

Lemma 3.10 *[46, Theorem 5][10] (perfect model vs. prioritized circumscription) Let* Π *be a stratified NDP and* $\{P_1, \ldots, P_k\}$ *a stratification of* Π*. Then,* M *is a perfect model of* Π *iff* M *is an Herbrand model of* $Circ(T(\Pi); P_1 > \cdots > P_k)$*.*

Let $T^+(\Pi)$ be a positive disjunctive program such that any clause (7) in $T(\Pi)$ is replaced by the rule

$$A_1 \mid \cdots \mid A_l \mid A_{m+1} \mid \cdots \mid A_n \leftarrow A_{l+1}, \ldots, A_m$$

in $T^+(\Pi)$.

Theorem 3.11 *(perfect model vs. p-answer set) Let* Π *be a stratified NDP with the stratification* $\{P_1, \ldots, P_k\}$*. Then,* M *is a perfect model of* Π *iff* M *is a p-answer set of* $(T^+(\Pi), \Phi_{STRAT})$ *where* $\Phi_{STRAT} = \{\, not\, p_{i+1}(\mathbf{x}) \preceq not\, p_i(\mathbf{y}) : p_i \in P_i\,,\, p_{i+1} \in P_{i+1}\ (i = 1, \ldots, k-1)\}$*.*

Proof. When there are no fixed and variable predicates, the PLP expression of prioritized circumscription of Definition 3.12 includes neither disjunctive rules of (ii) nor symmetric priorities on predicates from Q in Φ_{PCIRC}. Moreover, any p-answer set of $T^+(\Pi)$ is minimal wrt extensions of the predicates from P due to the minimality of answer sets (or minimal models) of a positive disjunctive program. Thus, priorities $p_i(\mathbf{x}) \preceq not\, p_i(\mathbf{x})$ $(i = 1, \ldots, k)$ in Φ_{PCIRC} are automatically satisfied. Then the result follows by Theorem 3.9 and Lemma 3.10. □

Example 3.11 Let Π be the program

$$p \mid q \leftarrow not\, r,$$
$$r \leftarrow not\, s$$

with the stratification $P_1 = \{s\}$, $P_2 = \{r\}$, $P_3 = \{p, q\}$. It is expressed by the PLP $(T^+(\Pi), \Phi_{STRAT})$ as

$$T^+(\Pi) : p \mid q \mid r \leftarrow,$$
$$r \mid s \leftarrow .$$
$$\Phi_{STRAT} : not\, p \preceq not\, r,\ \ not\, q \preceq not\, r,\ \ not\, r \preceq not\, s\,.$$

Then, $(T^+(\Pi), \Phi_{STRAT})$ has the p-answer set $\{r\}$, which coincides with the perfect model of Π.

The above theorem presents that a stratified NDP is equivalently expressed

[10] The expression is modified in our context.

by a *not*-free positive disjunctive program plus priorities.[11] The result is also directly extended to *locally stratified* NDPs.

4 Computation

4.1 ϕ-program

In this section, we provide an algorithm for selecting p-answer sets from answer sets. For this purpose, we introduce a program transformation which embeds priorities into a program. To make such embedding easier, we first eliminate NAF formulas in priorities without changing the meaning of a PLP.

Definition 4.1 (eliminating NAF from Φ) Given a PLP (P, Φ), define (P', Φ') which is obtained by replacing any NAF formula $not\, a$ in Φ with $\overline{a}$ in Φ', and introducing a new rule $\overline{a} \leftarrow not\, a$ to P for any such replacement. The resulting program is P'.

Example 4.1 Let (P, Φ) be the PLP such that

$$
\begin{array}{l}
P : p \leftarrow q, \\
\quad\; q \mid not\ q \leftarrow . \\
\Phi : q \preceq not\, q.
\end{array}
$$

Then, (P', Φ') becomes

$$
\begin{array}{l}
P' : p \leftarrow q, \\
\quad\; q \mid not\ q \leftarrow, \\
\quad\; \overline{q} \leftarrow not\, q. \\
\Phi' : q \preceq \overline{q}.
\end{array}
$$

(P', Φ') has the p-answer set $\{\overline{q}\}$ which corresponds to the p-answer set $\emptyset$ of (P, Φ).

Proposition 4.1 *(PLP with NAF-free Φ) Given a PLP (P, Φ), let (P', Φ') be a PLP which is obtained by Definition 4.1. If S is a p-answer set of (P, Φ), there is a p-answer set S' of (P', Φ') s.t. $S' \cap \mathcal{L}_P = S$. In converse, if S' is a p-answer set of (P', Φ'), there is a p-answer set S of (P, Φ) s.t. $S = S' \cap \mathcal{L}_P$.*

Proof. By the definition, $a \notin S$ iff $\overline{a} \in S'$ for any $a \in \mathcal{L}_P$. Then the result holds. □

[11] [13] presents a different method of replacing NAF with priorities over rules.

Thus, without loss of generality, in this section we consider PLPs which contain no NAF formulas in Φ.

Next we consider representing priorities in terms of rules, which is used for computing p-answer sets.

Definition 4.2 (ϕ-program) Given a PLP (P, Φ), the *ϕ-program* is defined as

$$P_\Phi = P \cup \{ \phi^+_{c_i \prec c_j} \leftarrow c_j, not\, c_i\,, \;\; \phi^-_{c_i \prec c_j} \leftarrow c_i, not\, c_j \mid (c_i \prec c_j) \in \Phi^* \}.$$

The newly introduced rules are called *ϕ-rules*, and the atoms $\phi^+_{c_i \prec c_j}$ and $\phi^-_{c_i \prec c_j}$ are called *ϕ-atoms*. The set of ϕ-rules is finite when the closure Φ^* is finite (modulo variable renaming). The idea of ϕ-rules is as follows. If an answer set contains c_j but does not contain c_i, the atom $\phi^+_{c_i \prec c_j}$ becomes true by the ϕ-rule; else if the converse is the case, the atom $\phi^-_{c_i \prec c_j}$ becomes true. Thus, if an answer set implies ϕ-atoms, it indicates that the answer set contains a literal which is subject to preference. In P_Φ, the "strict" priority relation $\prec$ is considered instead of $\preceq$. If $c_i \preceq c_j$ and $c_j \preceq c_i$ hold, two answer sets respectively containing c_i and c_j have an equal priority with respect to these literals. Using the ϕ-program, the following procedure selects p-answer sets from answer sets.

