Representing Priorities in Logic Programs

Chiaki Sakama
Department of Computer and Communication Sciences
Wakayama University
Sakaedani, Wakayama 640, Japan
`sakama@sys.wakayama-u.ac.jp`
`http://www.wakayama-u.ac.jp/~sakama`

Katsumi Inoue
Department of Information and Computer Sciences
Toyohashi University of Technology
Tempaku-cho, Toyohashi 441, Japan
`inoue@tutics.tut.ac.jp`
`http://ai.tutics.tut.ac.jp/~inoue`

Abstract

Reasoning with priorities is a central topic in knowledge representation. A number of techniques for prioritized reasoning have been developed in the field of AI, but existing logic programming lacks the mechanism of explicit representation of priorities in a program. In this paper, we introduce a framework for representing priorities in logic programming. *Prioritized logic programming* represents preference knowledge more naturally than stratified programs, and is used to reduce non-determinism in logic programming. Moreover, it can realize various forms of commonsense reasoning such as abduction, default reasoning, and prioritized circumscription. The proposed framework increases the expressive power of logic programming and exploits new applications in knowledge representation.

1 Introduction

Recent extensions of logic programming provide mechanisms of handling incomplete knowledge in many ways as normal, disjunctive, abductive, and extended logic programs. In these extended frameworks, each language introduces different kinds of non-determinism as

- multiple minimal models in a program,
- multiple explanations in abduction,
- conflicting extensions in default reasoning.

To reduce such non-determinism in programming knowledge, it is important to introduce explicit representation of preference and specify the intended meaning of a program.

Priorities in logic programming are conventionally expressed in terms of stratified negation. However, priorities in stratified programs are determined by the syntax of a program, and their application is restricted to programs having a single stratified structure. Also, existing abductive and disjunctive logic programs provide no mechanism of freely specifying preference over multiple solutions. In the field of AI, on the other hand, a number of prioritized reasoning systems are developed such as prioritized circumscription or prioritized default reasoning. Then our concern is whether such general prioritized reasoning systems in AI are realized in logic programming.

In this paper we present a framework of prioritized reasoning in logic programming. *Prioritized logic programming* introduces a mechanism of explicit representation of priorities in a program. The declarative semantics of such programs is given by the *preferred answer sets*, which distinguish answer sets according to programmers' preference. Prioritized logic programming can represent preference knowledge more naturally than stratified programs, and is used to reduce non-determinism in logic programming. Moreover, it can realize various forms of commonsense reasoning such as abduction, default reasoning, and prioritized circumscription. The proposed framework increases the expressive power of logic programming and exploits new applications in knowledge representation.

The rest of this paper is organized as follows. In Section 2, we introduce a framework of prioritized logic programming and present its properties. Section 3 presents applications of prioritized logic programming to various forms of commonsense reasoning. Section 4 discusses the expressive power of the proposed framework, and Section 5 summarizes the paper.

2 Prioritized Logic 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:

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

where each L_i is a positive or negative literal, and *not* is negation as failure. The left-hand side of the rule is the *head*, and the right-hand side is the *body*. A rule with the empty head is an *integrity constraint*. A rule with variables stands for the set of its ground instances as usual.

The class of GEDP is introduced in [LW92] as a subclass of *minimal belief and negation as failure* (MBNF). GEDP is a fairly general class of existing LP frameworks 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* [IS94], 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 ($k = l$).

The semantics of GEDP is given by the *answer set semantics*. First, let P be a GEDP without *not* (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 a *(consistent) answer set* of P iff S is a minimal set satisfying the following two conditions: (i) for each ground rule $L_1 \mid \ldots \mid L_l \leftarrow L_{l+1}, \ldots, L_m$ ($l \geq 1$) 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) S does not contain both L and $\neg L$ for any literal L.

Secondly, given any GEDP P and $S \subseteq \mathcal{L}_P$, let us consider the not-free GEDP P^S such that a rule $L_1 \mid \ldots \mid L_k \leftarrow L_{l+1}, \ldots, L_m$ is in P^S iff there is a ground rule of the form (1) from P such that $\{L_{k+1}, \ldots, L_l\} \subseteq S$ and $\{L_{m+1}, \ldots, L_n\} \cap S = \emptyset$. Then, S is an *answer set* of P iff S is an answer set of P^S. A GEDP having an answer set is called *consistent*.[1]

The above definition of answer sets reduces to that of Gelfond and Lifschitz [GL91] in EDP. Negation as failure in the head introduces additional utilities to logic programming. In particular, the following GEDP rule is useful in this paper:

$$L \mid not\, L \leftarrow .$$

The rule states that L is true or not, and has two answer sets $\{L\}$ and $\emptyset$.

Next, let $\mathcal{L}_P^* = \mathcal{L}_P \cup \{\, not\, L : L \in \mathcal{L}_P \,\}$. Then a reflexive and transitive relation $\preceq$ is defined between elements from $\mathcal{L}_P^*$. 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 priority over* e_1. We write $e_1 \prec e_2$ if $e_1 \preceq e_2$ and $e_2 \not\preceq e_1$. A priority relation over elements including variables $e_1(\mathbf{x}) \preceq e_2(\mathbf{y})$, where $\mathbf{x}$ and $\mathbf{y}$ are tuples of variables, stands for any relation $e_1(\mathbf{s}) \preceq e_2(\mathbf{t})$ for any instances $\mathbf{s}$ of $\mathbf{x}$ and $\mathbf{t}$ of $\mathbf{y}$.

