Monotonicity in Rule Based Update

Yan Zhang
School of Computing and Information Technology
University of Western Sydney, Nepean
Kingswood, NSW 2747, Australia
E-mail: yan@cit.nepean.uws.edu.au

Abstract

An important characteristic for many formulations of knowledge representation and reasoning is that they are nonmonotonic. It has been, however, illustrated that under certain conditions, a formulation may satisfy some restricted monotonicity in its reasoning [2]. In this paper, we investigate this issue under the context of rule based update. We first present a general framework of rule based update in which a knowledge base is viewed as a set of literals and can be updated with a set of update rules which is represented by an extended logic program (also called update program). We then show that given a knowledge base $\mathcal{B}$ and an update program Π, (1) there exist some facts that can be always added into $\mathcal{B}$ such that updating the expansion of $\mathcal{B}$ with Π will not destroy any fact obtained from the update of $\mathcal{B}$ with Π, and (2) on the other hand, there also exist some update rules that can be always added into Π such that updating $\mathcal{B}$ with the expansion of Π will not destroy any fact obtained from the update of $\mathcal{B}$ with Π. We illustrate how these results can be used to simplify an update evaluation. Our proofs of restricted monotonicity theorems are based on a generalization of Lifschitz-Turner's Splitting Set Theorem on extended logic programs [4].

1 Introduction

An important characteristic for many formulations of knowledge representation and reasoning is that they are nonmonotonic. It has been, however, illustrated that under certain conditions, a formulation may satisfy some restricted monotonicity in its reasoning [2]. The purpose of this paper is to investigate this issue under the context of rule based update.

Generally, the rule based update addresses the following problem: given a knowledge base and a set of update rules, what is the resulting knowledge base after updating this knowledge base with the set of update rules? For example, consider a domain of specifying user access rights to a computer system. Given an access policy base including facts that *Peter* can access file F and he is in group G, if the computer system officer wants to update the policy base in terms of a rule saying that if a user belongs to group G, then the user is no longer allowed to access file F, then after updating this policy base with the rule, we would expect that *Peter* cannot access file F

any more.

Although some syntactic and semantic aspects of rule based update has been studied by some researchers [5, 1, 6], the monotonicity property related to rule based update still remains unclear. In this paper, we first present a general framework of rule based update in which a knowledge base is viewed as a set of literals and can be updated with a set of update rules where an update rule may include both classical negation and negation as failure. To handle the possible conflict between inertia rules and update rules, a prioritized logic program is employed to formalize the update specification in our framework. We then show that given a knowledge base $\mathcal{B}$ and an update program Π, (1) there exist some facts that can be always added into $\mathcal{B}$ such that updating the expansion of $\mathcal{B}$ with Π will not destroy any fact obtained from the update of $\mathcal{B}$ with Π, and (2) on the other hand, there also exist some update rules that can be always added into Π such that updating $\mathcal{B}$ with the expansion of Π will not destroy any fact obtained from the update of $\mathcal{B}$ with Π. We illustrate how these results can be used to simplify an update evaluation. Our proofs of restricted monotonicity theorems are based on a generalization of Lifschitz-Turner's Splitting Set Theorem on extended logic programs [4], which presents another contribution of this paper.

The paper is organized as follows. Next section presents a general framework of rule based update. Section 3 investigates restricted monotonicity properties of our rule based update. Section 4 presents a generalization of Lifschitz-Turner's splitting theorem on extended logic programs, which provides a basis of proving our major restricted monotonicity results discussed in section 3. Section 5 then gives the proofs of two restricted monotonicity theorems for rule based update. Finally, section 6 concludes the paper with some remarks.

2 A Framework of Rule Based Update

In this section, we develop a general framework of rule based update. As we allow an update rule to include both classical negation and negation as failure, possible conflict between inertia rules and update rules may occur in the evaluation of an update. To solve this kind of conflict, a prioritized logic program is then used in our framework.

2.1 PLPs - An Overview

To begin with, we need to briefly review prioritized logic programs (PLPs) proposed by Zhang and Foo recently [8]. We first introduce the extended logic program and its answer set semantics defined by Gelfond and Lifschitz [3]. 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 symbol 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:

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

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* Π is a collection of rules.

To evaluate a extended logic program, Gelfond and Lifschitz proposed the answer set semantics for extended logic programs. For simplicity, we treat a rule r in Π with variables as the set of all ground instances of r formed from the set of ground literals of the language of Π. In the rest of paper, we will not explicitly declare this assumption whenever there is no ambiguity in our discussion. Let Π be an extended logic program not containing *not* and *Lit* the set of all ground literals in the language of Π. The *answer set* of Π, 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 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 an 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, denoted $Ans(\Pi)$, iff S is an answer set of Π^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. A rule with the form (1) is *satisfied* in a set of ground literals S if and only if the fact that $L_1, \cdots, L_m$ are in S and $L_{m+1}, \cdots$, and L_n are not in S implies the fact that L_0 is in S. Clearly, each rule of an extended logic program Π is satisfied in every answer set of Π.