Definition 4.3 (procedure for selecting p-answer sets)
Let (P, Φ) be a PLP such that the closure Φ^* is finite. Then, the following procedure outputs a set Δ of answer sets.

(i) Put Σ and Δ as the sets of all answer sets of P_Φ.
(ii) For every $T \in \Sigma$, check the following: for any $(c_i \prec c_j) \in \Phi^*$, if $\phi^-_{c_i \prec c_j} \in T$ and $\phi^+_{c_i \prec c_j} \in T'$ for some $T' \in \Sigma$, and there is no $(c_j \prec c_k) \in \Phi^*$ s.t. $\phi^+_{c_j \prec c_k} \in T$ and $\phi^-_{c_j \prec c_k} \in T'$, then discard T from Δ.

In the first step, we assume an external procedure for computing the answer sets of an GEDP P_Φ. A procedure for this purpose is given in [26] for function-free and range-restricted GEDPs. In the second step, any answer set which includes a literal with a relatively lower priority is discarded from Δ using priority information encoded in ϕ-atoms.

Note that if we check preference between answer sets without using ϕ-atoms, we have to check priority relations over all literals included in every answer set of P. By contrast, ϕ-atoms appear in an answer set of P_Φ only if the answer set contains any literal which is subject to priorities. Thus, to check preference between answer sets it is enough to compare answer sets containing ϕ-atoms and literals appearing in ϕ-atoms. Any answer set including no ϕ-atom is irrelevant to preference, and it becomes a p-answer set automatically.

We show that the above procedure is used for selecting the p-answer sets of a PLP.

Definition 4.4 (cycle-free) The p-answer sets of a PLP (P, Φ) are called *cycle-free* if $S_1 \sqsubseteq S_2$ implies $S_1 \sqsubset S_2$ for any two p-answer sets S_1 and S_2 of (P, Φ).

Theorem 4.2 *(soundness/completeness of the procedure) Let (P, Φ) be a PLP with finite Φ^*, and Δ the set produced by the above procedure. If $T \in \Delta$, there is a p-answer set S of (P, Φ) s.t. $S = T \cap \mathcal{L}_P$. The converse also holds if the p-answer sets of (P, Φ) are cycle-free.*

Proof. When T is an answer set of P_Φ, $T \cap \mathcal{L}_P$ is an answer set of P. Thus, for any $T \in \Delta$, $T \cap \mathcal{L}_P$ is an answer set of P. If there is no $\phi^-_{c_i \prec c_j}$ in T, T contains no literal c_i such that $(c_i \prec c_j) \in \Phi^*$, so S is a p-answer set of (P, Φ). Else if there is some $\phi^-_{c_i \prec c_j}$ in T, it implies either (a) $\neg\exists T' \in \Sigma$ s.t. $\phi^+_{c_i \prec c_j} \in T'$, or (b) $\exists T' \in \Sigma$ s.t. $\phi^+_{c_i \prec c_j} \in T'$, and $\phi^+_{c_j \prec c_k} \in T$ and $\phi^-_{c_j \prec c_k} \in T'$ for some $(c_j \prec c_k) \in \Phi^*$. In case of (a), there is no T' s.t. $T \sqsubseteq T'$. In case of (b), $c_k \in T \setminus T'$ for some $(c_j \prec c_k) \in \Phi^*$. Then, $T \not\sqsubseteq T'$ by the definition. Thus, in either case, there is no answer set $S' = T' \cap \mathcal{L}_P$ of P, which is preferable to $S = T \cap \mathcal{L}_P$. Hence, S is a p-answer set of (P, Φ).

The converse direction proceeds as follows. Since $\{\, T \cap \mathcal{L}_P \mid T \in \Sigma \,\}$ is the set of all answer sets of P which includes every p-answer set of (P, Φ), we show that any answer set removed from Δ by the procedure does not correspond to any p-answer set. Suppose $T \in \Sigma$. If $\phi^-_{c_i \prec c_j} \in T$ and $\exists T' \in \Sigma$ s.t. $\phi^+_{c_i \prec c_j} \in T'$, then there exist rules: $(\phi^+_{c_i \prec c_j} \leftarrow c_j, not\, c_i)$ and $(\phi^-_{c_i \prec c_j} \leftarrow c_i, not\, c_j)$ in P_Φ s.t. $c_i \in T \setminus T'$ and $c_j \in T' \setminus T$ with $(c_i \prec c_j) \in \Phi^*$. If there is no $\phi^+_{c_j \prec c_k}$ in T, there is no $c_k \in T \setminus T'$ such that $(c_j \prec c_k) \in \Phi^*$. Thus, $T \sqsubseteq T'$. As the p-answer sets of (P, Φ) are cycle-free, $T' \not\sqsubseteq T$ holds. Then, $T \cap \mathcal{L}_P$ cannot be a p-answer set of (P, Φ), so T is removed from Δ. Hence, for any p-answer set S of (P, Φ), there is a set $T \in \Delta$ s.t. $S = T \cap \mathcal{L}_P$. □

Example 4.2 Let (P, Φ) be the PLP such that

$$
\begin{array}{l}
P : p \mid q \mid r \leftarrow, \\
\quad\ \ s \leftarrow p. \\
\Phi : p \preceq q, \ \ r \preceq s.
\end{array}
$$

Then, the ϕ-program becomes

$$
\begin{array}{l}
P_\Phi : p \mid q \mid r \leftarrow, \\
\quad\ \ s \leftarrow p, \\
\quad\ \ \phi^+_{p \prec q} \leftarrow q, not\, p, \ \ \phi^-_{p \prec q} \leftarrow p, not\, q,
\end{array}
$$

$$\phi^+_{r\prec s} \leftarrow s, not\, r, \quad \phi^-_{r\prec s} \leftarrow r, not\, s.$$

First, put $\Sigma = \Delta = \{\{\, p, s, \phi^-_{p\prec q}, \phi^+_{r\prec s}\,\}, \{\, q, \phi^+_{p\prec q}\,\}, \{\, r, \phi^-_{r\prec s}\,\}\}$ as the set of answer sets of P_Φ. Next, for $\phi^-_{p\prec q}$ in the first answer set, $\phi^+_{p\prec q}$ is in the second answer set and there is no $\phi^-_{q\prec x}$ in the second one, so that the first one is discarded from Δ. Likewise, the third answer set is dropped from Δ. As a result, $\Delta = \{\{\, q, \phi^+_{p\prec q}\,\}\}$ and $\{\, q, \phi^+_{p\prec q}\,\} \cap \mathcal{L}_P = \{q\}$ is the unique p-answer set of (P, Φ).