A *prioritized logic program* (PLP) is defined as a pair (P, Φ) where P is a GEDP and Φ is a set of priority relations over elements from $\mathcal{L}_P^*$. The declarative semantics of PLP is defined using the answer sets. Given a PLP (P, Φ), suppose that S_1 and S_2 are two distinct answer sets of P. For any $e \in \mathcal{L}_P^*$, $e \in S_2 \setminus S_1$ means $S_2 \models e$ and $S_1 \not\models e$, where $S \models not\, L$ iff $S \not\models L$. Then, S_2 is *preferable* to S_1 (written $S_1 \preceq S_2$) if for some element $e_2 \in S_2 \setminus S_1$, (i) there is an element $e_1 \in S_1 \setminus S_2$ such that $e_1 \preceq e_2$, and (ii) there is no element $e_3 \in S_1 \setminus S_2$ such that $e_2 \prec e_3$. Here the relation $\preceq$ over answer sets is defined as reflexive and transitive, and the preference relation is defined by the closure. This preference definition means that $S_1 \preceq S_2$ holds iff $S_2 \setminus S_1$ has an element e_2 whose priority is higher than 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. Note that the condition (ii) is automatically satisfied if there is no priority

[1] In this paper we do not consider the contradictory answer set $\mathcal{L}_P$, since we are interested in consistent theories.

relation chained over more than two different elements (i.e., $e_1 \preceq e_2 \preceq e_3$ implies either $e_1 = e_2$ or $e_2 = e_3$). An answer set S of P is called a *preferred answer set* (or *p-answer set*, for short) of P if $S \preceq S'$ implies $S' \preceq S$ for any answer set S' of P.

By definition, (P, Φ) has a p-answer set if P has a finite number of answer sets. In particular, the p-answer sets of (P, Φ) coincide with the answer sets of P when $\Phi = \emptyset$. 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.

Prioritization in PLP is not necessarily monotonic, i.e., increasing priorities does not always decrease answer sets. For example, let $P = \{\, a \mid b \leftarrow,$ $b \mid c \leftarrow \}$, $\Phi_1 = \emptyset$, $\Phi_2 = \{\, a \preceq b \,\}$, and $\Phi_3 = \{\, a \preceq b,\ b \preceq c \,\}$. Then (P, Φ_1) has the p-answer sets $\{a, c\}$ and $\{b\}$; (P, Φ_2) has $\{b\}$; and (P, Φ_3) has $\{a, c\}$.

PLP is useful when a program has multiple answer sets and a programmer wants to filter them out according to her preference. For instance, indefinite aspects of disjunctive logic programming are reduced by PLP.

Example 2.1 Let P be the program

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

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 relation as $\Phi = \{\, \mathit{ignition\text{-}damaged} \preceq \mathit{battery\text{-}dead} \,\}$. Then, the p-answer set of (P, Φ) becomes $S = \{\, \mathit{turn\text{-}key},\ \neg \mathit{start},\ \mathit{battery\text{-}dead} \,\}$.

Note that the above situation is also expressed in a stratified program. Namely, rewriting the first rule by

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

S becomes the perfect model of the stratified program. However, such static stratification is not always useful when a situation changes dynamically. Suppose that the reasoner later finds that the car-radio works and there is the integrity constraint

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

saying that a radio does not work with a dead battery. In this case, she wants to get the alternative solution *ignition-damaged*, but it is not obtained from the stratified program. By contrast, if *radio-work* $\leftarrow$ is added to P, in the presence of the integrity constraint the p-answer set becomes $\{\, \mathit{turn\text{-}key},\ \neg \mathit{start},\ \mathit{radio\text{-}work},\ \mathit{ignition\text{-}damaged} \,\}$, as intended.

Thus PLP can select appropriate answer sets according to the change of the situation. Note that any knowledge which is irrelevant to priority relations is not affected by the selection of p-answer sets. For example, suppose that in the above program the first disjunctive rule is replaced with

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

in which *cold-morning* has no priority over the other two disjuncts. In this case, in addition to S, $\{\mathit{turn\text{-}key},\ \neg\mathit{start},\ \mathit{cold\text{-}morning}\}$ also becomes a p-answer set by definition.

