Two Results for Prioritized Logic Programming

YAN ZHANG
School of Computing and Information Technology
University of Western Sydney
Locked Bag 1797, Penrith South DC
NSW 1797, Australia
E-mail: yan@cit.uws.edu.au

Abstract

Prioritized default reasoning has illustrated its rich expressiveness and flexibility in knowledge representation and reasoning. However, many important aspects of prioritized default reasoning have yet to be thoroughly explored. In this paper, we investigate two properties of prioritized logic programs in the context of answer set semantics. Specifically, we reveal a close relationship between mutual defeasibility and uniqueness of the answer set for a prioritized logic program. We then explore how the splitting technique for extended logic programs can be extended to prioritized logic programs. We prove splitting theorems that can be used to simplify the evaluation of a prioritized logic program under certain conditions.

1 Introduction

Prioritized default reasoning has illustrated its rich expressiveness and flexibility in knowledge representation, reasoning about action and rule updates (Brewka, 1996; Grosof, 1997; Zhang & Foo, 1997). Recently, different approaches and formulations for prioritized default reasoning based on logic programming and default theories have been proposed (Brewka & Eiter, 1999; Delgrande *et al.*, 2000; Grosof, 1997; Wang *et al.*, 2000). However, most of these proposals mainly focus on the semantic development, while other important properties are usually not thoroughly explored.

In this paper, we investigate two specific properties of prioritized logic programs in the context of answer set semantics. First, we reveal a close relationship between mutual defeasibility and uniqueness of the answer set for a prioritized logic program. Mutual defeasibility can be viewed as a way of characterizing rules in a logic program, where two rules in the program are mutually defeasible if triggering one rule may cause a defeat of the other, and *vice versa*. It is quite easy to observe that a logic program does not contain a pair of mutually defeasible rules if this program is locally stratified. However, the converse does not hold. We then provide a sufficient condition to ensure the uniqueness of the answer set for a prioritized logic program. We show that our characteristic condition is weaker than the traditional local stratification for general logic programs (Apt & Bol, 1994).

Second, we investigate the splitting technique for prioritized logic programs. It is well known that Lifschitz and Turner's Splitting Set Theorem (Lifschitz & Turner,

1994) for extended logic programs may significantly simplify the computation of answer sets of an extended logic program. The basic idea of splitting technique is that under certain conditions, an extended logic program can be split into several "smaller components" such that the computation of the answer set of the original program is reduced to the computation of the answer set of these smaller components. We show that this splitting technique is also suitable for computing answer sets of prioritized logic programs. Furthermore, our splitting theorems for prioritized logic programs also provide a generalization of Lifschitz and Turner's result.

The paper is organized as follows. Section 2 develops the syntax and semantics of prioritized logic programs. In our formulation, a prioritized logic program is defined to be an extended logic program associating with a strict partial ordering on rules in the program. An answer set semantics for prioritized logic programs is then defined. Several basic properties of prioritized logic programs are also studied in this section. By introducing the concept of mutual defeasibility, section 3 proves a sufficient condition to characterize the unique answer set for a prioritized logic program. Section 4 then extends the splitting technique for extended logic programs to prioritized logic programs. Finally, section 5 concludes the paper with some remarks.

2 Prioritized Logic Programs

To specify prioritized logic programs (PLPs), we first introduce the extended logic program and its answer set semantics developed by Gelfond and Lifschitz (Gelfond & Lifschitz, 1991). A language $\mathcal{L}$ of extended logic programs is determined by its object constants, function constants and predicate constants. *Terms* are built as in the corresponding first order language; *atoms* have the form $P(t_1, \cdots, t_n)$, where t_i $(1 \leq i \leq n)$ is a term and P is a predicate constant of arity n; a *literal* is either an atom $P(t_1, \cdots, t_n)$ or a negative atom $\neg P(t_1, \cdots, t_n)$. A *rule* is an expression of the form:

$$L_0 \leftarrow L_1, \cdots, L_m, notL_{m+1}, \cdots, notL_n, \tag{1}$$

where each L_i $(0 \leq i \leq n)$ is a literal. L_0 is called the *head* of the rule, while $\{L_1, \cdots, L_m,\ notL_{m+1}, \cdots,\ notL_n\}$ is called the *body* of the rule. Obviously, the body of a rule could be empty. We also allow the head of a rule to be empty. In this case, the rule with an empty head is called *constraint*. A term, atom, literal, or rule is *ground* if no variable occurs in it. An *extended logic program* Π is a collection of rules. Π is *ground* if each rule in Π is ground.

Let r be a ground rule of the form (1), we use $pos(r)$ to denote the set of literals in the body of r without negation as failure $\{L_1, \cdots, L_m\}$, and $neg(r)$ the set of literals in the body of r with negation as failure $\{L_{m+1}, \cdots, L_n\}$. We specify $body(r)$ to be $pos(r) \cup neg(r)$. We also use $head(r)$ to denote the head of r: $\{L_0\}$. Then we use $lit(r)$ to denote $head(r) \cup body(r)$. By extending these notations, we use $pos(\Pi)$, $neg(\Pi)$, $body(\Pi)$, $head(\Pi)$, and $lit(\Pi)$ to denote the unions of corresponding components of all rules in the ground program Π, e.g. $body(\Pi) = \bigcup_{r\in\Pi} body(r)$. If

Π is a non-ground program, then notions $pos(\Pi)$, $neg(\Pi)$, $body(\Pi)$, $head(\Pi)$, and $lit(\Pi)$ are defined based on the ground instantiation (see below definition) of Π.

To evaluate an extended logic program, Gelfond and Lifschitz proposed an answer set semantics for extended logic programs. Let Π be a ground extended logic program not containing *not* and *Lit* the set of all ground literals in the language of Π. An *answer set* of Π is the smallest subset S of *Lit* such that (i) for any rule $L_0 \leftarrow L_1, \cdots, L_m$ from Π, if $L_1, \cdots, L_m \in S$, then $L_0 \in S$; and (ii) if S contains a pair of complementary literals, then $S = Lit$. Now let Π be a ground arbitrary extended logic program. For any subset S of Lit, let Π^S be the logic program obtained from Π by deleting (i) each rule that has a formula *not* L in its body with $L \in S$, and (ii) all formulas of the form *not* L in the bodies of the remaining rules[1]. We define that S is an *answer set* of Π iff S is an answer set of Π^S.

For a non-ground extended logic program Π, we usually view a rule in Π containing variables to be the set of all ground instances of this rule formed from the set of ground literals in the language. The collection of all these ground rules forms the *ground instantiation* Π' of Π. Then a set of ground literals is an answer set of Π if and only if it is an answer set of Π'. It is easy to see that an extended logic program may have one, more than one, or no answer set at all.

A *prioritized logic program* (PLP) $\mathcal{P}$ is a triple $(\Pi, \mathcal{N}, <)$, where Π is an extended logic program, $\mathcal{N}$ is a naming function mapping each rule in Π to a name, and $<$ is a strict partial ordering on names. The partial ordering $<$ in $\mathcal{P}$ plays an essential role in the evaluation of $\mathcal{P}$. We also use $\mathcal{P}(<)$ to denote the set of $<$-relations of $\mathcal{P}$. Intuitively $<$ represents a preference of applying rules during the evaluation of the program. In particular, if $\mathcal{N}(r) < \mathcal{N}(r')$ holds in $\mathcal{P}$, rule r would be preferred to apply over rule r' during the evaluation of $\mathcal{P}$ (i.e. rule r is more preferred than rule r'). Consider the following classical example represented in our formalism:

$\mathcal{P}_1 = (\Pi, \mathcal{N}, <)$:
$N_1 : Fly(x) \leftarrow Bird(x),\ not\ \neg Fly(x),$
$N_2 : \neg Fly(x) \leftarrow Penguin(x),\ not\ Fly(x),$
$N_3 : Bird(Tweety) \leftarrow,$
$N_4 : Penguin(Tweety) \leftarrow,$
$N_2 < N_1.$

Obviously, rules N_1 and N_2 conflict with each other as their heads are complementary literals, and applying N_1 will defeat N_2 and *vice versa*. However, as $N_2 < N_1$, we would expect that rule N_2 is preferred to apply first and then defeat rule N_1 so that the desired solution $\neg Fly(Tweety)$ can be derived.

Definition 1
Let Π be a ground extended logic program and r a ground rule of the form (1) (r does not necessarily belong to Π). Rule r is *defeated* by Π iff Π has an answer set and for any answer set S of Π, there exists some $L_i \in S$, where $m + 1 \leq i \leq n$.

Now our idea of evaluating a PLP is as follows. Let $\mathcal{P} = (\Pi, \mathcal{N}, <)$. If there are two rules r and r' in Π and $\mathcal{N}(r) < \mathcal{N}(r')$, r' will be ignored in the evaluation of

[1] We also call Π^S the Gelfond-Lifschitz transformation of Π in terms of S.