When the p-answer sets of a PLP (P, Φ) have a cycle, the above procedure is sound but not complete for computing p-answer sets.

Example 4.3 Let (P, Φ) be a PLP such that P has three answer sets $S_1 = \{\, e_1, e_2\,\}$, $S_2 = \{\, e_3, e_4\,\}$, $S_3 = \{\, e_5, e_6\,\}$, and $\Phi = \{\, e_2 \preceq e_3,\ e_4 \preceq e_5,\ e_6 \preceq e_1\,\}$. There is a cycle $S_1 \sqsubseteq S_2 \sqsubseteq S_3 \sqsubseteq S_1$. However, $S_2 \sqsubseteq S_1$ is not known by comparing S_1 and S_2 (with ϕ-atoms). In this case, all S_1, S_2 and S_3 are discarded from Δ in the procedure.

It is generally difficult to judge whether the p-answer sets of a PLP have a cycle or not. In fact, the structure of Φ is not useful to know the existence of a cycle in the above example.

4.2 Complexity result

We next address the computational complexity of PLP. A PLP (P, Φ) is *propositional* if P contains no variable and Φ is a set of priorities on ground elements from $\mathcal{L}^*_P$. In this section, we consider propositional PLPs.

We briefly review some basic concepts of computational complexity. The class P (resp. NP) represents the set of all decision problems solvable in polynomial time by deterministic (resp. non-deterministic) Turing machines. The *polynomial hierarchy* consists of classes Δ^P_k, Σ^P_k, and Π^P_k defined as

$$\Delta^P_0 = \Sigma^P_0 = \Pi^P_0 = \mathrm{P},$$

$$\Delta^P_{k+1} = \mathrm{P}^{\Sigma^P_k}, \quad \Sigma^P_{k+1} = \mathrm{NP}^{\Sigma^P_k}, \quad \Pi^P_{k+1} = \text{co-}\Sigma^P_{k+1} \ \ (k \geq 0).$$

In particular, Δ^P_1=P, Σ^P_1=NP, and Π^P_1=co-NP.

In the above, Δ^P_{k+1} (resp. Σ^P_{k+1}) is the set of problems solvable deterministically (resp. non-deterministically) in polynomial time with an oracle for the problems in Σ^P_k. The class Π^P_{k+1} consists of problems whose complements are in Σ^P_{k+1}.

For GEDPs, the next results hold.

Lemma 4.3 *[26] (complexity result for GEDP) Let P be a propositional GEDP. Then,*

(i) Deciding the existence of an answer set of P is Σ_P^2-complete.
(ii) Deciding whether a literal is true in some answer set of P is Σ_P^2-complete.
(iii) Deciding whether a literal is true in every answer set of P is Π_P^2-complete.

The complexities of problems in PLP are as follows.

Lemma 4.4 *(checking a p-answer set) Let (P, Φ) be a propositional PLP. Given a set S of literals, deciding whether S is a p-answer set of (P, Φ) is in Π_P^2.*

Proof. Given S, the reduct P^S is constructible in polynomial time. S is not an answer set of P iff there is a set $S' \subset S$ which satisfies every rule in P^S. Since a guess for S' is verified in polynomial time, deciding whether S is an answer set of P is in co-NP. On the other hand, given an answer set S, checking whether $S \sqsubset T$ holds for another answer set T of P is done in polynomial time. If such T does not exist, S is a p-answer set. As any answer set T of P is decided with a call to an NP-oracle, the problem is in co-NP$^{NP} = \Pi_P^2$. □

The next lemma is used in the proof of Theorem 4.6. (The expression is changed in our context.)

Lemma 4.5 *[15, Theorem 23] (complexity result for minimal abduction) Let P be a propositional normal disjunctive program and O a ground atom representing an observation. Then, deciding whether an atom is included in some credulous minimal explanation of O in P is Σ_3^P-complete.*

Theorem 4.6 *(complexity result for PLP) Let (P, Φ) be a propositional PLP. Then,*

(i) Deciding the existence of a p-answer set of (P, Φ) is Σ_P^2-complete.
(ii) Deciding whether a literal is true in some p-answer set of (P, Φ) is Σ_P^3-complete.
(iii) Deciding whether a literal is true in every p-answer set of (P, Φ) is Π_P^3-complete.

Proof. (i) (P, Φ) has a p-answer set iff P has an answer set. Then, the result holds by Lemma 4.3. (ii) To see the membership in Σ_P^3, first guess a set containing a literal. Then, whether it is a p-answer set can be verified in polynomial time using a Π_P^2 oracle (Lemma 4.4) and thus decidable with a query to a Σ_P^2 oracle. Hence, the problem is in Σ_P^3. On the other hand, deciding whether a literal is included in some (credulous) minimal explanation is Σ_3^P-complete in

NDPs (Lemma 4.5). Since GEDPs strictly include NDPs, the corresponding decision problem in GEDPs is Σ_3^P-hard. As minimal explanations are computed via p-answer sets (Theorem 3.2), the problem of deciding whether a literal is true in some p-answer set is also Σ_P^3-hard. (iii) is a complementary problem of (ii). Hence, the result holds by (ii). □

Corollary 4.7 *(complexity result for non-disjunctive PLP) Let (P, Φ) be a propositional PLP such that P is an ELP. Then,*

(i) Deciding the existence of a p-answer set of (P, Φ) is NP-complete.
(ii) Deciding whether a literal is true in some p-answer set of (P, Φ) is Σ_P^2-complete.
(iii) Deciding whether a literal is true in every p-answer set of (P, Φ) is Π_P^2-complete.

Proof. In the absence of disjunctions in a program, the complexity of each problem reduces in one level of the polynomial hierarchy. Then, the results hold. □

Comparing the results of Lemma 4.3 and Theorem 4.6, an introduction of priorities to a program causes an increase in complexity by one level of the polynomial hierarchy (for the problems of (ii) and (iii)).