In PLP priority relations are defined over elements from $\mathcal{L}^*_P$, but they are used to express preference between more general form of knowledge. When a priority relation exists between conjunctions (resp. disjunctions) of elements $e_1, \ldots, e_m \preceq e'_1, \ldots, e'_n$ (resp. $e_1 \mid \ldots \mid e_m \preceq e'_1 \mid \ldots \mid e'_n$), it is expressed in PLP by introducing rules $e_0 \leftarrow e_1, \ldots, e_m$ and $e'_0 \leftarrow e'_1, \ldots, e'_n$ (resp. $e_0 \leftarrow e_i$ $(i = 1, \ldots, m)$ and $e'_0 \leftarrow e'_j$ $(j = 1, \ldots, n)$) with the priority relation $e_0 \preceq e'_0$. Also, when a priority relation $e_1 \preceq e_2$ holds under some condition Γ, we can express the priority by introducing rules $e'_1 \leftarrow e_1, \Gamma$ and $e'_2 \leftarrow e_2, \Gamma$ together with the priority relation $e'_1 \preceq e'_2$. For example, consider the situation that a reasoner prefers coffee to tea when sleepy. This preference can be expressed as $\mathit{tea} \preceq \mathit{coffee} \leftarrow \mathit{sleepy}$. The conditional preference relation can be represented in PLP as

$$\begin{aligned} &\mathit{tea}' \leftarrow \mathit{tea},\ \mathit{sleepy},\\ &\mathit{coffee}' \leftarrow \mathit{coffee},\ \mathit{sleepy}, \end{aligned}$$

with the relation $\mathit{tea}' \preceq \mathit{coffee}'$. First suppose that he drinks either tea or coffee (but not both):

$$\begin{aligned} &\mathit{tea} \leftarrow \mathit{not\ coffee},\\ &\mathit{coffee} \leftarrow \mathit{not\ tea}. \end{aligned}$$

Then, if *sleepy* holds, $\{\mathit{sleepy},\ \mathit{coffee},\ \mathit{coffee}'\}$ becomes the p-answer set. Next if it turns that no coffee is available ($\neg\mathit{coffee} \leftarrow$), then $\{\mathit{sleepy},\ \mathit{tea},\ \mathit{tea}',\ \neg\mathit{coffee}\}$ becomes the p-answer set. Thus, PLP chooses appropriate answer sets according to varying situations. Note that in the first case the conditional preference could be expressed with rules like

$$\begin{aligned} &\mathit{tea} \leftarrow \mathit{not\ coffee},\ \mathit{not\ sleepy},\\ &\mathit{coffee} \leftarrow \mathit{not\ tea}. \end{aligned}$$

However, these rules cannot be applied to the second case.

Preference between conflicting default rules is also specified in PLP, which will be discussed later.

3 Commonsense Reasoning in PLP

In this section, we present applications of PLP to various forms of commonsense reasoning.

3.1 Abduction

Abduction is inference to explanations and here we consider *abductive logic programming*. We first review the framework of abductive logic programming in terms of GEDP. The following definition is due to [IS94].

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

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

The augmented disjunctive rules in (2) mean that "an abducible A is assumed or not". Let Π be an ALP and O a ground atom which represents an *observation*. Then, a set $E \subseteq \mathcal{A}$ is an *explanation* of O in Π if there is an 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 an answer set of $\Pi \cup \{\leftarrow \mathit{not}\, O\,\}$ such that $E = S \cap \mathcal{A}$.

Example 3.1 Let Π be the program

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

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

In abductive reasoning, selecting *best explanations* from many candidate explanations is an important problem. In particular, *minimal* explanations are usually preferred as simplest assumptions to explain a given observation. An explanation E is *minimal* if no $E' \subset E$ is an explanation. Then, minimal abduction is expressed in PLP as follows.

Definition 3.1 Given an ALP Π, *minimal abduction* is defined as a PLP (Π, Φ) where $\Phi = \{\, A \preceq \mathit{not}\, A \,:\, A \in \mathcal{A}\,\}$.

In the definition, the priority relation $A \preceq \mathit{not}\, A$ 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 [IS94] Let Π be an ALP and O an observation. Then, O has a minimal explanation E in Π iff $\Pi \cup \{\leftarrow \mathit{not}\, O\,\}$ has an $\mathcal{A}$-minimal answer set S such that $E = S \cap \mathcal{A}$. □

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

Proof: By Lemma 3.1, it is enough to show that S is an $\mathcal{A}$-minimal answer set of $\Pi \cup \{ \leftarrow \mathit{not}\, O \}$ iff S is a p-answer set of $(\Pi \cup \{ \leftarrow \mathit{not}\, O \}, \Phi)$.
Put $\Pi' = \Pi \cup \{ \leftarrow \mathit{not}\, O \}$ and let S be an answer set of Π'. Then, S is an $\mathcal{A}$-minimal answer set of Π' iff
for any answer set T of Π', $\exists A \in (S \setminus T) \cap \mathcal{A}$ implies $\exists A' \in (T \setminus S) \cap \mathcal{A}$
iff for any 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 answer set T of Π', $S \preceq T$ implies $T \preceq S$
iff S is a p-answer set of (Π', Φ). $\Box$

Example 3.2 In Example 3.1, let $\Phi = \{ \mathit{sprinkler\text{-}on} \preceq \mathit{not\ sprinkler\text{-}on}, \mathit{rained} \preceq \mathit{not\ rained} \}$. Then, $(\Pi \cup \{ \leftarrow \mathit{not}\, O \}, \Phi)$ 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 $\{ \mathit{sprinkler\text{-}on} \}$.

An abductive logic program generally has multiple minimal explanations. Further preference over minimal explanations is also specified in PLP.