$\mathcal{P}$, *only if* keeping r in Π and deleting r' from Π will result in a defeat of r'. By eliminating all such potential rules from Π, $\mathcal{P}$ is eventually reduced to an extended logic program in which the partial ordering $<$ has been removed. Our evaluation for $\mathcal{P}$ is then based on this *reduced* extended logic program.

Similarly to the case of extended logic programs, the evaluation of a PLP will be based on its ground form. We say that a PLP $\mathcal{P}' = (\Pi', \mathcal{N}', <')$ is the *ground instantiation* of $\mathcal{P} = (\Pi, \mathcal{N}, <)$ if (1) Π' is the ground instantiation of Π; and (2) $\mathcal{N}'(r_1') <' \mathcal{N}'(r_2') \in \mathcal{P}'(<')$ if and only if there exist rules r_1 and r_2 in Π such that r_1' and r_2' are ground instances of r_1 and r_2 respectively and $\mathcal{N}(r_1) < \mathcal{N}(r_2) \in \mathcal{P}(<)$. Under this definition, however, we require a restriction on a PLP since not every PLP's ground instantiation presents a consistent information with respect to the original PLP. Consider a PLP as follows:

$N_1 : P(f(x)) \leftarrow notP(x)$,
$N_2 : P(f(f(x))) \leftarrow notP(f(x))$,
$N_2 < N_1$.

If the only constant in the language is 0, then the set of ground instances of N_1 and N_2 includes rules like:

$N_1' : P(f(0)) \leftarrow notP(0)$,
$N_2' : P(f(f(0))) \leftarrow notP(f(0))$,
$N_3' : P(f(f(f(0)))) \leftarrow notP(f(f(0)))$,
$\cdots$,

It is easy to see that N_2' can be viewed as an instance for both N_1 and N_2. Therefore, the ordering $<'$ among rules $N_1', N_2', N_3', \cdots$ is no longer a partial ordering because of $N_2' <' N_2'$. Obviously, we need to exclude this kind of programs in our context. On the other hand, we also want to avoid a situation like $\cdots <' N_3' <' N_2' <' N_1'$ in the ground prioritized logic program because this $<'$ indicates that there is no most preferred rule in the program.

Given a PLP $\mathcal{P} = (\Pi, \mathcal{N}, <)$. We say that $\mathcal{P}$ is *well formed* if there is no rule r' that is an instance of two different rules r_1 and r_2 in Π and $\mathcal{N}(r_1) < \mathcal{N}(r_2) \in \mathcal{P}(<)$. Then it is not difficult to observe that the following fact holds.

Fact: If a PLP $\mathcal{P} = (\Pi, \mathcal{N}, <)$ is well formed, then in its ground instantiation $\mathcal{P}' = (\Pi', \mathcal{N}', <')$, $<'$ is a partial ordering and every non-empty subset of Π' has a least element with respect to $<'$.

Due to the above fact, in the rest of this paper, we will only consider well formed PLP programs in our discussions, and consequently, the evaluation for an arbitrary PLP $\mathcal{P} = (\Pi, \mathcal{N}, <)$ will be based on its ground instantiation $\mathcal{P}' = (\Pi', \mathcal{N}', <')$. Therefore, in our context a ground prioritized (or extended) logic program may contain infinite number of rules. In this case, we will assume that this ground program is the ground instantiation of some program that only contains finite number of rules. In the rest of the paper, whenever there is no confusion, we will only consider ground prioritized (extended) logic programs without explicit declaration.

Definition 2

(Zhang & Foo, 1997) Let $\mathcal{P} = (\Pi, \mathcal{N}, <)$ be a prioritized logic program. $\mathcal{P}^<$ is a *reduct* of $\mathcal{P}$ with respect to $<$ if and only if there exists a sequence of sets Π_i $(i = 0, 1, \cdots)$ such that:

1. $\Pi_0 = \Pi$;
2. $\Pi_i = \Pi_{i-1} - \{r_1, r_2, \cdots \mid$ (a) there exists $r \in \Pi_{i-1}$ such that for every j $(j = 1, 2, \cdots)$, $\mathcal{N}(r) < \mathcal{N}(r_j) \in \mathcal{P}(<)$ and $r_1, r_2, \cdots$ are defeated by $\Pi_{i-1} - \{r_1, r_2, \cdots\}$, and (b) there are no rules $r', r'', \cdots \in \Pi_{i-1}$ such that $N(r_j) < N(r')$, $N(r_j) < N(r''), \cdots$ for some j $(j = 1, 2, \cdots)$ and $r', r'', \cdots$ are defeated by $\Pi_{i-1} - \{r', r'', \cdots\}\}$;
3. $\mathcal{P}^< = \bigcap_{i=0}^{\infty} \Pi_i$.

In Definition 2, $\mathcal{P}^<$ is an extended logic program obtained from Π by eliminating some rules from Π. In particular, if $\mathcal{N}(r) < \mathcal{N}(r_1)$, $\mathcal{N}(r) < \mathcal{N}(r_2)$, $\cdots$, and $\Pi_{i-1} - \{r_1, r_2, \cdots\}$ defeats $\{r_1, r_2, \cdots\}$, then rules $r_1, r_2, \cdots$ will be eliminated from Π_{i-1} if no *less preferred rule* can be eliminated (i.e. conditions (a) and (b)). This procedure is continued until a fixed point is reached. It should be noted that condition (b) in the above definition is necessary because without it some unintuitive results may be derived. For instance, consider $\mathcal{P}_1$ again, if we add additional preference $N_3 < N_2$ in $\mathcal{P}_1$, then using a modified version of Definition 2 without condition (b),

$\{Fly(Tweety) \leftarrow Bird(Tweety), not \neg Fly(Tweety),$
$\quad Bird(Tweety) \leftarrow,$
$\quad Penguin(Tweety) \leftarrow\}$

is a reduct of $\mathcal{P}_1$, from which we will conclude that Tweety can fly.

Definition 3
(Zhang & Foo, 1997) Let $\mathcal{P} = (\Pi, \mathcal{N}, <)$ be a PLP and Lit the set of all ground literals in the language of $\mathcal{P}$. For any subset S of Lit, S is an *answer set* of $\mathcal{P}$ iff S is an answer set for some reduct $\mathcal{P}^<$ of $\mathcal{P}$.

Using Definitions 2 and 3, it is easy to conclude that $\mathcal{P}_1$ has a unique reduct as follows:

$\mathcal{P}_1^< = \{\neg Fly(Tweety) \leftarrow Penguin(Tweety), not\ Fly(Tweety),$
$\qquad Bird(Tweety) \leftarrow,$
$\qquad Penguin(Tweety) \leftarrow\},$

from which we obtain the following answer set of $\mathcal{P}_1$:

$S = \{Bird(Tweety), Penguin(Tweety), \neg Fly(Tweety)\}.$

Now we consider another program $\mathcal{P}_2$:

$N_1 : A \leftarrow,$
$N_2 : B \leftarrow not\ C,$
$N_3 : D \leftarrow,$
$N_4 : C \leftarrow not\ B,$
$N_1 < N_2, N_3 < N_4.$

According to Definition 2, it is easy to see that $\mathcal{P}_2$ has two reducts:

$\{A \leftarrow, \quad D \leftarrow, \quad C \leftarrow not\ B\}$, and
$\{A \leftarrow, \quad B \leftarrow not\ C, \quad D \leftarrow\}$.

From Definition 3, it follows that $\mathcal{P}_2$ has two answer sets: $\{A, C, D\}$ and $\{A, B, D\}$.

To see whether our PLP semantics gives intuitive results in prioritized default reasoning, we further consider a program $\mathcal{P'}_2$ - a variation of program $\mathcal{P}_2$, as follows.

$N_1 : A \leftarrow,$
$N_2 : B \leftarrow not\ C,$
$N_3 : C \leftarrow not\ B,$
$N_1 < N_2.$

It is easy to see that $\mathcal{P'}_2$ has one answer set $\{A, C\}$. People may think that this result is not quite intuitive because rule N_2 is defeated in the evaluation although there is no preference between N_2 and N_3. To explain why $\{A, C\}$ is a *reasonable* answer set of $\mathcal{P'}_2$, we should review the concept of defeatness in our formulation (Definition 1). In a PLP, when we specify one rule is less preferred than the other, for instance, $N_1 < N_2$ (N_2 is less preferred than N_1), it does not mean that N_2 should be defeated by N_1 iff conflict occurs between them. Instead, it just means that N_2 has a lower priority than N_1 to be taken into account in the evaluation of the *whole* program while other rules should be retained in the evaluation process if no preference is specified between N_2 and them. This intuition is captured by the notion of defeatness in Definition 1 and Definition 2.