5 Related work

5.1 Prioritized Logic Programming

In this section, we compare the PLP with the existing prioritized logic programming systems. We focus on the following points for comparison.

Priority: The definition of priority relations.
Language: The class of programs on which priority reasoning is introduced.
Commonsense reasoning: Applications to commonsense reasoning in AI.

5.1.1 Stratified programs

Stratified programs introduce a restricted form of priorities to logic programs.

Priority: In stratified programs priorities over atoms are decided by the syntactic structure of a program. By contrast, priorities in PLP are specified separately from the program. Hence, different programmers can specify different priorities in the same program (as far as they do not contradict each

other) without changing the body of the program. In converse, any change in a program does not affect priorities. Moreover, priorities in PLP generalize those in stratified programs in the following sense. First, any stratification of a program can be expressed in terms of priorities in a PLP (Theorem 3.11), but the converse transformation, representing arbitrary priorities Φ in a single stratification, is generally impossible. Secondly, in a stratified program *every* atom must be *ranked* according to the syntax of the program, while no such restriction exists in PLP and priority are defined on any subset of $\mathcal{L}_P^*$. Thirdly, PLP can express priorities between not only atoms but also literals and NAF formulas in GEDPs.

Language: Stratified programs are defined as a subset of normal disjunctive programs. A PLP is defined for GEDPs which include normal disjunctive programs.

Commonsense reasoning: Stratified programs can realize a restricted version of prioritized circumscription [18]. Those restrictions are substantially relaxed in PLP (Section 3.3.2). Further comparison is presented in Section 5.2.3.

5.1.2 Brewka

Brewka [6] introduces priorities to Reiter's default logic to resolve conflicts between default rules. In [7] a version of logic programming is proposed.

Priority: A strict partial order $<$, i.e., an *irreflexive* and transitive relation, is introduced over rules. By contrast, we used a reflexive and transitive relation over literals and NAF formulas. Prioritization over rules is simulated in PLP as presented in Sections 2.3 and 3.2.2. This point is also discussed later in Section 5.1.8.

Reflexive relations permit to represent cyclic priorities which are useful for representing *tie* situations. An example of this is demonstrated for representing priorities over fixed predicates of circumscription in Section 3.3. Note that in PLP the existence of reflexive relations between elements and the absence of relations are different in effect. For instance, consider the theory $T = \{\, p \leftarrow q \,\}$ where p has the predicate to be minimized and q has the fixed predicate. It is represented in the PLP (Π, Φ_{CIRC}) with

$$\begin{aligned} \Pi &= \{\, p \leftarrow q,\ \ q \mid not\, q \leftarrow \}, \\ \Phi_{CIRC} &= \{\, p \preceq notp,\ q \preceq not\, q,\ not\, q \preceq q \,\}. \end{aligned}$$

Then, the program has two p-answer sets $\emptyset$ and $\{p, q\}$ which correspond to the two Herbrand models of the circumscription of T. If we represent the equal priority simply by not mentioning any priority between q and $not\, q$, Π with

$\Phi'_{CIRC} = \{ p \preceq notp \}$ has the unique p-answer set $\emptyset$. The another model $\{p, q\}$ does not become a p-answer set because there is no priority to select it. Thus, a reflexive relation is effective for representing tie situations which are not affected by other priorities. (See also the comparison of priority in Section 5.1.3.

Language: [7] considers ELPs which are a strict subclass of GEDPs. The well-founded semantics is considered as an underlying semantics.

Commonsense reasoning: The primary interest of Brewka is to resolve conflicts between default rules. PLP is used for not only default reasoning but other (prioritized) commonsense reasoning such as abduction and circumscription. On the other hand, Brewka [7] introduces a method of encoding preference information in a program and using them to reason about priorities. The PLP framework would be also extended in this direction but it is not addressed in this paper. [12]

5.1.3 Brewka and Eiter

Brewka and Eiter [8] introduce preference over answer sets in extended logic programs.

Priority: In [8] a strict partial order is defined over rules. Hence, the same argument as in the comparison with Brewka is applied. Moreover, [8] defines a preferred answer sets for *fully prioritized* programs. For instance, consider the program

$$
\begin{aligned}
&r_1 : a \leftarrow c, \mathit{not}\, b,\\
&r_2 : b \leftarrow d, \mathit{not}\, a,\\
&r_3 : c \leftarrow \mathit{not}\, d,\\
&r_4 : d \leftarrow \mathit{not}\, c,
\end{aligned}
$$

with the priority $r_2 \preceq r_1$ (r_1 is preferred over r_2). In this case, they consider a total-order over rules which is compatible with $r_2 \preceq r_1$ (called *full prioritization*). Their preferred answer set then becomes $\{a, c\}$ if $r_4 \preceq r_3 \preceq r_2 \preceq r_1$ for instance, while it becomes $\{b, d\}$ if $r_2 \preceq r_1 \preceq r_3 \preceq r_4$. On the other hand, in PLP the p-answer set is selected according to the existing priority $r_2 \preceq r_1$. In the above program, using the transformation for rule prioritization in Section 2.3, the PLP expression of the above program becomes

$$
\Pi : a \leftarrow c, \mathit{not}\, b\,, \quad r_1 \leftarrow c, \mathit{not}\, b\,,
$$

[12] Priorities with preconditions, which is presented in an example of [7], is also encoded in PLP using the technique of Section 2.3.

$b \leftarrow d, \mathit{not}\, a\,, \quad r_2 \leftarrow d, \mathit{not}\, a\,,$
$c \leftarrow \mathit{not}\, d\,,$
$d \leftarrow \mathit{not}\, c\,,$
$\Phi : r_2 \preceq r_1.$

Then, (Π, Φ) has the unique p-answer set $\{\, a, c, r_1 \,\}$ which corresponds to $\{a, c\}$.

Generally, in [8] the absence of priority between rules r_i and r_j implies two possibilities $r_i \preceq r_j$ and $r_j \preceq r_i$, which are independent of the existing priorities. On the other hand, in PLP the existing priorities dominate the selection of p-answer sets, and the absence of priorities means a selection which may vary according to the existing priorities. In the above program, r_1 has a priority over r_2, then an answer set which includes r_1 is selected as the unique p-answer set (and consequently, r_3 is preferred over r_4). If one desires to consider two possibilities of the preference between r_3 and r_4 independent of the existing $r_2 \preceq r_1$, it is done in PLP by explicitly specifying symmetric priorities $r_3 \preceq r_4$ and $r_4 \preceq r_3$.

