**University of Western Sydney**

**School of Computing and Information Technology**

# The Characterization on the Uniqueness of Answer Set for Prioritized Logic Programs

**Yan Zhang and Yun Bai**

{yan,ybai}@cit.uws.edu.au

JULY 2003

TECHNICAL REPORT NO. CIT/33/2003

**Abstract**

Prioritized logic programming has illustrated its rich expressiveness and flexibility in knowledge representation and reasoning. However, some important aspects of prioritized logic programs have yet to be thoroughly explored. In this paper, we investigate basic properties of prioritized logic programs in the context of answer set semantics. Specifically, we propose a characterization on the uniqueness of answer set for prioritized logic programs, which has a weaker form than the traditional local stratification for general logic programs.
**keywords**: foundation of logic programming, knowledge representation, semantics

## 1 Introduction

Prioritized logic programming has illustrated its rich expressiveness and flexibility in knowledge representation, reasoning about action and logic program based updates [2, 5, 7, 8]. However, some important aspects of prioritized logic programs have yet to be thoroughly explored. In this paper, we investigate several properties of prioritized logic programs in the context of answer set semantics. Specifically, we propose a characterization on the uniqueness of answer set for prioritized logic programs. We show that our characteristic condition is weaker than the traditional local stratification for general logic programs [1]. The paper is organized as follows. Section 2 briefly reviews the syntax and semantics of prioritized logic programs. Section 3 proves a unique answer set theorem for prioritized logic programs. Finally, section 4 concludes the paper with some remarks.

## 2 Prioritized Logic Programs

To begin with, we first briefly review prioritized logic programs (PLPs) proposed by the author recently [7]. To specify PLPs, we first introduce the extended logic program and its answer set semantics developed by Gelfond and Lifschitz [4]. A language $\mathcal{L}$ of extended logic programs is determined by its object constants, function constants and predicates 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 $L_1, \cdots, L_m$,*not* $L_{m+1}, \cdots$, *not* $L_n$ is called the *body* of the rule. Obviously, the body of a rule could be empty. A term, atom, literal, or rule is *ground* if no variable occurs in it. An *extended logic program* $\Pi$ is a collection of rules.

To evaluate an extended logic program, Gelfond and Lifschitz proposed answer set semantics for extended logic programs. Let $\Pi$ be an extended logic program not containing *not* and *Lit* the set of all ground literals in the language of $\Pi$. The *answer set* of $\Pi$, denoted as $Ans(\Pi)$, is the smallest subset $S$ of *Lit* such that (i) for any rule $L_0 \leftarrow L_1, \cdots, L_m$ from $\Pi$, 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 $\Pi$ be an arbitrary extended logic program. For any subset $S$ of *Lit*, let $\Pi^S$ be the logic program obtained from $\Pi$ 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 $\Pi$, denoted by $Ans(\Pi)$, iff $S$ is an answer set of $\Pi^S$, i.e. $S = Ans(\Pi^S)$.

It is easy to see that an extended logic program may have one, more than one, or no answer set at all. The language $\mathcal{L}^P$ of PLPs is a language $\mathcal{L}$ of extended logic programs [4] with the following augments:
- *Names:* $N, N_1, N_2, \cdots$.
- A strict partial ordering $<$ on names.
- A naming function $\mathcal{N}$, which maps a rule to a name.

A *prioritized logic program* (PLP) $\mathcal{P}$ is a triple $(\Pi, \mathcal{N}, <)$, where $\Pi$ is an extended logic program, $\mathcal{N}$ is a naming function mapping each rule in $\Pi$ 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:

$\mathcal{P}_1$:
$N_1 : Fly(x) \leftarrow Bird(x)$, *not* $\neg Fly(x)$,
$N_2 : \neg Fly(x) \leftarrow Penguin(x)$, *not* $Fly(x)$,
$N_3 : Bird(Tweety) \leftarrow$,

[1]We also call $\Pi^S$ is the Gelfond-Lifschitz transformation of $\Pi$ in terms of $S$.

$N_4 : Penguin(Tweety) \leftarrow,$
$N_2 < N_1.$

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 $N_1$ after applying $N_2$ so that the desired solution $\neg Fly(Tweety)$ can be derived.

In a PLP or an extended logic program, we usually view a rule including variables to be the set of all ground instances of this rule formed from the set of ground literals in the language.