Back to the above example, although there is no direct conflict between N_1 and N_2 *and* no preference is specified between N_2 and N_3 (where conflict exists between them), N_2 indeed has a lower priority than N_1 to be applied in the evaluation of $\mathcal{P'}_2$, which causes N_2 to be defeated.

Now we illustrate several basic properties of prioritized logic programs. As we mentioned earlier, when we evaluate a PLP, a rule including variables is viewed as the set of its all ground instances. Therefore, we are actually dealing with *ground* prioritized logic programs that may consist of infinite collection of rules. We first introduce some useful notations. Let Π be an extended logic program. We use $\mathcal{A}(\Pi)$ to denote the class of all answer sets of Π. Suppose $\mathcal{P} = (\Pi, \mathcal{N}, <)$ is a PLP. From Definition 2, we can see that a reduct $\mathcal{P}^<$ of $\mathcal{P}$ is generated from a sequence of extended logic programs: $\Pi = \Pi_0, \Pi_1, \Pi_2, \cdots$. We use $\{\Pi_i\}$ $(i = 0, 1, 2, \cdots)$ to denote this sequence and call it a *reduct chain* of $\mathcal{P}$.

Proposition 1
Let $\mathcal{P} = (\Pi, \mathcal{N}, <)$ be a PLP and $\{\Pi_i\}$ $(i = 0, 1, 2, \cdots)$ its reduct chain. Suppose Π has an answer set. Then for any i and j where $i < j$, $\mathcal{A}(\Pi_j) \subseteq \mathcal{A}(\Pi_i)$.

Proof
Let $\{\Pi_i\}$ $(i = 0, 1, 2, \cdots)$ be a reduct chain of $\mathcal{P}$. Suppose S_j is an answer set of Π_j for some $j > 0$. To prove the result, it is sufficient to show that S_j is also an answer set of Π_{j-1}. According to Definition 2, Π_j is obtained by eliminating some rules from Π_{j-1} where all these eliminated rules are defeated by Π_j. So we can express:

$$\Pi_j = \Pi_{j-1} - \{r_1, r_2, \cdots\}.$$

Since $r_1, r_2, \cdots$ are defeated by Π_j, we can write rules $r_1, r_2, \cdots$ to the following forms:

$r_1 : L_1 \leftarrow \cdots, not\ L'_1, \cdots,$
$r_2 : L_2 \leftarrow \cdots, not\ L'_2, \cdots,$
$\cdots,$

where $L'_1, L'_2, \cdots \in S_j$. Now consider Gelfond-Lifschitz transformation of Π_j in terms of S_j. It is clear that during the transformation, each rule in Π_j including *not* L'_1, *not* L'_2, $\cdots$ in its body will be deleted. From here it follows that adding any rule with *not* L'_1 *not* L'_2, $\cdots$ in its body will not play any role in the evaluation of the answer set of the program. So we add rules $r_1, r_2, \cdots$ into Π_j, This makes Π_{j-1}. Then we have $\Pi_j^{S_j} = \Pi_{j-1}^{S_j}$. So S_j is also an answer set of Π_{j-1}. $\Box$

Proposition 1 shows an important property of the reduct chain of $\mathcal{P}$: each Π_i is consistent with Π_{i-1} but becomes more *specific* than Π_{i-1} in the sense that all answer sets of Π_i are answer sets of Π_{i-1} but some answer sets of Π_{i-1} are filtered out if they conflict with the preference partial ordering $<$.

Example 1
Consider a PLP $\mathcal{P}_3 = (\Pi, \mathcal{N}, <)$:

$N_1 : A \leftarrow not\ B,$
$N_2 : B \leftarrow not\ A,$
$N_3 : C \leftarrow not\ B,\ not\ D,$
$N_4 : D \leftarrow not\ C,$
$N_1 < N_2, N_3 < N_4.$

From Definition 2, we can see that $\mathcal{P}_3$ has a reduct chain $\{\Pi_i\}$ $(i = 0, 1, 2)$:

Π_0:
$A \leftarrow not\ B,$
$B \leftarrow not\ A,$
$C \leftarrow not\ B,\ not\ D,$
$D \leftarrow not\ C,$
Π_1:
$A \leftarrow not\ B,$
$C \leftarrow not\ B,\ not\ D,$
$D \leftarrow not\ C,$
Π_2:
$A \leftarrow not\ B,$
$C \leftarrow not\ B,\ not\ D.$

It is easy to verify that Π_0 has three answer sets $\{A, C\}$, $\{B, D\}$ and $\{A, D\}$, Π_1 has two answer sets $\{A, C\}$ and $\{A, D\}$, and Π_2 has a unique answer set which is also the answer set of $\mathcal{P}_3$: $\{A, C\}$.

The following theorem shows the answer set relationship between a PLP and its corresponding extended logic programs.

Theorem 1
Let $\mathcal{P} = (\Pi, \mathcal{N}, <)$ be a PLP and S a subset of *Lit*. Then the following are equivalent:

1. S is an answer set of $\mathcal{P}$.
2. S is an answer set of each Π_i for some reduct chain $\{\Pi_i\}$ $(i = 0, 1, 2, \cdots)$ of $\mathcal{P}$.

Proof
$(1 \Rightarrow 2)$ Let $\mathcal{P}^<$ be a reduct of $\mathcal{P}$ obtained from a reduct chain $\{\Pi_i\}$ $(i = 0, 1, 2, \cdots)$ of $\mathcal{P}$. By applying Theorem 3 in section 3, it is easy to show that any reduct chain of $\mathcal{P}$ is finite. Therefore, there exists some k such that $\{\Pi_i\}$ $(i = 0, 1, 2, \cdots, k)$ is the reduct chain. This follows that $\mathcal{P}^< = \Pi_k \subseteq \Pi_i$ $(i = 1, \cdots, k)$. So from Proposition 1, an answer set of $\mathcal{P}^<$ is also an answer set of Π_i $(i = 1, \cdots, k)$.

$(2 \Rightarrow 1)$ Given a reduct chain $\{\Pi_i\}$ $(i = 0, 1, 2, \cdots)$ of $\mathcal{P}$. From the above, since $\{\Pi_i\}$ $(i = 0, 1, 2, \cdots)$ is finite, we can assume that $\{\Pi_i\}$ $(i = 0, 1, 2, \cdots, k)$ is the reduct chain. As $\Pi_j \subseteq \Pi_i$ if $j > i$, it follows that $\bigcap_{i=0}^{k} \Pi_i = \Pi_k$. So the fact that S is an answer set of Π_k implies that S is also an answer set of $\mathcal{P}$. □

Corollary 1
If a PLP $\mathcal{P} = (\Pi, \mathcal{N}, <)$ has an answer set S, then S is also an answer set of Π.

Proof
From Theorem 1, it shows that if $\mathcal{P}$ has an answer set S, then S is also an answer set of each Π_i for $\mathcal{P}$'s a reduct chain $\{\Pi_i\}$ $(i = 0, 1, 2, \cdots)$, where $\Pi_0 = \Pi$. So S is also an answer set of Π. □

The following theorem presents a sufficient and necessary condition for the answer set existence of a PLP.

Theorem 2
Let $\mathcal{P} = (\Pi, \mathcal{N}, <)$ be a PLP. $\mathcal{P}$ has an answer set if and only if Π has an answer set.

Proof
According to Corollary 1, we only need to prove that if Π has an answer set, then $\mathcal{P}$ also has an answer set. Suppose Π has an answer set and $\{\Pi_i\}$ $(i = 0, 1, \cdots)$ is a reduct chain of $\mathcal{P}$. From the construction of $\{\Pi_i\}$ (Definition 2), it is easy to see that every Π_i $(i = 0, 1, \cdots)$ must have an answer set. On the other hand, as we have mentioned in the proof of Theorem 1, $\mathcal{P}$'s reduct chain is actually finite: $\{\Pi_i\}$ $(i = 0, 1, \cdots, k)$. That follows $\mathcal{P}^< = \Pi_k$. Since Π_k has an answer set, it concludes $\mathcal{P}$ has an answer set as well. □

Proposition 2
Suppose a PLP $\mathcal{P}$ has a unique reduct. If $\mathcal{P}$ has a consistent answer set, then $\mathcal{P}$'s every answer set is also consistent.

Proof
The fact that $\mathcal{P}$ has a consistent answer set implies that $\mathcal{P}$'s reduct $\mathcal{P}^<$ (note $\mathcal{P}^<$ is an extended logic program) has a consistent answer set. Then from the result showed in section 2 of (Lifschitz & Turner, 1994) (i.e. if an extended logic program has a consistent answer set, then its every answer set is also consistent), it follows that $\mathcal{P}^<$'s every answer set is also consistent. □

3 Mutual Defeasibility and Unique Answer Set

In this section, we try to provide a sufficient condition to characterize the uniqueness of the answer set for a prioritized logic program. The following definition extends the concept of local stratification for general logic programs (Apt & Bol, 1994; Choelwinski, 1994; Das, 1992) to extended logic programs.