Theorem 3.3 Let (Π', Φ) be a PLP representing minimal abduction where $\Pi' = \Pi \cup \{ \leftarrow \mathit{not}\, O \}$ for an observation O. Also, let Ψ be a set of priority relations of the form $\mathit{not}\, A_i \preceq \mathit{not}\, A_j$ where A_i and A_j are abducibles, and $\Phi' = \Phi \cup \Psi$. Then, S is a p-answer set of (Π', Φ') iff S is a p-answer set of (Π', Φ) such that for any p-answer set T of (Π', Φ), $\mathit{not}\, A_i \in S \setminus T$ and $\mathit{not}\, A_j \in T \setminus S$ imply $\mathit{not}\, A_i \not\prec \mathit{not}\, A_j$ in Φ'.

Proof: If S is a p-answer set of (Π', Φ), there is some A_i such that $\mathit{not}\, A_i \in S \setminus T$ and $A_i \in T \setminus S$ with $A_i \preceq \mathit{not}\, A_i$ for any p-answer set T of (Π', Φ). Since there is no $\mathit{not}\, A_j \in T \setminus S$ such that $\mathit{not}\, A_i \prec \mathit{not}\, A_j$, $T \preceq S$ holds by definition. Hence, S is a p-answer set of (Π', Φ').

On the other hand, if S is a p-answer set of (Π', Φ'), then S is a p-answer set of (Π', Φ). In this case, since $S \preceq T$ implies $T \preceq S$ for any p-answer set T of (Π', Φ), $\mathit{not}\, A_i \preceq \mathit{not}\, A_j$ implies $\mathit{not}\, A_j \preceq \mathit{not}\, A_i$ for any $\mathit{not}\, A_i \in S \setminus T$ and $\mathit{not}\, A_j \in T \setminus S$. $\Box$

In the theorem, the additional relation $\mathit{not}\, A_i \preceq \mathit{not}\, A_j$ says that an abducible A_j is less likely to happen than A_i. Introducing this relation to Φ, the p-answer set S of (Π', Φ) satisfying $\mathit{not}\, A_j$ with the highest priority becomes the p-answer set of (Π', Φ'). In this case, minimal explanations containing no A_j are selected as the best explanations.

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*. In this case, $(\Pi \cup \{ \leftarrow \mathit{not}\, O \}, \Phi \cup \{ \mathit{not\ rained} \preceq \mathit{not\ sprinkler\text{-}on} \})$ has the unique p-answer set which implies the preferred minimal explanation $\{ \mathit{rained} \}$.

3.2 Default Reasoning

Default reasoning is a basic mechanism in commonsense reasoning. Poole [Poo88] proposed a simple framework for default reasoning, which is reformulated by Inoue [Ino94] 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. Given $K = (P, \Delta)$, its *extension base* is defined as an answer set of $P \cup D$ where D is a maximal subset of the ground instances of Δ such that $P \cup D$ is consistent.

Example 3.3 Let $K = (P, \Delta)$ be a knowledge system such that

$$\begin{aligned}
P: \quad & \neg flies(x) \leftarrow penguin(x), \\
& bird(x) \leftarrow penguin(x), \\
& bird(polly) \leftarrow, \\
& penguin(tweety) \leftarrow, \\
\Delta: \quad & flies(x) \leftarrow bird(x).
\end{aligned}$$

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

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

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

Definition 3.2 Given a knowledge system $K = (P, \Delta)$, its PLP expression (Π, Φ) is defined as follows.

1. Any rule in P is included in Π.
2. Any rule $Head \leftarrow Body$ in Δ is transformed to the rules

$$Head \leftarrow \delta(\mathbf{x}),\ Body, \tag{3}$$

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

   in Π, where $\mathbf{x}$ represents variables appearing in the rule, and $\delta(\mathbf{x})$ is uniquely associated with each rule from Δ.
3. For any $\delta(\mathbf{x})$ introduced above, the relation $not\, \delta(\mathbf{x}) \preceq \delta(\mathbf{x})$ is in Φ.

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

Let $\mathcal{D}$ be the set of every atom $\delta(\mathbf{x})$ in Π.[2] 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 we have the following results.

Lemma 3.4 [Ino94] Let $K = (P, \Delta)$ be a knowledge system and Π the transformed program as above. Then, S is a $\mathcal{D}$-maximal answer set of Π iff $S \cap \mathcal{L}_K$ is an extension base of K. □

Theorem 3.5 Let $K = (P, \Delta)$ be a knowledge system and (Π, Φ) its PLP expression. Then, S is a p-answer set of (Π, Φ) iff $S \cap \mathcal{L}_K$ is an extension base of K.

Proof: Using Lemma 3.4, the proof is similar to that of Theorem 3.2. □

Example 3.4 The knowledge system in Example 3.3 is expressed in PLP as

$$\begin{aligned}\Pi: \quad & \neg \mathit{flies}(x) \leftarrow \mathit{penguin}(x),\\ & \mathit{bird}(x) \leftarrow \mathit{penguin}(x),\\ & \mathit{bird}(\mathit{polly}) \leftarrow,\\ & \mathit{penguin}(\mathit{tweety}) \leftarrow,\\ & \mathit{flies}(x) \leftarrow \delta(x),\, \mathit{bird}(x),\\ & \delta(x) \mid \mathit{not}\, \delta(x) \leftarrow,\end{aligned}$$

with $\Phi = \{\, \mathit{not}\, \delta(x) \preceq \delta(x) \,\}$. Then (Π, Φ) has the unique p-answer set $\{\, \mathit{bird}(p),\, \mathit{penguin}(t),\, \mathit{bird}(t),\, \mathit{flies}(p),\, \neg \mathit{flies}(t),\, \delta(p) \,\}$, which corresponds to the extension base S.