Language: Their preferred answer set semantics is defined for ELPs which are a strict subclass of GEDPs.

Commonsense reasoning: Their primary concern is to resolve conflicting multiple answer sets and no application to other nonmonotonic formalisms is presented.

There are some other important differences between [8] and ours.

Monotonicity vs. Nonmonotonicity: Their framework is monotonic with respect to the introduction of preference information. That is, introducing priorities monotonically reduces the number of answer sets. This means that once some conclusion is believed by the current preference knowledge, there is no way to invalidate the conclusion by introducing new preference knowledge. By contrast, in PLP adding preference information may nonmonotonically revise the previous beliefs (Proposition 2.2).

Preference information is possibly incomplete. Then, the p-answer sets select answer sets according to the priorities available in Φ. However, the selection might change by the introduction of new preference information. Such a change often happens in the real life. For example, we make a plan to manage daily jobs according to their priorities, while we are obliged to change the plan when an urgent job (with the highest priority) comes up. Thus, we consider the nonmonotonic aspect of prioritized reasoning is important and useful in commonsense reasoning.

Principles of prioritized reasoning: Brewka and Eiter also introduce gen-

eral principles for priorities as follows.

Principle I: Let B_1 and B_2 be two belief sets of a prioritized theory $(T, <)$ generated by the set of (ground) rules $R \cup \{d_1\}$ and $R \cup \{d_2\}$, where d_1, $d_2 \notin R$, respectively. If d_1 is preferred over d_2, then B_2 is not a (maximally) preferred belief set of T.

Principle II: Let B be a preferred belief set of a prioritized theory $(T, <)$ and r a (ground) rule such that at least one prerequisite of r is not in B. Then B is a preferred belief set of $(T \cup \{r\}, <')$ whenever $<'$ agrees with $<$ on priorities among rules in T.

In the above, belief sets corresponds to answer sets in our context, and a prerequisite means a literal (without NAF) in the body of a rule. Roughly speaking, the first principle means that a belief set is preferred if it is generated by a rule with a relatively higher priority. The second principle says that adding a rule which is not applicable in a preferred belief set never changes this preference as far as the preference over old knowledge is kept.

Our p-answer sets satisfy the Principle I. That is, if answer sets S_1 and S_2 are respectively produced by rules r_1 and r_2, and the priority $r_2 \preceq r_1$ is given, then S_1 is preferred to S_2 as presented in Section 3.2.2. However, p-answer sets do not satisfy the Principle II in general. Take for instance, the following program P from [8]:

$$r_1 : b \leftarrow a,\ not\ \neg b,$$
$$r_2 : \neg a \leftarrow not\ a,$$
$$r_3 : a \leftarrow not\ \neg a,$$

where r_1 is preferred over r_2, and r_2 is preferred over r_3. The program has two answer sets $S_1 = \{\neg a\}$ and $S_2 = \{a, b\}$. Regarding the Principle II, S_1 is the preferred answer set of $\{r_2, r_3\}$, then adding r_1, whose prerequisite a is not satisfied by S_1, should be ignored in selecting preferred answer sets regardless of the priority on r_1. As a result, Brewka and Eiter select S_1 as the preferred answer set of P. On the other hand, in PLP using the program transformation in Section 2.3, the p-answer set becomes $\{\, a, b, r_1, r_3 \,\}$, which corresponds to S_2.

In contrast to Brewka and Eiter's Principle II, our selection of S_2 is explained as follows. S_2 is the preferred answer set of $\{r_1, r_3\}$. By adding r_2 to $\{r_1, r_3\}$, we keep S_2 as the p-answer set of $\{r_1, r_2, r_3\}$. That is, *the introduction of* r_2*, whose priority is lower than* r_1*, does not affect the consequence of* r_1. Brewka and Eiter's preferred answer sets do not satisfy this property.

Hence, we consider that Brewka and Eiter's Principle II is optional, and the utility of the property would depend on applications.

5.1.4 Wang et al.

Wang *et al.* [56] introduce *priority logic* having the following feature.

Priority: A priority constraint, which is not necessarily a partial order, is defined over rules.

Language: They consider rules of the form $\beta \leftarrow \alpha_1, \ldots, \alpha_m$ where β and α_i are first-order formulas. The meaning of a program is defined by stable arguments.

Commonsense reasoning: (Propositional) default theories and defeasible inheritance networks are represented by priority logic.

Their claim is that nonmonotonic reasoning is replaced by monotonic inference plus priority constraints. This view is interesting, but it is not clear how general this replacement is possible. According to [31], priority logic and Reiter's default logic have the same expressive power. From the complexity viewpoint, PLP is more expressive than default logic (Section 4.2), thereby more expressive than priority logic.

5.1.5 Zang and Foo

Zang and Foo [57] introduce yet another "PLP", which is close to [8].

Priority: A strict partial order is defined over rules.

Language: Preferred answer sets are introduced for ELPs.

Commonsense reasoning: Their prioritized logic programs are devised to resolve conflicting multiple answer sets. Its application to program update is presented in [58], while no explicit connection to other nonmonotonic formalism is presented.

Zang and Foo also introduce the framework of *dynamic preference* like [7], which enables a programmer to dynamically specify preference information in a program.

5.1.6 Buccafurri et al.

Buccafurri *et al.* introduce a language called *disjunctive ordered logic* (DOL). In [10] the authors introduce another language called DLP$^<$.

Priority: A strict partial order is defined over (sets) of rules.

Language: Each language handles extended disjunctive programs (DOL includes no NAF). DLP$^<$ extends the answer set semantics, while DOL considers a different semantics.

Commonsense reasoning: DOL realizes defeasible reasoning by preferring more specific rules, and DLP$^<$ effectively realizes inheritance.

The above two languages introduce priorities to disjunctive logic programs, but the purpose is different from PLP. DOL and DLP$^<$ introduce priorities to resolve conflicts in default reasoning, while PLP introduces priorities to reduce non-determinism which arises in disjunctive logic programs. From the complexity viewpoint, DOL and DLP$^<$ are at the same complexity level as disjunctive logic programming, which is in contrast to PLP.