Definition 4
Let Π be an extended logic program and Lit be the set of all ground literals of Π.

1. A *local stratification* for Π is a function *stratum* from Lit to the countable ordinals.
2. Given a local stratification *stratum*, we extend it to ground literals with negation as failure by setting $stratum(\text{not } L) = stratum(L)+1$, where L is a ground literal.
3. A rule $L_0 \leftarrow L_1, \cdots, L_m, \text{not } L_{m+1}, \cdots, \text{not } L_n$ in Π is *locally stratified* with respect to *stratum* if
   $stratum(L_0) \geq stratum(L_i)$, where $1 \leq i \leq m$, and
   $stratum(L_0) > stratum(notL_j)$, where $m+1 \leq j \leq n$.
4. Π is called *locally stratified* with respect to *stratum* if all of its rules are locally stratified. Π is called *locally stratified* if it is locally stratified with respect to some local stratification.

It is easy to see that the corresponding extended logic program of $\mathcal{P}_1$ (see section 2) is not locally stratified. In general, we have the following sufficient condition to ensure the uniqueness of the answer set for an extended logic program.

Proposition 3
Let Π be an extended logic program. If Π is locally stratified, then Π has a unique answer set[2].

Now we define the concept of mutual defeasibility which plays a key role in investigating a sufficient condition for the unique answer set of a PLP.

Definition 5
Let Π be an extended logic program and r_p and r_q be two rules in Π. We define a set $\mathcal{D}(r_p)$ of literals with respect to r_p as follows:

$\mathcal{D}_0 = \{head(r_p)\}$;
$\mathcal{D}_i = \mathcal{D}_{i-1} \cup \{head(r) \mid head(r') \in pos(r)$ where $r \in \Pi$ and r' are those rules such that $head(r') \in \mathcal{D}_{i-1}\}$;
$\mathcal{D}(r_p) = \bigcup_{i=1}^{\infty} \mathcal{D}_i$.

We say that r_q is *defeasible through* r_p in Π if and only if $neg(r_q) \cap \mathcal{D}(r_p) \neq \emptyset$. r_p and r_q are called *mutually defeasible* in Π if r_q is defeasible through r_p and r_p is defeasible through r_q in Π.

[2] Recall that if Π has an inconsistent answer set, we will denote it as Lit. This proposition is a direct generalization of the result for general logic programs as described in (Gelfond & Lifschitz, 1988).

Intuitively, if r_q is defeasible through r_p in Π, then there exists a sequence of rules $r_1, r_2, \cdots, r_l, \cdots$ such that $head(r_p)$ occurs in $pos(r_1)$, $head(r_i)$ occurs in $pos(r_{i+1})$ for all $i = 1, \cdots$, and for some k, $head(r_k)$ occurs in $neg(r_q)$. Under this condition, it is clear that by triggering rule r_p in Π, it is possible to defeat rule r_q if rules $r_1, \cdots, r_k$ are triggered as well. As a special case that $\mathcal{D}(r_p) = \{head(r_p)\}$, r_q is defeasible through r_p iff $head(r_p) \in neg(r_q)$. The following proposition simply describes the relationship between local stratification and mutual defeasibility.

Proposition 4
Let Π be an extended logic program. If Π is locally stratified, then there does not exist mutually defeasible pair of rules in Π.

The above result is easy to prove from the corresponding result for general logic programs showed in (Gelfond & Lifschitz, 1988) based on Gelfond and Lifschitz's translation from an extended logic program to a general logic program (Gelfond & Lifschitz, 1991). It is observed that for a PLP $\mathcal{P} = (\Pi, \mathcal{N}, <)$, if Π is locally stratified, then $\mathcal{P}$ will have a unique answer set. In other words, Π's local stratification implies that $\mathcal{P}$ has a unique answer set. However, this condition seems too strong because many prioritized logic programs will still have unique answer sets although their corresponding extended logic programs are not locally stratified. For instance, program $\mathcal{P}_1$ presented in section 2 has a unique answer set but its corresponding extended logic program is not locally stratified. But one fact is clear: the uniqueness of reduct for a PLP is necessary to guarantee this PLP to have a unique answer set.

The above observation suggests that we should first investigate the condition under which a prioritized logic program has a unique reduct. Then by applying Proposition 3 to the unique reduct of the PLP, we obtain the unique answer set condition for this PLP.

Definition 6
Let $\mathcal{P} = (\Pi, \mathcal{N}, <)$ be a PLP. A *<-partition* of Π in $\mathcal{P}$ is a finite collection $\{\Pi_1, \cdots, \Pi_k\}$, where $\Pi = \Pi_1 \cup \cdots \cup \Pi_k$ and Π_i and Π_j are disjoint for any $i \neq j$, such that

1. $\mathcal{N}(r) < \mathcal{N}(r') \in \mathcal{P}(<)$ implies that there exist some i and j $(1 \leq i < j)$ such that $r' \in \Pi_j$ and $r \in \Pi_i$;
2. for each rule $r' \in \Pi_j$ $(j > 1)$, there exists some rule $r \in \Pi_i$ $(1 \leq i < j)$ such that $\mathcal{N}(r) < \mathcal{N}(r') \in \mathcal{P}(<)$.

Example 2
Consider a PLP $\mathcal{P}_4 = (\Pi, \mathcal{N}, <)$:

$N_1 : A \leftarrow not\ B, not\ C,$
$N_2 : B \leftarrow not\ \neg C,$
$N_3 : C \leftarrow not\ A, not\ \neg C,$
$N_4 : \neg C \leftarrow not\ C,$
$N_1 < N_2, N_2 < N_4, N_3 < N_4.$

It is easy to verify that a <-partition of Π in $\mathcal{P}_4$ is $\{\Pi_1, \Pi_2, \Pi_3\}$, where

Π_1:
  $N_1 : A \leftarrow not\ B,\ not\ C,$
  $N_3 : C \leftarrow not\ A,\ not\ \neg C,$
Π_2:
  $N_2 : B \leftarrow not\ \neg C,$
Π_3:
  $N_4 : \neg C \leftarrow not\ C.$

In fact, this program has a unique answer set $\{B, C\}$.

Theorem 3
Every prioritized logic program has a $<$-partition.

Proof
For a given PLP $\mathcal{P} = (\Pi, \mathcal{N}, <)$, we construct a series of subsets of Π as follows:
$\Pi_1 = \{r \mid$ there does not exist a rule $r' \in \Pi$ such that $\mathcal{N}(r') < \mathcal{N}(r)\}$;
$\Pi_i = \{r \mid$ for all rules such that $\mathcal{N}(r') < \mathcal{N}(r)$, $r' \in \bigcup_{j=1}^{i-1} \Pi_j\}$.
We prove that $\{\Pi_1, \Pi_2, \cdots\}$ is a $<$-partition of $\mathcal{P}$. First, it is easy to see that Π_i and Π_j are disjoint. Now we show that this partition satisfies conditions 1 and 2 described in Definition 6. Let $\mathcal{N}(r) < \mathcal{N}(r') \in \mathcal{P}(<)$. If there does not exist any rule $r'' \in \Pi$ such that $\mathcal{N}(r'') < \mathcal{N}(r)$, then $r \in \Pi_1$. Otherwise, there exists some i $(i > 1)$ such that $r \in \Pi_i$ and for all rules satisfying $\mathcal{N}(r'') < \mathcal{N}(r)$ $r'' \in \Pi_1 \cup \cdots \cup \Pi_{i-1}$. Let $r' \in \Pi_j$. Since $\mathcal{N}(r) < \mathcal{N}(r')$, it follows that $1 < j$. From the construction of Π_j, we also conclude $r \in \Pi_1 \cup \cdots \cup \Pi_{j-1}$. Since $r' \in \Pi_j$, it follows $i \leq j-1$. That is, $i < j$. Condition 2 directly follows from the construction of the partition described above.

Now we show that $\{\Pi_1, \Pi_2, \cdots\}$ must be a finite set. First, if Π is finite, it is clear $\{\Pi_1, \Pi_2, \cdots\}$ must be a finite set. If Π contains infinite rules, then according to our assumption presented in section 2, $\mathcal{P}$ must be the ground instantiation of some program, say $\mathcal{P}^* = (\Pi^*, \mathcal{N}^*, <^*)$. Then we can use the same way to define a $<$-partition for $\mathcal{P}^*$. Since Π^* is finite, the partition of $\mathcal{P}^*$ must be also finite: $\{\Pi_1^*, \Pi_2^*, \cdots, \Pi_k^*\}$. As $\mathcal{P}^*$ is well formed, it implies that for each i, Π_i is the ground instantiation of Π_i^*. So $\{\Pi_1, \Pi_2, \cdots\}$ is finite. □

Theorem 4
(**Unique Answer Set Theorem**) Let $\mathcal{P} = (\Pi, \mathcal{N} <)$ be a PLP and $\{\Pi_1, \cdots, \Pi_k\}$ be a $<$-partition of Π in $\mathcal{P}$. $\mathcal{P}$ has a unique reduct if there does not exist two rules r_p and r_q in Π_i and Π_j $(i, j > 1)$ respectively such that r_p and r_q are mutually defeasible in Π. $\mathcal{P}$ has a unique answer set if $\mathcal{P}$ has a unique locally stratified reduct.