Next let us consider a default theory containing conflicting default rules.

Example 3.5 Let K' be the same knowledge system as K in Example 3.3 except that the first rule $\neg \mathit{flies}(x) \leftarrow \mathit{penguin}(x)$ in P is placed in Δ as a default rule. Then, in addition to S, K' has the alternative extension base $S' = \{\, \mathit{bird}(p),\, \mathit{penguin}(t),\, \mathit{bird}(t),\, \mathit{flies}(p),\, \mathit{flies}(t) \,\}$.

In the above situation, the knowledge system has two alternative extension bases, but S is preferred to S' as the more specific extension. To select the right extension, the default rule for *penguin* must be preferred to that for *bird*. PLP is also used to specify such priorities between default rules.

Theorem 3.6 Let (Π, Φ) be a PLP representing a knowledge system. Also, let Ψ be a set of priority relations of the form $\delta_i \preceq \delta_j$ where δ_i and δ_j are atoms in Φ, and $\Phi' = \Phi \cup \Psi$. Then, S is a p-answer set of (Π, Φ') iff S is a p-answer set of (Π, Φ) such that for any p-answer set T of (Π, Φ), $\delta_i \in S \setminus T$ and $\delta_j \in T \setminus S$ imply $\delta_i \not\preceq \delta_j$ in Φ'.

[2] Here, $\delta(\mathbf{x})$ is identified with its ground instances.

Proof: Similar to the proof of Theorem 3.3. □

The above theorem presents that when any default preference $\delta_i \preceq \delta_j$ is introduced to Φ, the p-answer set S of (Π, Φ) containing the default δ_j with the highest priority is selected as the p-answer set of (Π, Φ'). Thus, priorities between default rules are specified in PLP and the extensions generated by the most preferred defaults are obtained as p-answer sets.

Back to the above example, the knowledge system K' is expressed in PLP as (Π', Φ') where Π' contains rules

$$\neg flies(x) \leftarrow \delta'(x),\ penguin(x),$$
$$\delta'(x) \mid not\, \delta'(x) \leftarrow,$$

instead of the first rule in Π, and $\Phi' = \Phi \cup \{\, not\, \delta'(x) \preceq \delta'(x),\ \delta(x) \preceq \delta'(x)\,\}$ where a higher priority is given to the more specific default $\delta'(x)$. In this case, (Π', Φ') has the unique p-answer set $\{\, bird(p),\ penguin(t),\ bird(t),\ flies(p),$ $\neg flies(t),\ \delta(p),\ \delta'(p),\ \delta'(t)\,\}$, which corresponds to S.
Thus, PLP can realize a prioritized default reasoning system like [Bre94].

3.3 Prioritized Circumscription

Prioritized circumscription is a representative formalism of prioritized nonmonotonic reasoning in AI. This section considers realizing prioritized circumscription in PLP. We first review the framework of prioritized circumscription [Lif86].

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* $Circ(T; P_1 > \ldots > P_k; Z)$ minimizes extensions of P_i with a priority higher than those of P_j $(i < j)$ with Z *varied.* The set of all predicates other than P and Z from T are *fixed* and denoted as Q. 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 $[\![M_1]\!](P_j) \subset [\![M_2]\!](P_j)$. A model M of T is a model of $Circ(T; P_1 > \ldots > P_k; Z)$ iff there is no model N of T such that $N \ll M$.

To realize prioritized circumscription in the context of logic programming, we assume a first-order theory T as a set of *clauses* $C:\ A_1 \vee \ldots \vee A_l \vee \neg B_1 \vee \ldots \vee \neg B_m$ 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. Now the PLP expression of prioritized circumscription is defined as follows.

Definition 3.3 Given a prioritized circumscription $Circ(T; P_1 > \ldots > P_k; Z)$, its PLP expression (Π, Φ) is defined as follows.

1. For any clause C in T, Π has the rule

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

2. For any fixed or variable predicate λ in T, Π has the rule
$$\lambda(\mathbf{x}) \mid \mathit{not}\,\lambda(\mathbf{x}) \leftarrow .$$
3. Priority relations are given as
$$\begin{aligned}\Phi &= \{\, p_i(\mathbf{x}) \preceq \mathit{not}\,p_i(\mathbf{x}) \;:\; p_i \in P_i \; (i = 1, \ldots, k) \,\} \\ &\cup \{\, \mathit{not}\,p_{i+1}(\mathbf{x}) \preceq \mathit{not}\,p_i(\mathbf{y}) \;:\; p_i \in P_i \; (i = 1, \ldots, k-1) \,\} \\ &\cup \{\, q(\mathbf{x}) \preceq \mathit{not}\,q(\mathbf{x}),\;\; \mathit{not}\,q(\mathbf{x}) \preceq q(\mathbf{x}) \;:\; q \in Q \,\}.\end{aligned}$$