5.1.7 Others

Priority: Priorities are defined over (conflicting) default rules [1,13,24,21] and (sets of) atoms [45]. [25] uses priorities with preconditions.

Language: Extended logic programs [1,13,24,21] and Datalog with integrity constraints [45], which are all strict subclasses of GEDPs. [25] considers constraint (definite) logic programs.

Commonsense reasoning: Analyti and Pramanik [1] introduce priorities to resolve contradiction in a program. Dimopoulos and Kakas [13] replace NAF by prioritized reasoning, and apply their method to temporal reasoning. Gelfond and Son [21] introduce meta-level axioms for prioritized defeasible reasoning. Pradhan and Minker [45] and Grosof [24] use priorities for combining conflicting knowledge bases of multi-agents. These work introduce priorities to select intended conclusions from conflicting knowledge. By contrast, PLP is used for not only resolving confliction, but reducing various kinds of non-determinism in logic programming and realizing various forms of commonsense reasoning. Govindarajan *et al.* [25] use priority knowledge to select best solutions in the context of constraint logic programming.

5.1.8 Rule-based vs. Literal-based

As presented above, most prioritized LP-languages introduce priorities between rules. It is in contrast to PLP in which priorities are specified over literals and NAF-formulas. We discussed in Sections 2.3 and 3.2.2 how to express priorities between (default) rules in PLP. Thus, PLP can simulate reasoning with prioritized rules. On the other hand, it is unknown how to specify priorities over disjunctive or abductive knowledge in terms of languages with rule-based preference.

PLP can realize abduction, default reasoning, circumscription, and their prioritized versions. We compare our PLP methods with the existing frameworks for (prioritized) commonsense reasoning in AI.

5.2.1 (Prioritized) Abduction

Minimal explanations are usually computed by comparing generated explanations. In the context of abductive logic programming, minimal explanations are computed by selecting $\mathcal{A}$-minimal answer sets of a GEDP (Lemma 3.1). On the other hand, PLP encodes the selection of minimal explanations in the language using the priorities Φ_{MA}.[13] Moreover, PLP can specify further preference over minimal explanations as in Section 3.1.2. Eiter and Gottlob [14] introduce priorities to abduction. In their framework, the set of abducibles are partitioned into levels of priorities and explanations containing the most preferable hypotheses are selected. Such a hierarchical structure is easily expressed in our prioritized abduction. However, the converse translation, representing arbitrary priorities over abducibles in a single abducible hierarchy is generally impossible.

5.2.2 (Prioritized) Default Reasoning

There are several systems which incorporate priorities into default reasoning. For instance, Baader and Hollunder [3], Brewka [6], Delgrande and Schaub [12], and Rintanen [50] introduce a strict partial/total order over (normal) defaults, These formalisms specify the order of default applications in constructing default extensions. Our approach is a bit different from them in the sense that we compare preference between extension bases, rather than specifying the order of rule applications in the process of computation. Resolving conflicting defaults has been discussed by several researchers in the context of extended logic programs [34,43,28]. These approaches use program transformations to resolve contradiction in a program. By contrast, PLP expresses priorities over defaults outside a program, which enables us to specify priorities independent of a program.

[13] [16] presents an algorithm of computing minimal explanations in (function-free) definite logic programs via answer sets of disjunctive logic programs.

5.2.3 (Prioritized) Circumscription

Several researchers propose methods for compiling (prioritized) circumscription into logic programs. Gelfond and Lifschitz [18] provide a method of compiling prioritized circumscription into stratified logic programs. In their framework, however, every clause is assumed to contain at most one variable predicate and no fixed predicate. Moreover, they do not transform any clause having more than one disjunct included in the same strata nor any negative clause in first-order theories. By contrast, the PLP expression of prioritized circumscription presented in this paper has no such restriction. Sakama and Inoue [52] present another transformation from circumscription to a GEDP. The transformation is not necessarily done in polynomial-time as it requires the computation of *characteristic clauses* [27]. The transformation of [52] is extended to prioritized circumscription by several researchers [11,55], but it still requires the computation of characteristic clauses.

5.2.4 PLP vs. NMR

We have presented methods of realizing (prioritized) commonsense reasoning in terms of PLP. On the other hand, it is unknown how to express PLP in terms of the existing frameworks of nonmonotonic reasoning in general. For instance, a predicate hierarchy in prioritized circumscription is expressed by a set of priorities in a PLP, but the converse translation, representing a set of priorities with a pre-order priority relation in a single predicate hierarchy, is generally impossible.[14] From the complexity viewpoint, expressing PLPs in terms of existing major nonmonotonic logics, which are at the second level of the polynomial hierarchy [22,31], is most unlikely possible.

6 Concluding remarks

Prioritized logic programming realizes reasoning with priorities, which is useful for reducing non-determinism in logic programming. PLP can specify preference knowledge separate from programming knowledge. This means that a control part which determines strategies for problem-solving is separated from a logic part which specifies a declarative background knowledge. Such a separation accords with Kowalski's principle of logic programming [33]. We introduced PLP under the answer set semantics, while an analogous mechanism is easily devised for other semantics of logic programming.

[14] Grosof [23] introduced a generalized circumscription having pre-order priority relations over first-order predicates.

From the AI side, PLP can express various forms of commonsense reasoning in the single language. This is meaningful for comparing commonsense reasoning in different languages and for better understanding the nature of priorities in each reasoning. Moreover, such characterization exploits strong links between logic programming and commonsense reasoning in AI.

Currently, PLP has no efficient implementation. The selection algorithm introduced in Section 4.1 requires computation of every answer set in advance. On the other hand, translating PLPs to some existing LP language would provide an immediate way of implementing PLP. Some hints might be in studies like [12] which presents a method of embedding priorities into default theories. However, it is unlikely that PLPs can be efficiently translated into existing LP languages in general. This is because the computational complexity of PLP is at the third level of the polynomial hierarchy, while the complexities of most existing LP languages lie within the second level. The complexity result Corollary 4.7 suggests the existence of a polynomial-time transformation from non-disjunctive PLPs to disjunctive LPs. However, it is at present an open question whether there exists a modular transformation for this purpose.