Proof
According to Proposition 3, it is sufficient to only prove the first part of this theorem: $\mathcal{P}$ has a unique reduct if there does not exist two rules r_p and r_q in Π_i and Π_j $(i, j > 1)$ respectively such that r_p and r_q are mutually defeasible in Π.

We assume that $\mathcal{P}$ has two different reducts, say $\mathcal{P}^{<(1)}$ and $\mathcal{P}^{<(2)}$. This follows that there exist at least two different rules r_p and r_q such that (1) $r_p \in \Pi_i$ and $r_q \in \Pi_j$, where $i, j > 1$; (2) $r_q \in \mathcal{P}^{<(1)}$, $r_q \notin \mathcal{P}^{<(2)}$, and $r_p \notin \mathcal{P}^{<(1)}$; and (3)

$r_p \in \mathcal{P}^{<(2)}$, $r_p \notin \mathcal{P}^{<(1)}$, and $r_q \notin \mathcal{P}^{<(2)}$. According to Definition 2, $\mathcal{P}^{<(1)}$ and $\mathcal{P}^{<(2)}$ are generated from two reduct chains $\{\Pi_0^{(1)}, \Pi_1^{(1)}, \cdots\}$ and $\{\Pi_0^{(2)}, \Pi_1^{(2)}, \cdots\}$ respectively.

Without loss of generality, we may assume that for all $0 \leq i < k$, $\Pi_i^{(1)} = \Pi_i^{(2)}$, and

$$\Pi_k^{(1)} = \Pi_{k-1}^{(1)} - \{r_1, \cdots, r_l, r_p, \cdots\},$$
$$\Pi_k^{(2)} = \Pi_{k-1}^{(2)} - \{r_1, \cdots, r_l, r_q, \cdots\},$$

where we set $\Pi_{k-1} = \Pi_{k-1}^{(1)} = \Pi_{k-1}^{(2)}$ and the only difference between $\Pi_k^{(1)}$ and $\Pi_k^{(2)}$ is due to rules r_p and r_q. Let r_p and r_q have the following forms:

$$r_p : L_p \leftarrow \cdots, \mathit{not}\ L'_p, \cdots,$$
$$r_q : L_q \leftarrow \cdots, \mathit{not}\ L'_q, \cdots.$$

Comparing $\Pi_k^{(1)}$ and $\Pi_k^{(2)}$, it is clear that the only difference between these two programs is about rules r_p and r_q. Since $\Pi_k^{(1)}$ defeats r_p and $\Pi_k^{(2)}$ defeats r_q, it follows that $L'_q \in S_k^{(1)}$ and $L'_p \in S_k^{(2)}$, where $S_k^{(1)}$ and $S_k^{(2)}$ are answer sets of $\Pi_k^{(1)}$ and $\Pi_k^{(2)}$ respectively. Then there must exist some rule in $\Pi_k^{(1)}$ of the form:

$$r^{(1)} : L'_p \leftarrow \cdots,$$

and some rule in $\Pi_k^{(2)}$ of the form:

$$r^{(2)} : L'_q \leftarrow \cdots.$$

Furthermore, since $\Pi_k^{(1)} - \{r_p, r_q\}$ does not defeat rule r_p and $\Pi_k^{(2)} - \{r_p, r_q\}$ does not defeat rule r_q (otherwise $\Pi_k^{(1)} = \Pi_k^{(2)}$), it is observed that rule r_q triggers rule $r^{(1)}$ in $\Pi_k^{(1)}$ that defeats r_p, and rule r_p triggers rule $r^{(2)}$ in $\Pi_k^{(2)}$ that defeats r_q. This follows that r_p and r_q are mutually defeasible in Π. □

Note that according to Proposition 4, the condition for $\mathcal{P} = (\Pi, \mathcal{N}, <)$ to have a unique answer set stated in Theorem 4 is weaker than the local stratification requirement for Π to have a unique answer set as showed by Proposition 3.

Example 3
Consider PLP $\mathcal{P}_5 = (\Pi, \mathcal{N}, <)$ as follows:

$$N_1 : A \leftarrow \mathit{not}\ B, \mathit{not}\ C, \mathit{not}\ D,$$
$$N_2 : B \leftarrow \mathit{not}\ A, \mathit{not}\ D,$$
$$N_3 : C \leftarrow \mathit{not}\ A, \mathit{not}\ D,$$
$$N_4 : D \leftarrow \mathit{not}\ A,$$
$$N_1 < N_2 < N_3.$$

Clearly, a $<$-partition of Π is as follows:

Π_1:
$N_1 : A \leftarrow \mathit{not}\ B, \mathit{not}\ C\ \mathit{not}\ D,$
$N_4 : D \leftarrow \mathit{not}\ A,$
Π_2:
$N_2 : B \leftarrow \mathit{not}\ A,$
Π_3:
$N_3 : C \leftarrow \mathit{not}\ A.$

Although Π is not locally stratified, from Theorem 4, $\mathcal{P}_5$ should have a unique reduct $\{N_1\}$ since N_2 and N_3 are not mutually defeasible. This also concludes that $\mathcal{P}_5$ has a unique answer set $\{A\}$.

4 Splitting Prioritized Logic Programs

It has been observed that deciding whether a prioritized logic program has an answer set is NP-complete (Zhang, 2001). That means, in practice it is unlikely to implement a polynomial algorithm to compute the answer set of a prioritized logic program. Hence, finding suitable strategy to simplify such computation is an important issue. Similarly to the case of extended logic programs (Lifschitz & Turner, 1994), we will show that under proper conditions, a PLP $\mathcal{P}$ can be split into several smaller components $\mathcal{P}_1, \cdots, \mathcal{P}_k$ such that the evaluation of $\mathcal{P}$'s answer sets can be based on the evaluation of the answer sets of $\mathcal{P}_1, \cdots, \mathcal{P}_k$. To describe our idea, we first consider the case of splitting a PLP into two parts.

Example 4
Consider the following PLP $\mathcal{P}_6 = (\Pi, \mathcal{N}, <)$:

$N_1 : A \leftarrow not\neg A, notD,$
$N_2 : D \leftarrow not\neg D,$
$N_3 : \neg D \leftarrow notD,$
$N_4 : B \leftarrow notC,$
$N_5 : C \leftarrow notB,$
$N_6 : A \leftarrow C, \neg D,$
$N_1 < N_4, N_6 < N_2.$

Clearly, this PLP has a unique reduct $\{N_1, N_3, N_5, N_6\}$, which gives a unique answer set $\{A, C, \neg D\}$.

We observe that Π actually can be split into two segments $\Pi_1 = \{N_1, N_2, N_3\}$ and $\Pi_2 = \{N_4, N_5, N_6\}$ such that $head(\Pi_2) \cap body(\Pi_1) = \emptyset$. Now we try to reduce the computation of $\mathcal{P}_6$'s answer sets to the computation of two smaller PLPs' answer sets. Firstly, we define a PLP $\mathcal{P}_6^1 = (\Pi_1^*, \mathcal{N}, <)$ by setting $\Pi_1^* = \Pi_1 \cup \{N_0 : First \leftarrow\}$ and $\mathcal{P}_6^1(<) = \{N_0 < N_2\}$. The role of rule N_0 is to introduce a $<$-relation $N_0 < N_2$ to replace the original $<$-relation $N_6 < N_2$ in $\mathcal{P}_6$ that is missed from $\mathcal{P}_6^1$ by eliminating N_6 from Π_1. The unique answer set of $\mathcal{P}_6^1$ is $S_1 = \{First, A, \neg D\}$. Since $head(\Pi_2) \cap body(\Pi_1) = \emptyset$, it is easy to see that in each of $\mathcal{P}_6$'s answer sets, any literals derived by using rules in Π_2 will not trigger or defeat any rules in Π_1. This implies that every literal in $\mathcal{P}_6^1$'s answer set (except *First*) will also occur in an answer set of the original $\mathcal{P}_6$. Therefore, we can define another PLP $\mathcal{P}_6^2 = (\Pi_2^*, \mathcal{N}, <)$ by setting $\Pi_2^* = \{N_4, N_5\} \cup \{N_0 : First \leftarrow,\ N_6' : A \leftarrow C\}$ and $N_0 < N_4$. Here N_6 in Π_2 is replaced by N_6' under $\mathcal{P}_6^1$'s answer set S_1 providing $\neg D$ to be true. Then $\mathcal{P}_6^2$ also has a unique answer set $S_2 = \{First, A, C\}$. Finally, the unique answer set of $\mathcal{P}_6$ is obtained by $S_1 \cup S_2 - \{First\} = \{A, C, \neg D\}$.