In the transformation, minimizing extensions of predicates from P is expressed by the relation $p_i(\mathbf{x}) \preceq \mathit{not}\,p_i(\mathbf{x})$ in Φ. Also, the predicate hierarchy $P_1 > \ldots > P_k$ is expressed in Φ as $\mathit{not}\,p_{i+1}(\mathbf{x}) \preceq \mathit{not}\,p_i(\mathbf{y})$, which means that extensions from p_i is minimized at a higher priority than those from p_{i+1}. 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 [SI95]. In this case, extensions of variable predicates can be varied, while those of fixed predicates are not affected by the preference over minimized predicates. Such a situation is expressed by the symmetric priority relations $q(\mathbf{x}) \preceq \mathit{not}\,q(\mathbf{x})$ and $\mathit{not}\,q(\mathbf{x}) \preceq q(\mathbf{x})$ in Φ.

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

Lemma 3.7 Let $\mathit{Circ}(T; P; Z)$ be a (parallel) circumscription and (Π, Φ) its PLP expression. Then, M is an Herbrand model of $\mathit{Circ}(T; P; Z)$ iff M is a p-answer set of (Π, Φ).

Proof: M is a model of $\mathit{Circ}(T; P; Z)$ 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 $\exists p \in P\ (\mathit{not}\,p \in N \setminus M \wedge p \in M \setminus N) \wedge \neg\exists p' \in P\ (\mathit{not}\,p' \in M \setminus N \wedge p' \in N \setminus M)$
iff $M \preceq N$ and $N \not\preceq M$. Hence, for any M and N such that $M \cap Q = N \cap Q$, $N \ll M$ iff $M \preceq N$ and $N \not\preceq M$, thereby $N \not\ll M$ iff $M \preceq N$ implies $N \preceq M$. 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 $M \preceq N$ holds by $q \preceq \mathit{not}\,q$. In this case, $N \preceq M$ also holds by $\mathit{not}\,q \preceq q$. Thus, $M \preceq N$ iff $N \preceq M$. 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 $M \preceq N$ implies $N \preceq M$ $(*)$. Let $M \cap (Q \cup Z) = \Gamma$. If M is an Herbrand model of $\mathit{Circ}(T; P; Z)$, then M is a minimal model of $T \cup \Gamma$. In this case, M is a minimal model of $T \cup \{\lambda \mid \mathit{not}\,\lambda \leftarrow \in \Pi\}^M$ 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 relation $(*)$ holds for answer sets M and N of Π. Hence, M is an Herbrand model of $\mathit{Circ}(T; P; Z)$ iff M is a p-answer set of (Π, Φ). $\square$

Theorem 3.8 Let $Circ(T; P_1 > \ldots > P_k; Z)$ be a prioritized circumscription and (Π, Φ) its PLP expression. Then, M is an Herbrand model of $Circ(T; P_1 > \ldots > P_k; Z)$ iff M is a p-answer set of (Π, Φ).

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

$$\exists i\ \ \exists p_i \in P_i\,(p_i \in M \setminus N) \,\wedge\, \neg\exists p'_i \in P_i\,(p'_i \in N \setminus M)$$
$$\wedge\, \forall p_j \in P_j\,(j < i)\,(p_j \in M \Leftrightarrow p_j \in N).\ \ (*)$$

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

$$\exists i\ \ \exists p_i \in P_i\,(p_i \in M \setminus N) \,\wedge\, \exists k\,(i < k)\,\exists p_k \in P_k\,(p_k \in N \setminus M)$$
$$\wedge \neg\exists p'_i \in P_i\,(p'_i \in N \setminus M) \,\wedge\, \forall p_j \in P_j\,(j < i)\,(p_j \in M \Leftrightarrow p_j \in N)$$

iff $\exists i\ \ \exists p_i \in P_i\,(not\, p_i \in N \setminus M) \,\wedge\, \exists k\,(i < k)\,\exists p_k \in P_k\,(not\, p_k \in M \setminus N)$
$$\wedge \neg\exists p'_i \in P_i\,(not\, p'_i \in M \setminus N) \,\wedge\, \neg\exists p_j \in P_j\,(j < i)\,(not\, p_j \in M \setminus N)$$
$$\wedge \neg\exists p'_j \in P_j\,(j < i)\,(not\, p'_j \in N \setminus M)$$

iff $\exists i\ \ \exists p_i \in P_i\,(not\, p_i \in N \setminus M) \,\wedge\, \exists k\,(i < k)\,\exists p_k \in P_k\,(not\, p_k \in M \setminus N)$
$$\wedge \neg\exists p_j \in P_j\,(j \leq i)\,(not\, p_j \in M \setminus N) \,\wedge\, \neg\exists p'_j \in P_j\,(j < i)\,(not\, p'_j \in N \setminus M).\ (\dagger)$$