**Definition 1** *Let* $\Pi$ *be a ground extended logic program and r a rule with the form* $L_0 \leftarrow L_1, \cdots, L_m$, *not* $L_{m+1}, \cdots,$ *not* $L_n$ *(r does not necessarily belong to* $\Pi$*). Rule r is* defeated *by* $\Pi$ *iff* $\Pi$ *has an answer set and for any answer set* $Ans(\Pi)$ *of* $\Pi$*, there exists some* $L_i \in Ans(\Pi)$*, where* $m + 1 \leq i \leq n$*.*

The evaluation of a PLP will be based on its ground form. That is, for any PLP $\mathcal{P} = (\Pi, \mathcal{N}, <)$, we consider its *ground instantiation* $\mathcal{P}' = (\Pi', \mathcal{N}', <')$, where $\Pi'$, $\mathcal{N}'$ and $<'$ are ground instantiations of $\Pi$, $\mathcal{N}$ and $<$ respectively[2]. However, this requires some restriction on a PLP since not every PLP's ground instantiation presents a consistent information with respect to the original PLP.

Given a PLP $\mathcal{P} = (\Pi, \mathcal{N}, <)$. We say $\mathcal{P}$ is *well formed* if there does not exist a rule $r'$ that is an instance of two different rules $r_1$ and $r_2$ in $\Pi$ and $\mathcal{N}(r_1) < \mathcal{N}(r_2) \in \mathcal{P}(<)$. In the rest of this paper, we will only consider well formed PLPs in our discussions, and consequently, the evaluation for an arbitrary program $\mathcal{P} = (\Pi, \mathcal{N}, <)$ 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.

Let us consider program $\mathcal{P}_1$ once again. Since $N_2 < N_1$ and $N_1$ is defeated by $\mathcal{P}_1 - \{N_1\}$ (i.e. the unique answer set of $\mathcal{P}_1 - \{N_1\}$ is $\{Bird(Tweety), Penguin(Tweety), \neg Fly(Tweety)\}$), rule $N_1$ should be ignored during the evaluation of $\mathcal{P}_1$.

**Definition 2** *Let* $\mathcal{P} = (\Pi, \mathcal{N}, <)$ *be a prioritized extended logic program.* $\mathcal{P}^<$ *is a* reduct *of* $\mathcal{P}$ *with respect to* $<$ *if and only if there exists a sequence of sets* $\Pi_i$ *(*$i = 0, 1, \cdots$*) 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* $r_1, \cdots,$ *are defeated by* $\Pi_{i-1} - \{r_1, r_2, \cdots\}$*, and (b) there does not exist a rule* $r' \in \Pi_{i-1}$ *such that* $N(r_j) < N(r')$ *for some j (*$j = 1, 2, \cdots$*) and* $r'$ *is defeated by* $\Pi_{i-1} - \{r'\}\}$*;*
3. $\mathcal{P}^< = \bigcap_{i=0}^{\infty} \Pi_i$*.*

In Definition 2, $\mathcal{P}^<$ is a ground extended logic program obtained from $\Pi$ by eliminating some rules from $\Pi$. 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 $\Pi_{i-1}$ 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 is worth to note that the generation of a reduct of a PLP is based on the ground form of its extended logic program part. Furthermore, if $\mathcal{N}(r_1) < \mathcal{N}(r_2)$ holds in a PLP where $r_1$ or $r_2$ includes variables, then $\mathcal{N}(r_1) < \mathcal{N}(r_2)$ is actually viewed as the set of $<$-relations $\mathcal{N}(r'_1) < \mathcal{N}(r'_2)$, where $r'_1$ and $r'_2$ are ground instances of $r_1$ and $r_2$ respectively.

**Definition 3** *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* $\mathcal{P}$*, denoted as* $Ans^P(\mathcal{P})$*, iff* $S = Ans(\mathcal{P}^<)$ *for some reduct* $\mathcal{P}^<$ *of* $\mathcal{P}$*. Given a PLP* $\mathcal{P}$*, a ground literal L is* entailed *from* $\mathcal{P}$*, denoted as* $\mathcal{P} \models L$*, if L belongs to every answer set of* $\mathcal{P}$*.*

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

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

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

[2]Note that if $\mathcal{P}'$ is a ground instantiation of $\mathcal{P}$, then $\mathcal{N}(r_1) < \mathcal{N}(r_2) \in \mathcal{P}(<)$ implies $\mathcal{N}'(r'_1) <' \mathcal{N}(r'_2) \in \mathcal{P}'(<')$, where $r'_1$ and $r'_2$ are ground instances of $r_1$ and $r_2$ respectively.