From the above example, we see that if in a PLP $\mathcal{P} = (\Pi, \mathcal{N}, <)$, Π can be split into two parts Π_1 and Π_2 such that $head(\Pi_2) \cap body(\Pi_1) = \emptyset$, then it is possible to also split $\mathcal{P}$ into two smaller PLPs $\mathcal{P}^1$ and $\mathcal{P}^2$ such that $\mathcal{P}$'s every answer set can be computed from $\mathcal{P}^1$ and $\mathcal{P}^2$'s. To formalize our result, we first introduce some useful notions. Given a PLP $\mathcal{P} = (\Pi, \mathcal{N}, <)$, we define a map $first^< : \Pi \longrightarrow \Pi$, such that $first^<(r) = r'$ if there exists some r' such that $\mathcal{N}(r') < \mathcal{N}(r) \in \mathcal{P}(<)$ and there does not exist another $r'' \in \Pi$ satisfying $\mathcal{N}(r'') < \mathcal{N}(r') \in \mathcal{P}(<)$; otherwise $first^<(r) = undefined$. Intuitively, $first^<(r)$ gives the rule which is most preferred than r in $\mathcal{P}$. As $<$ is a strict partial ordering, there may be more than one most preferred rules than r.

To define a split of a PLP, we first introduce the concept of e-reduct of an extended logic program. Let Π be an extended logic program and X be a set of ground literals. The *e-reduct* of Π with respect to set X is an extended logic program, denoted as $e(\Pi, X)$, obtained from Π by deleting (1) each rule in Π that has a formula $notL$ in its body with $L \in X$, and (2) all formulas of the form L in the bodies of the remaining rules with $L \in X$. Consider an example that $X = \{C\}$ and Π consists of two rules:

$A \leftarrow B, notC,$
$B \leftarrow C, notA.$

Then $e(\Pi, X) = \{B \leftarrow notA\}$. Intuitively, the e-reduct of Π with respect to X can be viewed as a simplified program of Π given the fact that every literal in X is true. For a rule $r \in e(\Pi, X)$, we use $original(r)$ to denote r's original form in Π. In the above example, it is easy to see that $original(B \leftarrow notA)$ is $B \leftarrow C, notA$. Now a split of a PLP can be formally defined as follows.

Definition 7
Let $\mathcal{P} = (\Pi, \mathcal{N}, <)$. We say that $(\mathcal{P}^1, \mathcal{P}^2)$ is a *split* of $\mathcal{P}$, if there exist two disjoint subsets Π_1 and Π_2 of Π, where $\Pi = \Pi_1 \cup \Pi_2$, such that

1. $head(\Pi_2) \cap body(\Pi_1) = \emptyset$,
2. $\mathcal{P}^1 = (\Pi_1 \cup \{N_0 : First \leftarrow\}, \mathcal{N}, <)$[3], where for any $r, r' \in \Pi_1$, $\mathcal{N}(r) < \mathcal{N}(r') \in \mathcal{P}(<)$ implies $\mathcal{N}(r) < \mathcal{N}(r') \in \mathcal{P}^1(<)$, and if there exists some $r'' \in \Pi_2$ and $first^<(r) = r''$, then $N_0 < \mathcal{N}(r) \in \mathcal{P}^1(<)$;
3. $\mathcal{P}^2 = (e(\Pi_2, S_1) \cup \{N_0 : First \leftarrow\}, \mathcal{N}, <)$, where S_1 is an answer set of $\mathcal{P}^1$, for any $r, r' \in e(\Pi_2, S_1)$, $\mathcal{N}(original(r)) < \mathcal{N}(original(r')) \in \mathcal{P}(<)$ implies $\mathcal{N}(r) < \mathcal{N}(r') \in \mathcal{P}^2(<)$, and if there exists some $r'' \in \Pi_1$ and $first^<(original(r)) = r''$, then $N_0 < \mathcal{N}(r) \in \mathcal{P}^2(<)$.

A split $(\mathcal{P}^1, \mathcal{P}^2)$ is called *S-dependent* if S is an answer set of $\mathcal{P}^1$ and $\mathcal{P}^2$ is formed based on S as described in condition 3 above, i.e. $\mathcal{P}^2 = (e(\Pi_2, S) \cup \{N_0 : First \leftarrow \}, \mathcal{N}, <)$.

In Example 4, it is easy to verify that $(\mathcal{P}_6^1, \mathcal{P}_6^2)$ is a split of $\mathcal{P}_6$. Now we have the major result of splitting a PLP.

Theorem 5

[3] Here we assume that *First* is a ground literal not occurring in $\mathcal{P}$.

Let $(\mathcal{P}^1, \mathcal{P}^2)$ be a S_1-dependent split of $\mathcal{P}$ as defined in Definition 7. A set of ground literals S is a consistent answer set of $\mathcal{P}$ if and only if $S = S_1 \cup S_2 - \{First\}$, where S_2 is an answer set of $\mathcal{P}^2$, and $S_1 \cup S_2$ is consistent.

Proof

We prove this theorem in two steps. Suppose Π^* is a reduct of $\mathcal{P}$. According to Definition 2, Π^* can be represented as the form $\Pi^* = \Pi_1^* \cup \Pi_2^*$, where $\Pi_1^* \subseteq \Pi_1$ and $\Pi_2^* \subseteq \Pi_2$. So every answer set of Π^* is also an answer set of $\mathcal{P}$. In the first step, we prove <u>*Result 1*</u>: a set S of ground literals is a consistent answer set of Π^* iff $S = S_1 \cup S_2$, where S_1 is an answer set of Π_1^*, S_2 is an answer set of $e(\Pi_2^*, S_1)$, and $S_1 \cup S_2$ is consistent. In the second step, we prove <u>*Result 2*</u>: $\Pi_1^* \cup \{First \leftarrow\}$ is a reduct of $\mathcal{P}^1$ and $e(\Pi_2^*, S_1) \cup \{First \leftarrow\}$ is a reduct of $\mathcal{P}^2$. Then the theorem is proved directly from these two results.

We first prove <u>*Result 1*</u>. ($\Leftarrow$) Let $S = S_1 \cup S_2$ and $\Pi^* = \Pi_1^* \cup \Pi_2^*$, where S_1 is an answer set of Π_1^* and S_2 is an answer set of $e(\Pi_2^*, S_1)$ and $S_1 \cup S_2$ is consistent. Consider the Gelfond-Lifschitz transformation of Π^* in terms of S, Π^{*S}. Π^{*S} is obtained from Π^* by deleting

(i) each rule in $\Pi_1^* \cup \Pi_2^*$ that has a formula *not* L in its body with $L \in S$; and
(ii) all formulas of the form *not* L in the bodies of the remaining rules.

Since $body(\Pi_1^*) \cap head(\Pi_2^*) = \emptyset$, during the step (i) in the above transformation, for each literal $L \in S_1$, only rules of the form $L' \leftarrow \cdots, not\ L, \cdots$ in Π_1^* or Π_2^* will be deleted. On the other hand, for each literal $L \in S_2$, only rules of the form $L' \leftarrow \cdots, not\ L, \cdots$ in Π_2^* will be deleted and no rules in Π_1^* can be deleted because $head(\Pi_2^*) \cap body(\Pi_1^*) = \emptyset$. Therefore, we can denote Π^{*S} as $\Pi_1^{*'} \cup \Pi_2^{*'}$, where $\Pi_1^{*'}$ is obtained from Π_1^* in terms of literals in S_1, and $\Pi_2^{*'}$ is obtained from Π_2^* in terms of literals in $S_1 \cup S_2$ during the above transformation procedure. Then it is easy to see that $\Pi_1^{*'} = \Pi_1^{*S_1}$. So S_1 is an answer set of $\Pi_1^{*'}$.