Here,

$$\exists i\ \ \exists p_i \in P_i\,(not\, p_i \in N \setminus M) \,\wedge\, \exists k\,(i < k)\,\exists p_k \in P_k\,(not\, p_k \in M \setminus N)$$
$$\wedge \neg\exists p_j \in P_j\,(j \leq i)\,(not\, p_j \in M \setminus N)\ \ \ (\ddagger)$$

implies $M \preceq N$ and $N \not\preceq M$, so $(\dagger)$ implies $M \preceq N$ and $N \not\preceq M$. Conversely, $M \preceq N$ and $N \not\preceq M$ 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., $\exists p_j \in P_j\,(j \leq i')\,(not\, p_j \in M \setminus N)$, then $M \preceq N$ implies $N \preceq M$, which contradicts the assumption. Hence, $\neg\exists p_j \in P_j\,(j \leq i')\,(not\, p_j \in M \setminus N)$ also holds. Since i' is minimal satisfying $\exists p_{i'} \in P_{i'}\,(not\, p_{i'} \in N \setminus M)$, $\neg\exists p'_j \in P_j\,(j < i')\,(not\, p'_j \in N \setminus M)$. Then, by putting $i = i'$, $(\ddagger)$ implies $(\dagger)$, thereby $M \preceq N$ and $N \not\preceq M$ imply $(\dagger)$. Hence, for any M and N such that $M \cap Q = N \cap Q$, $N \ll M$ iff $M \preceq N$ and $N \not\preceq M$, thereby $N \not\ll M$ iff $M \preceq N$ implies $N \preceq M$. On the other hand, for any M and N such that $M \cap Q \neq N \cap Q$, $M \preceq N$ iff $N \preceq M$ by the same argument as in Lemma 3.7. Therefore, for any M and N, $N \not\ll M$ iff $M \preceq N$ implies $N \preceq M$. Since Herbrand models M and N of $Circ(T; P_1, \ldots, P_k; Z)$ are (p-)answer sets of Π by Lemma 3.7, M is an Herbrand model of $Circ(T; P_1 > \ldots > P_k; Z)$ iff M is a p-answer set of (Π, Φ). □

Example 3.6 Let T be a first-order theory such that

$$flies(x) \vee ab_2(x) \vee \neg bird(x),$$
$$\neg flies(x) \vee ab_1(x) \vee \neg penguin(x),$$
$$bird(x) \vee \neg penguin(x),$$

$$bird(polly),$$
$$penguin(tweety),$$

where $P_1 = \{\, ab_1 \,\}$ and $P_2 = \{\, ab_2 \,\}$ with $P_1 > P_2$, and $Z = \{\, flies \,\}$ and $Q = \{\, penguin, bird \,\}$. In this case, its PLP expression becomes

$$\begin{aligned} \Pi: \quad & flies(x) \mid ab_2(x) \leftarrow bird(x), \\ & ab_1(x) \leftarrow penguin(x),\ flies(x), \\ & bird(x) \leftarrow penguin(x), \\ & bird(polly) \leftarrow, \\ & penguin(tweety) \leftarrow, \\ & \lambda(x) \mid not\ \lambda(x) \leftarrow, \end{aligned}$$

where $\lambda = flies$, $bird$, $penguin$, and $\Phi = \{\, ab_1(x) \preceq not\, ab_1(x),\ ab_2(x) \preceq not\, ab_2(x),\ not\, ab_2(x) \preceq not\, ab_1(y),\ bird(x) \preceq not\ bird(x),\ not\ bird(x) \preceq bird(x),\ penguin(x) \preceq not\ penguin(x),\ not\ penguin(x) \preceq penguin(x) \,\}$. Then, (Π, Φ) has two p-answer sets $\{\, bird(p),\ flies(p),\ penguin(t),\ bird(t),\ ab_2(t) \,\}$ and $\{\, bird(p),\ penguin(p),\ ab_2(p),\ penguin(t),\ bird(t),\ ab_2(t) \,\}$, which correspond to the Herbrand models of $Circ(T; P_1 > P_2; Z)$.

Gelfond and Lifschitz [GL88] 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 predicates. By contrast, the PLP characterization presented above has no such restriction.

In the absence of fixed and variable predicates, prioritized circumscription is also expressed by the *perfect models* [Prz88] of a stratified disjunctive program. When a stratified disjunctive program P contains no integrity constraints, the perfect models of P coincide with the answer sets of P, thereby also coincide with the p-answer sets of $(P, \emptyset)$.[3] Perfect models of a stratified disjunctive program are also characterized by PLP in the following manner. Given a stratified disjunctive program Π, define the *positive disjunctive program* $Pos(\Pi)$ which is obtained from Π by shifting $not\, A$ in the body of any rule to A in the head.

Corollary 3.9 Let Π be a stratified disjunctive program including no integrity constraints, with the stratification $P_1 > \ldots > P_k$. Then, M is a perfect model of Π iff M is a p-answer set of $(pos(\Pi), \Phi)$ where $\Phi = \{\, not\, p_{i+1}(\mathbf{x}) \preceq not\, p_i(\mathbf{y}) : p_i \in P_i\ (i = 1, \ldots, k-1)\}$.