There are several directions for future research. The present PLP framework specifies priorities outside a program. Extending the language to be able to specify dynamic priorities inside a program will increase the utility of PLP. Examples of this direction are in [7,57]. In this paper, we considered a problem setting such that priorities are given in advance. On the other hand, Inoue and Sakama [30] introduce a framework of *preference abduction* in which preference information is abduced by an observation. Thus, preference abduction is used for revising a PLP; when new information arrives at a PLP, preference abduction can produce new priorities.

Commonsense (nonmonotonic) reasoning and reasoning with priorities are closely related. Shoham [51] argues that the non-standard behavior of nonmonotonic reasoning is due to preference mechanisms within it. According to Shoham, "nonmonotonic logics are the result of associating a standard logic with a preference relation on models". Examples of research along this line are [13,56,9]. Using the program transformation from a GEDP to a positive disjunctive program (plus integrity constraints) in [26], PLP is also expressed in terms of a monotonic positive disjunctive program plus priorities. However, it is not clear whether such a translation, from nonmonotonic logics to monotonic logics plus priorities, is generally possible or not. The general correspondence between nonmonotonic reasoning and prioritized reasoning is a challenging topic.

Acknowledgements

The authors thank Thomas Eiter for useful discussion on the subject of this paper. We also thank anonymous referees for comments on an earlier draft of this paper.

References

[1] A. Analyti and S. Pramanik, Reliable semantics for extended logic programs with rule prioritization, *Journal of Logic and Computation* 5(3)(1995) 303–324.

[2] K. R. Apt, H. A. Blair, and A. Walker, Towards a theory of declarative knowledge, in: J. Minker, ed., *Foundations of Deductive Databases and Logic Programming* (Morgan Kaufmann, Los Altos, CA, 1988) 89–148.

[3] F. Baader and B. Hollunder, Priorities on defaults with prerequisites, and their application in treating specificity in terminological default logic, *Journal of Automated Reasoning* 15(1995) 41–68.

[4] C. Baral and M. Gelfond, Logic programming and knowledge representation, *Journal of Logic Programming* 19-20(1994) 73–148.

[5] G. Bossu and P. Siegel, Saturation, nonmonotonic reasoning and the closed world assumption, *Artificial Intelligence* 25 (1995) 13–63.

[6] G. Brewka, Reasoning about priorities in default logic, in: *Proceedings of the 12th National Conference on Artificial Intelligence* (MIT Press, Cambridge, MA, 1994) 940–945.

[7] G. Brewka, Well-founded semantics for extended logic programs with dynamic preferences, *Journal of AI Research* 4(1996) 19–36.

[8] G. Brewka and T. Eiter, Preferred answer sets for extended logic programs, *Artificial Intelligence* 109(1999) 297–356.

[9] F. Buccafurri, N. Leone, and P. Rullo, Semantics and expressiveness of disjunctive ordered logic, *Annals of Mathematics and Artificial Intelligence* 25 (1999) 311–337.

[10] F. Buccafurri, W. Faber, and N. Leone, Disjunctive logic programs with inheritance, in: *Proceedings of the 1999 International Conference on Logic Programming* (MIT Press, Cambridge, MA, 1999) 79–93.

[11] J. Chen, Embedding prioritized circumscription in logic programs, in: *Proceedings of the 10th International Symposium on Foundations of Intelligent Systems (ISMIS'97), Lecture Notes in Artificial Intelligence* 1325 (Springer-Verlag, Berlin, 1997) 50–59.

[12] J. Delgrande and T. Schaub, Compiling reasoning with and about preference into default logic, in: *Proceedings of the 15th International Joint Conference on Artificial Intelligence* (Morgan Kaufmann, Los Altos, CA, 1997) 168–174.

[13] Y. Dimopoulos and A. C. Kakas, Logic programming without negation as failure, in: *Proceedings of the 1995 International Logic Programming Symposium* (MIT Press, Cambridge, MA, 1995) 369–383.

[14] T. Eiter and G. Gottlob, The complexity of logic-based abduction, *J. ACM* 42(1995) 3–42.

[15] T. Eiter, G. Gottlob, and N. Leone, Abduction from logic programs: semantics and complexity, *Theoretical Computer Science* 189(1-2) (1997) 129–177.

[16] T. Eiter, W. Faber, N. Leone, and G. Pfeifer, The diagnosis front-end of the dlv system, *AI Communications* 12(1999) 99–111, IOS Press.

[17] D. W. Etherington, Formalizing nonmonotonic reasoning systems, *Artificial Intelligence* 31(1987) 41–85.

[18] M. Gelfond and V. Lifschitz, Compiling circumscriptive theories into logic programs, in: *Proceedings of the 7th National Conference on Artificial Intelligence* (MIT Press, Cambridge, MA, 1988) 455–459.

[19] M. Gelfond, Epistemic approach to formalization of commonsense reasoning, Technical Report TR-91-2, University of Texas at El Paso, 1991.

[20] M. Gelfond and V. Lifschitz, Classical negation in logic programs and disjunctive databases, *New Generation Computing* 9(3,4) (1991) 365–385.

[21] M. Gelfond and T. C. Son, Reasoning with prioritized defaults, in: *Proceedings of the 3rd International Workshop on Logic Programming and Knowledge Representation, Lecture Notes in Artificial Intelligence* 1471 (Springer-Verlag, Berlin, 1998) 164–223.

[22] G. Gottlob, Complexity results for nonmonotonic logics, *Journal of Logic and Computation* 2(3) (1992) 397–425.

[23] B. N. Grosof, Generalizing prioritization, in: *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning* (Morgan Kaufmann, Los Altos, CA, 1991) 289–300.

[24] B. N. Grosof, Prioritized conflict handling for logic programs, in: *Proceedings of the 1997 International Logic Programming Symposium* (MIT Press, Cambridge, MA, 1997), 197–211.

[25] K. Govindarajan, B. Jayaraman, and S. Mantha, Preference logic programming, in: *Proceedings of the 12th International Conference on Logic Programming* (MIT Press, Cambridge, MA, 1995), 731–745.

[26] K. Inoue and C. Sakama, Negation as failure in the head, *Journal of Logic Programming* 35(1) (1998) 39–78. A shorter version in: On positive occurrences of negation as failure, in: *Proceedings of the 4th International Conference on*

Principles of Knowledge Representation and Reasoning (Morgan Kaufmann, Los Altos, CA, 1994) 293–304.