$Ans^P(\mathcal{P}_1) = \{Bird(Tweety), Penguin(Tweety), \neg Fly(Tweety)\}$.

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 $\{A, B, D\}$.

## 3 Basic Properties of Prioritized Logic Programs

In this section, 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 $\Pi$ be an extended logic program. We use $\mathcal{A}(\Pi)$ to denote the class of all answer sets of $\Pi$. 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 $\Pi$ has an answer set. Then for any $i$ and $j$ where $i < j$, $\mathcal{A}(\Pi_j) \subseteq \mathcal{A}(\Pi_i)$.*

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

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

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

$r_1 : L_1 \leftarrow \cdots, \text{not}\, L_1', \cdots,$
$r_2 : L_2 \leftarrow \cdots, \text{not}\, L_2', \cdots,$
$\quad \cdots,$

*where $L_1', L_2', \cdots \in S_j$. Now consider Gelfond-Lifschitz transformation of $\Pi_j$ in terms of $S_j$. It is clear that during the transformation, each rule in $\Pi_j$ 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 $\Pi_j$, This makes $\Pi_{j-1}$. Then we have $\Pi_j^{S_j} = \Pi_{j-1}^{S_j}$. So $S_j$ is also an answer set of $\Pi_{j-1}$.* ■

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

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

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

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

$\Pi_0$:

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

$\Pi_1$:

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

$\Pi_2$:

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

*It is easy to verify that $\Pi_0$ has three answer sets $\{A, C\}$, $\{B, D\}$ and $\{A, D\}$, $\Pi_1$ has two answer sets $\{A, C\}$ and $\{A, D\}$, and $\Pi_2$ 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 $\Pi_i$ for some reduct chain $\{\Pi_i\}$ ($i = 0, 1, 2, \cdots$) of $\mathcal{P}$.*

**Proof 2** *(1 ⇒ 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 $\Pi_i$ ($i = 1, \cdots, k$).*

*(2 ⇒ 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 $\Pi_k$ 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 $\Pi$.*

**Proof 3** *From Theorem 1, it shows that if $\mathcal{P}$ has an answer set S, then S is also an answer set of each $\Pi_i$ 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 $\Pi$.*

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 $\Pi$ has an answer set.*

**Proof 4** *According to Corollary 1, we only need to prove that if $\Pi$ has an answer set, then $\mathcal{P}$ also has an answer set. Suppose $\Pi$ 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 $\Pi_i$ ($i = 0, 1, \cdots$) 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 $\Pi_k$ has an asnwer 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 5** *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 [6] (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.*

## 4 A Unique Answer Set Theorem

Now we try to provide a unique characterization of the answer set for a prioritized logic program. To investigate this issue, we first extend the concept of local stratification for general logic programs [1] to extended logic programs.

**Definition 4** *Let $\Pi$ be an extended logic program and Lit be the set of all ground literals of $\Pi$.*

1. *A* local stratification *for $\Pi$ 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(not L) $=$ stratum(L) $+ 1$, where L is a ground literal.*
3. *A rule $L_0 \leftarrow L_1, \cdots, L_m$, not $L_{m+1}, \cdots$, not $L_n$ in $\Pi$ 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. *$\Pi$ is called* locally stratified *with respect to stratum if all of its rules are locally stratified. $\Pi$ is called* locally stratified *if it is locally stratified with respect to some local stratification.*

Let $\Pi$ be a ground extended logic program and $r$ be a rule in $\Pi$ of the form:

$L_0 \leftarrow L_1, \cdots, L_m$, not $L_{m+1}, \cdots$, not $L_n$.

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 $\Pi$, e.g. $body(\Pi) = \bigcup_{r \in \Pi} body(r)$. If $\Pi$ 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 of $\Pi$.

**Definition 5** *Let $\Pi$ be an extended logic program and $r_p$ and $r_q$ be two rules in $\Pi$. 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 $\Pi$ if and only if $neg(r_q) \cap \mathcal{D}(r_p) \neq \emptyset$. $r_p$ and $r_q$ are called* mutually defeasible *in $\Pi$ if $r_q$ is defeasible through $r_p$ and $r_p$ is defeasible through $r_q$ in $\Pi$.*

Intuitively, if $r_q$ is defeasible through $r_p$ in $\Pi$, 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 $\Pi$, it is possible to defeat rule $r_q$ if rules $r_1, \cdots, r_k$ are triggered as well. As a special case that $\mathcal{D}(r_p) = \emptyset$, $r_q$ 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 3** *Given a ground extended logic program $\Pi$. If $\Pi$ is locally stratified, then there are no mutually defeasible pairs of rules in $\Pi$.*

It is easy to observe that the converse of Proposition 3 does not hold. Consider an extended logic program consisting of three rules

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

There does not exist two rules in this program that are mutually defeasible. But this program is not locally stratified.