Proof: In the absence of fixed and variable predicates, any p-answer set of $Pos(\Pi)$ is minimal wrt extensions of P. Hence, we can drop priority

[3] The perfect models do not coincide with the answer sets in a stratified program with integrity constraints in general. For example, consider the program $P = \{\, b \leftarrow not\, a,\ \leftarrow\ b \,\}$ with the stratification $\{a\} > \{b\}$. Then P has the perfect model $\{a\}$, while no answer set exists.

relations $p_i(\mathbf{x}) \preceq not\, p_i(\mathbf{x})$ $(i = 1, \ldots, k)$ from Φ. Then the result follows from [Prz88, Theorem 5] and Theorem 3.8. □

The above corollary presents that any stratification is equivalently specified in PLP without using negation as failure in a program.[4]

4 Discussion

Priorities in logic programming are conventionally expressed in terms of stratified negation. In contrast to stratification, priorities in PLP are specified separately from the program. Hence, different programmers can specify different priorities in the same program without changing the program itself, while any change in a program does not affect the priority relations. 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 priority relations in PLP (Corollary 3.9), but the converse transformation, i.e., representing arbitrary priority relations Φ in a single stratification, is not possible in general. 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 relations are freely specified on any subset of $\mathcal{L}_P^*$. Thirdly, PLP can express priorities between not only atoms but also literals and negation-as-failure formulas in GEDP.

Priorities in PLP are used to specify preference between default rules. Dimopoulos and Kakas [DK95] provide a method of specifying priorities over conflicting default rules in extended logic programs. Brewka [Bre96] extends the well-founded semantics to handling prioritized reasoning in extended logic programs. Comparing these work with ours, PLP is defined for a wider class of programs and used for not only default reasoning but other prioritized commonsense reasoning. Brewka also introduces a method of encoding preference 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.

The expressive power of PLP is analyzed from the computational aspect as follows. In Theorem 3.2 it was shown that minimal explanations can be expressed in terms of p-answer sets of PLP. On the other hand, Eiter *et al.* [EGL95] argue that reasoning tasks for minimal explanations in (propositional) abductive disjunctive programs are in general at the third level Σ_3^P/Π_3^P of the polynomial hierarchy. Since it is known that the computational complexity of the standard answer sets lies at the second level of the hierarchy, these results suggest that the p-answer sets are more expressive than the standard answer sets unless the polynomial hierarchy collapses.

[4] [DK95] presents a different method of replacing negation as failure with preference over rules.

5 Summary

This paper introduced a novel framework of representing priorities in logic programming. It was shown that PLP can encode preference over incomplete knowledge and reduce non-determinism in logic programming. Moreover, various forms of commonsense reasoning such as (prioritized) minimal abduction, (prioritized) default reasoning, and prioritized circumscription were realized in terms of PLP. Thus, PLP increases the expressive power of logic programming and exploits new applications in knowledge representation. Note that we introduced PLP under the answer set semantics, while the framework is also applicable to any semantics of logic programming. In this paper we were concerned with the semantic aspects of PLP and future work involves the procedural issue of PLP.

References

[Bre94] Brewka, G., Reasoning About Priorities in Default Logic, *Proc. AAAI-94*, pp. 940–945, MIT Press.

[Bre96] Brewka, G., Well-Founded Semantics for Extended Logic Programs with Dynamic Preferences, *J. AI Research* 4:19–36, 1996.

[DK95] Dimopoulos, Y. and Kakas, A. C., Logic Programming without Negation as Failure, *Proc. ILPS'95*, pp. 369–383, MIT Press.

[EGL95] Eiter, T., Gottlob, G. and Leone, N., Complexity Results for Abductive Logic Programming, *LPNMR'95, LNAI* 928, pp. 1–14, Springer.

[GL88] Gelfond, M. and Lifschitz, V., Compiling Circumscriptive Theories into Logic Programs, *Proc. AAAI-88*, pp. 455–459, MIT Press.

[GL91] Gelfond, M. and Lifschitz, V., Classical Negation in Logic Programs and Disjunctive Databases, *New Generation Computing* 9:365–385, 1991.

[IS94] Inoue, K. and Sakama, C., On Positive Occurrences of Negation as Failure. *Proc. KR'94*, pp. 293–304, Morgan Kaufmann.

[Ino94] Inoue, K., Hypothetical Reasoning in Logic Programs, *J. Logic Programming* 18:191–227, 1994.

[Lif86] Lifschitz, V., On the Satisfiability of Circumscription, *Artificial Intelligence* 28:17–27, 1986.

[LW92] Lifschitz, V. and Woo, T. Y. C., Answer Sets in General Nonmonotonic Reasoning, *Proc. KR'92*, pp. 603–614, Morgan Kaufmann.

[Poo88] Poole, D., A Logical Framework for Default Reasoning, *Artificial Intelligence* 36:27–47, 1988.

[Prz88] Przymusinski, T. C., On the Declarative Semantics of Deductive Databases and Logic Programs, in: *Foundations of Deductive Databases and Logic Programming*, pp. 193–216, Morgan Kaufmann, 1988.

[SI95] Sakama, C. and Inoue, K., Embedding Circumscriptive Theories in General Disjunctive Programs, *LPNMR'95, LNAI* 928, 344–357, Springer.