On the other hand, from the construction of $e(\Pi_2^*, S_1)$, it is observed that there exists the following correspondence between $\Pi_2^{*'}$ and $e(\Pi_2^*, S_1)$: for each rule

$$L_0 \leftarrow L_1, \cdots, L_k, L_{k+1}, \cdots, L_m$$

in $\Pi_2^{*'}$, there is a rule of the form

$$L_0 \leftarrow L_1, \cdots, L_k, notL_{m+1}, \cdots, notL_n$$

in $e(\Pi_2, S_1)$ such that $L_{k+1}, \cdots, L_m \in S_1$ and $L_{m+1}, \cdots, L_n \notin S_1$; on the other hand, for each rule $L_0 \leftarrow L_1, \cdots, L_k,\ not\ L_{m+1}, \cdots,\ not\ L_n$ in $e(\Pi_2^*, S_1)$, if none of $L_{m+1}, \cdots, L_n$ is in S_2, then there exists a rule $L_0 \leftarrow L_1, \cdots, L_k, L_{k+1}, \cdots, L_m$ in $\Pi_2^{*'}$ such that $L_{k+1}, \cdots, L_m \in S_1$. From this observation, it can be seen that there exists a subset Δ of S_1 such that $\Delta \cup S_2$ is an answer set of $\Pi_2^{*'}$. This follows that $S_1 \cup S_2$ is the smallest set such that for each rule $L_0 \leftarrow L_1, \cdots, L_m \in \Pi^{*S}$, $L_1, \cdots, L_m \in S$ implies $L_0 \in S$. That is, S is an answer set of Π^{*S} and also an answer set of Π^*.

($\Rightarrow$) Let $\Pi^* = \Pi_1^* \cup \Pi_2^*$ and S be a consistent answer set of Π^*. It is clear that for each literal $L \in S$, there must exist some rule of the form $L \leftarrow \cdots$ in Π. So we can write S as a form of $S_1' \cup S_2'$ such that $S_1' \subseteq head(\Pi_1^*)$ and $S_2' \subseteq head(\Pi_2^*)$. Note that $S_1' \cap S_2'$ may not be empty. Now we transfer set S_1' into S_1 by the following step: if $S_1' \cap S_2' = \emptyset$, then $S_1 = S_1'$; otherwise, let

$$\begin{aligned} S_1 = S_1' - \{L \mid\ & L \in S_1' \cap S_2', \text{ and for each rule} \\ & L \leftarrow L_1, \cdots, L_m, notL_{m+1}, \cdots, notL_n \text{ in } \Pi_1^*, \text{ there exists some} \\ & L_j\ (1 \leq j \leq m) \notin S_1' \text{ or } L_j \in S_1'(m+1 \leq j \leq n)\}. \end{aligned}$$

In above translation, since every L deleted from S_1 is also in S_2', the answer set S of Π^* can then be expressed as $S = S_1 \cup S_2'$. An important fact is observed from the construction of S_1:

Fact 1. $L \in S_1$ iff there exists some rule in Π_1^* of the form

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

such that $L_1, \cdots, L_m \in S_1$ and $L_{m+1}, \cdots,$ or $L_n \notin S_1$.

Now we prove S_1 is an answer set of Π_1^*. We do Gelfond-Lifschitz transformation on Π^* in terms of set $S = S_1 \cup S_2'$. After such transformation, we can write Π^{*S} as form $\Pi_1^{*'} \cup \Pi_2^{*'}$, where $\Pi_1^{*'} \subseteq \Pi_1^*$ and $\Pi_2^{*'} \subseteq \Pi_2^*$. As $head(\Pi_2^*) \cap body(\Pi_1^*) = \emptyset$, any literal in S_2' will not cause a deletion of a rule from Π_1^* in the Gelfond-Lifschitz transformation. Then it is easy to see that $\Pi_1^{*'} = \Pi_1^{*S_1}$. From **Fact 1**, it concludes that literal $L \in S_1$ iff there is a rule $L \leftarrow L_1, \cdots L_m$ in $\Pi_1^{*S_1}$ and $L_1, \cdots, L_m \in S_1$. This follows that S_1 is an answer set of $\Pi_1^{*S_1}$, and then an answer set of Π_1^*.

Now we transfer S_2' into S_2 by the following step: if $S_1 \cap S_2' = \emptyset$, then $S_2 = S_2'$; otherwise, let

$$\begin{aligned} S_2 = S_2' - \{L \mid\ & L \in S_1 \cap S_2', \text{ and for each rule} \\ & L \leftarrow L_1, \cdots, L_m,\ not\ L_{m+1}, \cdots,\ not\ L_n \text{ in } \Pi_2^*, \text{ there exists some} \\ & L_j(1 \leq j \leq m) \notin S_1 \cup S_2', \text{ or } L_j \in S_1 \cup S_2'(m+1 \leq j \leq n)\}. \end{aligned}$$

After this translation, S can be expressed as $S = S_1 \cup S_2$. An important fact is also observed from the translation of S_2:

Fact 2. $L \in S_2$ iff there exists some rule in $\Pi_2^{*'}$ of the form

$$L \leftarrow L_1, \cdots, L_k, L_{k+1}, \cdots, L_m$$

such that $L_1, \cdots, L_k \in S_1$ and $L_{k+1}, \cdots, L_m \in S_2$.

Now we prove S_2 is an answer set of $e(\Pi_1^*, S_1)$. Recall that $\Pi^{*S} = \Pi_1^{*'} \cup \Pi_2^{*'} = \Pi_1^{*S_1} \cup \Pi_2^{*'}$. From **Fact 2**, it is clear that there exists a subset Δ of S_1 such that S_2 is an answer set of $e(\Pi_2^{*'}, \Delta)$ and $e(\Pi_2^{*'}, \Delta)^{S_2} = e(\Pi_2^{*'}, \Delta)$. On the other hand, from the construction of $e(\Pi_2^*, S_1)$, it is easy to see that $e(\Pi_2^*, S_1)^{S_2} = e(\Pi_2^{*'}, \Delta) = e(\Pi_2^{*'}, \Delta)^{S_2}$. So S_2 is also an answer set of $e(\Pi_2^*, S_1)$.

Now we show *Result 2*. The fact that $\Pi_1^* \cup \{First \leftarrow\}$ is a reduct of $\mathcal{P}^1$ is proved based on a construction of a 1-1 correspondence between the computation of $\mathcal{P}$'s reduct and $\mathcal{P}^1$'s reduct. Let $\{\Pi^i\}$ $(i = 0, 1, \cdots)$ be the series generated by computing $\mathcal{P}$'s reduct (see Definition 2), and $\{\Pi'^i\}$ $(i = 0, 1, \cdots)$ be the series generated by computing $\mathcal{P}^1$'s reduct. From the specification of $\mathcal{P}^1$ and condition $head(\Pi_2) \cap body(\Pi_1) = \emptyset$, we observe that for each Π^i, which is obtained from Π^{i-1} by eliminating some rules from Π^{i-1}, if some rule in Π_1 is deleted, then this rule must be also deleted in Π'^i; if no rule in Π_1 is deleted (e.g. all rules deleted from Π^{i-1} are in Π_2), then we set $\Pi'^i = \Pi'^{i-1}$. Then it is clear that every rule in

$\Pi_1^* \cup \{First \leftarrow\}$ must be also in the reduct of $\mathcal{P}^1$ and *vice versa*. This concludes that $\Pi_1^* \cup \{First \leftarrow\}$ is a reduct of $\mathcal{P}^1$. Similarly we can show that $e(\Pi_2^*, S_1) \cup \{First \leftarrow\}$ is a reduct of $\mathcal{P}^2$. $\square$

Once a PLP has a split, by applying Theorem 5, we eventually reduce the computation of a large PLP's answer sets to the computation of two smaller PLPs' answer sets. In a general case, it is also possible to split a large PLP into a series of smaller PLPs.

Definition 8
Let $\mathcal{P} = (\Pi, \mathcal{N}, <)$. We say that $(\mathcal{P}^1, \cdots, \mathcal{P}^k)$ is a *split* of $\mathcal{P}$, if there exist k disjoint subsets $\Pi_1, \cdots, \Pi_k$ of Π, where $\Pi = \bigcup_{i=1}^{k} \Pi_i$, such that

1. $head(\Pi_i) \cap body(\bigcup_{j=1}^{i-1} \Pi_j) = \emptyset$, $(i = 2, \cdots, k)$,
2. $\mathcal{P}^1 = (\Pi_1 \cup \{N_0 : First \leftarrow\}, \mathcal{N}, <)$, where for any $r, r' \in \Pi_1$, $\mathcal{N}(r) < \mathcal{N}(r') \in \mathcal{P}(<)$ implies $\mathcal{N}(r) < \mathcal{N}(r') \in \mathcal{P}^1(<)$, and if there exists some $r'' \notin \Pi_1$ and $first^{<}(r) = r''$, then $N_0 < \mathcal{N}(r) \in \mathcal{P}'(<)$;
3. $\mathcal{P}^i = (e(\Pi_i, \bigcup_{j=1}^{i-1} S_j) \cup \{N_0 : First \leftarrow\}, \mathcal{N}, <)$, where S_j is an answer set of $\mathcal{P}^j$, for any $r, r' \in e(\Pi_i, \bigcup_{j=1}^{i-1} S_j)$, $\mathcal{N}(original(r)) < \mathcal{N}(original(r')) \in \mathcal{P}(<)$ implies $\mathcal{N}(r) < \mathcal{N}(r') \in \mathcal{P}^i(<)$, and if there exists some $r'' \notin \Pi_i$ and $first^{<}(original(r)) = r''$, then $N_0 < \mathcal{N}(r) \in \mathcal{P}^i(<)$.