**Proposition 4** *Let $\Pi$ be a ground extended logic program. If $\Pi$ is locally stratified, then $\Pi$ has a unique answer set*[3].

[3]Recall that if $\Pi$ has an inconsistent answer set, we will denote it as *Lit*.

The above result is easy to prove from the corresponding result for general logic programs showed in [3] based on Gelfond and Lifschitz's translation from an extended logic program to a general logic program [4]. It is observed that for a PLP $\mathcal{P} = (\Pi, \mathcal{N}, <)$, if $\Pi$ is locally stratified, then $\mathcal{P}$ will also have a unique answer set. In other words, $\Pi$'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 an arbitrary PLP. A $<$-*partition *of $\Pi$ in $\mathcal{P}$ is a finite collection $\{\Pi_1, \cdots, \Pi_k\}$, where $\Pi = \Pi_1 \cup \cdots \cup \Pi_k$ and $\Pi_i$ and $\Pi_j$ 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 $\mathcal{P}_4 = (\Pi, \mathcal{N}, <)$:*

$\mathcal{P}_4$:
$N_1 : A \leftarrow \text{not } B, \text{not } C,$
$N_2 : B \leftarrow \text{not } \neg C,$
$N_3 : C \leftarrow \text{not } A, \text{not } \neg C,$
$N_4 : \neg C \leftarrow \text{not } C,$
$N_1 < N_2, N_2 < N_4, N_3 < N_4.$

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

$\Pi_1$:
$N_1 : A \leftarrow \text{not } B, \text{not } C,$
$N_3 : C \leftarrow \text{not } A, \text{not } \neg C,$
$\Pi_2$:
$N_2 : B \leftarrow \text{not } \neg C,$
$\Pi_3$:
$N_4 : \neg C \leftarrow \text{not } C.$

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

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

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

**Proof 6** *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 $\Pi_i$ and $\Pi_j$ ($1 < i, j$) respectively such that $r_p$ and $r_q$ are mutually defeasible in $\Pi$.*

*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 $1 < i, j$; (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, suppose 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:*

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

*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 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* $\Pi$*.* ■

## 5 Concluding Remarks

In this paper we investigated basic properties of PLPs under answer set semantics and provided a unique characterization for the answer set of PLPs. It should be noted that although the uniqueness of answer set for general and extended logic programs has been studied previously, this paper presents the first investigation on this issue for prioritized logic programs. The detailed comparison between our prioritized logic programs and other related proposals is beyond the scope of this paper. Here we only illustrate the most important feature of our approach in preferred defeasible reasoning. The major difference between our approach and other approaches is that by viewing the preference to be defeasible, our approach guarantees that every prioritized logic program has an answer set iff the underlying extended logic program has one - this principle is essential for dealing with logic program update in many situations as we have shown in [8]. So in general our approach provides a flexible framework for prioritized defeasible reasoning.

## References

[1] K.R. Apt and R.N. Bol, Logic programming and negation: A survey. *Journal of Logic Programming*, **19,20** (1994) 9-71.

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

[3] M. Gelfond and V. Lifschitz, The stable model semantics for logic programming. In *Proceedings of the Fifth Joint International Conference and Symposium*, pp 1070-1080. MIT Press, 1988.

[4] M. Gelfond and V. Lifschitz, Classical negation in logic programs and disjunctive databases. *New Generation Computing*, **9** (1991) 365-386.

[5] B.N. Grosof, Prioritized conflict handling for logic programs. In *Proceedings of the 1997 International Logic Program Symposium (ILPS'97)*, pp 197-212. MIT Press, 1997.

[6] V. Lifschitz and H. Turner, Splitting a logic program. In *Proceedings of Eleventh International Conference on Logic Programming*, pp 23-37. MIT Press, 1994.

[7] Zhang and N.Y. Foo, Answer sets for prioritized logic programs. In *Proceedings of the 1997 International Logic Programming Symposium (ILPS'97)*, pp 69-83. MIT Press, 1997

[8] Y. Zhang, Logic program based updates. Manuscript, 2003.