[27] K. Inoue, Linear resolution for consequence finding, *Artificial Intelligence* 56 (1992) 301–353.

[28] K. Inoue, Hypothetical reasoning in logic programs, *Journal of Logic Programming* 18(3) (1994) 191–227.

[29] K. Inoue and C. Sakama, A fixpoint characterization of abductive logic programs, *Journal of Logic Programming* 27(2) (1996) 107–136. A shorter version: Transforming abductive logic programs to disjunctive programs, in: *Proceedings of the 10th International Conference on Logic Programming* (MIT Press, Cambridge, MA, 1993) 335–353.

[30] K. Inoue and C. Sakama, Abducing priorities to derive intended conclusions, in: *Proceedings of the 16th International Joint Conference on Artificial Intelligence* (Morgan Kaufmann, Los Altos, CA, 1999) 44–49.

[31] T. Janhunen, On the intertranslatability of autoepistemic, default and priority logics, and parallel circumscription, in: *Proceedings of the European Workshop on Logics in Artificial Intelligence (JELIA'98), Lecture Notes in Artificial Intelligence* 1489 (Springer-Verlag, Berlin, 1998) 216–232.

[32] A. C. Kakas, R. A. Kowalski, and F. Toni, Abductive logic programming, *Journal of Logic and Computation* 2 (1992) 719–770.

[33] R. A. Kowalski, Algorithm = Logic + Control, *Communications of the ACM* 22(1979) 424–435.

[34] R. A. Kowalski and F. Sadri, Logic programs with exception, *New Generation Computing* 9(3,4) (1991) 387–400.

[35] V. Lifschitz, Computing circumscription, in: *Proceedings of the 9th International Joint Conference on Artificial Intelligence* (Morgan Kaufmann, Los Altos, CA, 1985) 121–127.

[36] V. Lifschitz, On the satisfiability of circumscription, *Artificial Intelligence* 28(1986) 17–27.

[37] V. Lifschitz and T. Y. C. Woo, Answer sets in general nonmonotonic reasoning (preliminary report), in: *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, (Morgan Kaufmann, Los Altos, CA, 1992) 603–614.

[38] V. Lifschitz, Minimal belief and negation as failure, *Artificial Intelligence* 70(1994) 53–72.

[39] V. Lifschitz, Circumscription, in: D. M. Gabbay *et al* (eds.), *Handbook of Logic in Artificial Intelligence and Logic Programming*, vol.3 (Clarendon Press, Oxford, 1994) 297–352.

[40] J. W. Lloyd, *Foundations of Logic Programming*, 2nd edition, (Springer-Verlag, Berlin 1987).

[41] J. Lobo, J. Minker, and A. Rajasekar, *Foundations of disjunctive logic programming* (MIT Press, Cambridge, MA, 1992).

[42] J. McCarthy, Circumscription – a form of nonmonotonic reasoning, *Artificial Intelligence* 13(1980) 27–39.

[43] L. M. Pereira, J. J. Alferes and N. Aparicio, Contradiction removal within well-founded semantics, in *Proceedings of the 1st International Workshop on Logic Programming and Nonmonotonic Reasoning*, (MIT Press, Cambridge, MA, 1991) 105–119.

[44] D. Poole, A logical framework for default reasoning, *Artificial Intelligence* 36(1) (1988) 27–47.

[45] S. Pradhan and J. Minker, Using priorities to combine knowledge bases, *Journal of Intelligent and Cooperative Information Systems* 5 (2,3) (1996) 333-364.

[46] T. C. Przymusinski, On the declarative semantics of deductive databases and logic programs, in: J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming* (Morgan Kaufmann, Los Altos, CA, 1988) 193–216.

[47] T. C. Przymusinski, Stable semantics for disjunctive programs, *New Generation Computing* 9 (3,4) (1991) 401–424.

[48] R. Reiter, A logic for default reasoning, *Artificial Intelligence* 13(1980) 81–132.

[49] R. Reiter and G. Criscuolo, On interacting defaults, in: *Proceedings of the 7th International Joint Conference on Artificial Intelligence* (Morgan Kaufmann, Los Altos, CA, 1981) 270–276.

[50] J. Rintanen, Lexicographic priorities in default logic, *Artificial Intelligence* 106 (1998) 221–265.

[51] Y. Shoham, Nonmonotonic logics: meaning and utility, in: *Proceedings of the 10th International Joint Conference on Artificial Intelligence* (Morgan Kaufmann, Los Altos, CA, 1987) 388–393.

[52] C. Sakama and K. Inoue, Embedding circumscriptive theories in general disjunctive programs, in: *Proceedings of the 3rd International Conference on Logic Programming and Nonmonotonic Reasoning, Lecture Notes in Artificial Intelligence* 928 (Springer-Verlag, Berlin, 1995) 344–357.

[53] C. Sakama and K. Inoue, Representing priorities in logic programs, in: *Proceedings of the 1996 Joint International Conference and Symposium on Logic Programming* (MIT Press, Cambridge, MA, 1996) 82–96.

[54] M. E. Stickel, Rationale and methods for abductive reasoning in natural-language interpretation, in: *Proceedings of the International Scientific Symposium on Natural Language and Logic, Lecture Notes in Artificial Intelligence* 459 (Springer-Verlag, Berlin, 1989) 233–252.

[55] T. Wakaki and K. Satoh, Compiling prioritized circumscription into extended logic programs, in: *Proceedings of the 15th International Joint Conference on Artificial Intelligence* (Morgan Kaufmann, Los Altos, CA, 1997) 182–187.

[56] X. Wang, J.-H. You, and L.-Y. Yuan, Nonmonotonic reasoning by monotonic inferences with priority constraints, in: *Proceedings of the 2nd International Workshop on Nonmonotonic Extensions of Logic Programming, Lecture Notes in Artificial Intelligence* 1216 (Springer-Verlag, Berlin, 1996) 91–109.

[57] Y. Zang and N. Foo, Answer sets for prioritized logic programs, in: *Proceedings of the 1997 International Logic Symposium* (MIT Press, Cambridge, MA, 1997) 69–83.

[58] Y. Zang and N. Foo, Updating logic programs, in: *Proceedings of the 13th European Conference on Artificial Intelligence* (John Wiley & Sons, UK, 1998) 403–407.