A split $(\mathcal{P}^1, \cdots, \mathcal{P}^k)$ is called $\bigcup_{i=1}^{k-1} S_i$*-dependent* if S_i is an answer set of $\mathcal{P}^i$ $(i = 1, \cdots, k-1)$ and each $\mathcal{P}^{i+1}$ is formed based on $\bigcup_{j=1}^{i} S_j$ as described in condition 3 above.

Now using a similar technique as described in the proof of Theorem 5, we have the following general splitting result.

Theorem 6
Let $(\mathcal{P}^1, \cdots, \mathcal{P}^k)$ be a $\bigcup_{i=1}^{k-1} S_i$*-dependent* split of $\mathcal{P}$ as defined in Definition 8. A set of ground literals S is a consistent answer set of $\mathcal{P}$ if and only if $S = \bigcup_{i=1}^{k} S_i - \{First\}$, where S_k is an answer set of $\mathcal{P}^k$, and $\bigcup_{i=1}^{k} S_i$ is consistent.

Example 5
Consider PLP $\mathcal{P}_7 = (\Pi, \mathcal{N}, <)$ as follows:

$N_1 : A \leftarrow notB$,
$N_2 : B \leftarrow notA$,
$N_3 : C \leftarrow not\neg C$,
$N_4 : D \leftarrow notB$,
$N_5 : \neg D \leftarrow notD$,
$N_1 < N_2 < N_3 < N_4 < N_5$.

Let $\Pi_1 = \{N_1, N_2\}$, $\Pi_2 = \{N_3, N_4\}$, and $\Pi_3 = \{N_5\}$. Clearly, $head(\Pi_2) \cap body(\Pi_1) = \emptyset$ and $head(\Pi_3) \cap body(\Pi_1 \cup \Pi_2) = \emptyset$. Then a split of $\mathcal{P}_7$, denoted as $(\mathcal{P}_7^1, \mathcal{P}_7^2, \mathcal{P}_7^3)$, can be constructed. Ignoring the detail, this split is illustrated as follows:

$\mathcal{P}_7^1$:	$\mathcal{P}_7^2$:	$\mathcal{P}_7^3$:
$N_0 : First \leftarrow$,	$N_0 : First \leftarrow$,	$N_0 : First \leftarrow$,
$N_1 : A \leftarrow notB$,	$N_3 : C \leftarrow not\neg C$,	
$N_2 : B \leftarrow notA$,	$N_4 : D \leftarrow notB$,	
$N_1 < N_2$,	$N_0 < N_3 < N_4$,	

Then according to Theorem 6, each answer set of $\mathcal{P}_7$ can be represented as $S_1 \cup S_2 \cup S_3 - \{First\}$, where S_1, S_2, and S_3 are answer sets of $\mathcal{P}_7^1$, $\mathcal{P}_7^2$ and $\mathcal{P}_7^3$ respectively, which are $\{First, A\}$, $\{First, C, D\}$ and $\{First\}$ respectively. So $\{A, C, D\}$ is the unique answer set of $\mathcal{P}_7$.

5 Conclusion

In this paper, we have proved two major results for prioritized logic programs: the unique answer set theorem and splitting theorems for prioritized logic programs. By introducing the concept of mutual defeasibility, the first result provides a sufficient condition for the unique answer set of a prioritized logic program. It is observed that the sufficient condition in Theorem 4 is weaker than the local stratification as required for extended logic programs.

Our splitting theorems, on the other hand, illustrated that as in the case of extended logic programs, under certain conditions, the computation of answer sets of a prioritized logic program can be simplified. It is interesting to note that by omitting preference relation $<$, our splitting theorems actually also present new results for splitting usual extended logic program which generalizes Lifschitz and Turner's result (Lifschitz & Turner, 1994). Consider an extended logic program Π consisting of the following rules:

$A \leftarrow not\ C$,
$A \leftarrow not\ B$,
$B \leftarrow not\ A$.

This program does not have a non-trivial split under Lifschitz and Turner's Splitting Set Theorem, but under our splitting theorem condition, Π can be split into Π_1 and Π_2 as follows:

Π_1:	Π_2:
$A \leftarrow not\ C$,	$A \leftarrow not\ B$,
	$B \leftarrow not\ A$,

such that $body(\Pi_1) \cap head(\Pi_2) = \emptyset$. It is observed that $\{A\}$ is the unique answer set of Π_1, and the unique answer set of Π is then obtained from Π_1's answer set $\{A\}$ and the answer set of $e(\Pi_2, \{A\})$, which is also $\{A\}$. So we get the unique answer set of Π $\{A\}$. A detailed discussion on the relationship between Lifschitz and Turner's Splitting Set Theorem and our splitting result on extended logic programs is referred to the author's another paper (Zhang, 1999).

We should state that our results proved in this paper are based on a specific formulation of prioritized logic programming, and hence it is not clear yet whether they are generally suitable for other prioritized default reasoning systems, e.g. (Delgrande *et al.*, 2000; Grosof, 1997; Wang *et al.*, 2000). However, since the traditional answer set semantics was employed in our development of prioritized logic programming, we would expect that our results could be extended to other answer set semantics based PLP frameworks. This will be an interesting topic for our future work.

Finally, it is also worth mentioning that besides the idea of developing a "prioritized version" of answer set semantics for PLP (like the approach we discussed in this paper), there are other approaches to PLP in which the semantics of PLP is defined by modular and simple translation of PLP programs into standard logic programs. Work on this direction is due to Gelfond and Son (Gelfond & Son, 1998), Delgrande, Schaub and Tompits (Delgrande *et al.*, 2000) and some other researchers. Recently, Schaub and Wang further investigated a series of uniform characterizations among these approaches (Schaub & Wang, 2001). For these approaches, the classical splitting set theorem (Lifschitz & Turner, 1994) can be used to simplify the reasoning procedure of PLP.

References

Apt, K.R., & Bol, R.N. (1994). Logic programming and negation: A survey. *Journal of logic programming*, **19,20**, 9–71.

Brewka, G. (1996). Well-founded semantics for extended logic programs with dynamic preferences. *Journal of artificial intelligence research*, **4**, 19–36.

Brewka, G., & Eiter, T. (1999). Preferred answer sets for extended logic programs. *Artificial intelligence*, **109**, 297–356.

Choelwinski, P. (1994). Stratified default logic. *Pages 456–470 of: Proceedings of computer science logic*. Springer, LNCS 933.

Das, S.K. (1992). *Deductive database and logic programming*. Addison-Wesley Publishers Ltd.

Delgrande, J., Schaub, T., & Tompits, H. (2000). Logic programs with complied preferences. *Pages 392–398 of: Proceedings of the 14th european conference on artificial intelligence (ecai-2000)*.

Gelfond, M., & Lifschitz, V. (1988). The stable model semantics for logic programming. *Pages 1070–1080 of: Proceedings of the fifth joint international conference and symposium*. MIT Press.

Gelfond, M., & Lifschitz, V. (1991). Classical negation in logic programs and disjunctive databases. *New generation computing*, **9**, 365–386.

Gelfond, M., & Son, T.C. (1998). Reasoning with prioritized defaults. *Pages 164–224 of: Lnai 1471*. Springer.

Grosof, B.N. (1997). Prioritized conflict handling for logic programs. *Pages 197–212 of: Proceedings of the 1997 international logic program symposium (ilps'97)*. MIT Press.

Lifschitz, V., & Turner, H. (1994). Splitting a logic program. *Pages 23–37 of: Proceedings of eleventh international conference on logic programming*. MIT Press.

Schaub, T., & Wang, K. (2001). A comparative study of logic programs with preference. *Pages 597–602 of: Proceedings of the 17th international joint conference on artificial intelligence (ijcai-2001)*. Morgan Kaufmann Publishers Inc.

Wang, K., Zhou, L., & Lin, F. (2000). Alternating fixpoint theory for logic programs with priority. *Pages 164–178 of: Proceedings of international joint conference on computational logic (cl-2000)*.

Zhang, Y. (1999). Monotonicity in rule based update. *Pages 471–485 of: Proceedings of the 1999 international conference on logic programming (iclp'99)*. MIT Press.

Zhang, Y. (2001). The complexity of logic program update. *Pages 630–643 of: Proceedings of the 14th australian joint conference on artificial intelligence (ai2001)*. Springer, LNAI 2256.

Zhang, Y., & Foo, N.Y. (1997). Answer sets for prioritized logic programs. *Pages 69–83 of: Proceedings of the 1997 international logic programming symposium (ilps'97)*. MIT Press.