The language $\mathcal{L}^P$ of PLPs is a language $\mathcal{L}$ of extended logic programs [3] 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 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 $\mathcal{P}$. 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:

$\mathcal{P}_1$:
$N_1 : Fly(x) \leftarrow Bird(x), not \neg Fly(x),$

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

$N_2 : \neg Fly(x) \leftarrow Penguin(x),\ not\ Fly(x)$,
$N_3 : Bird(Tweety) \leftarrow$,
$N_4 : Penguin(Tweety) \leftarrow$,
$N_2 < N_1$.

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 after applying N_2 so that the desired solution $\neg Fly(Tweety)$ can be derived. This idea is formalized by following definitions.

Definition 1 *Let* Π *be an 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* Π*). Rule* r *is* defeated *by* Π *iff* Π *has answer set(s) and for every answer set* $Ans(\Pi)$ *of* Π*, there exists some* $L_i \in Ans(\Pi)$*, where* $m+1 \leq i \leq n$.

Definition 2 *Let* $\mathcal{P} = (\Pi, \mathcal{N}, <)$ *be a PLP and* $\mathcal{P}(<^+)$ *denote the* $<$*-closure of* $\mathcal{P}$ *(i.e.* $\mathcal{P}(<^+)$ *is the smallest set containing all preference relations of* $\mathcal{P}$ *and closed under transitivity).* $\mathcal{P}^<$ *is a* reduct *of* $\mathcal{P}$ *with respect to* $<$ *iff there exists a sequence of sets* Π_i $(i = 0, 1, \cdots)$ *such that:*

1. $\Pi_0 = \Pi$;
2. $\Pi_i = \Pi_{i-1} - \{r_1, \cdots, r_k \mid$ *(a) there exists* $r \in \Pi_{i-1}$ *such that* $\mathcal{N}(r) < \mathcal{N}(r_i) \in \mathcal{P}(<^+)$ $(i = 1, \cdots, k)$ *and* $r_1, \cdots, r_k$ *are defeated by* $\Pi_{i-1} - \{r_1, \cdots, r_k\}$*, and (b) there does not exist a rule* $r' \in \Pi_{i-1}$ *such that* $\mathcal{N}(r_j) < \mathcal{N}(r')$ *for some* j $(j = 1, \cdots, k)$ *and* r' *is defeated by* $\Pi_{i-1} - \{r'\}\}$;
3. $\mathcal{P}^< = \bigcap_{i=0}^{\infty} \Pi_i$.

Clearly $\mathcal{P}^<$ is an extended logic program obtained from Π by eliminating some rules from Π. In particular, if $\mathcal{N}(r) < \mathcal{N}(r')$ and $\Pi - \{r'\}$ defeats r', rule r' is eliminated from Π if no *less preferred rule* can be eliminated (i.e. conditions (a) and (b) in (ii)). This procedure is continued until a fixed point is reached. Note that due to the transitivity of $<$, we need to consider each $\mathcal{N}(r) < \mathcal{N}(r')$ in the $<$-closure of $\mathcal{P}$. It should be also noted that the reduct of a PLP may not be unique generally [8]. Now it is quite straightforward to define the answer set for a prioritized logic program.

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* $S = Ans(\mathcal{P}^<)$*, where* $Ans(\mathcal{P}^<)$ *is an answer set of extended logic program* $\mathcal{P}^<$*. A ground literal* L *is* derivable *from a PLP* $\mathcal{P}$*, denoted as* $\mathcal{P} \vdash L$*, iff* $\mathcal{P}$ *has answer set(s) and* L *belongs to every answer set of* $\mathcal{P}$.

Example 1 Using Definition 1 and 2, it is not difficult to conclude that $\mathcal{P}_1$ has a unique reduct: $\mathcal{P}_1^{<} = \{\neg Fly(x) \leftarrow Penguin(x),\ not\ Fly(x),\ Bird(Tweety) \leftarrow,\ Penguin(Tweety) \leftarrow\}$, and then from Definition 3, it has a unique answer set $Ans^P(\mathcal{P}_1) = \{Bird(Tweety),\ Penguin(Tweety),\ \neg Fly(Tweety)\}$. ■

2.2 Generalized Rule Based Update

Consider a language $\mathcal{L}$ of extended logic programs as described in section 2. We specify that a *knowledge base* $\mathcal{B}$ is a *consistent* set of ground literals of $\mathcal{L}$ and an *update program* Π is a set of rules of $\mathcal{L}$ with form (1) that are called *update rules*. Note that we allow a knowledge base to be *incomplete*. That is, a literal not in a knowledge base is treated as *unknown*.

We will use a prioritized logic program to specify an update of $\mathcal{B}$ by Π, where Π is an extended logic program. For this purpose, we first need to extend language $\mathcal{L}^P$ by the following way. We specify $\mathcal{L}^P_{new}$ to be a language of PLPs based on $\mathcal{L}^P$ with one more augment: For each predicate symbol P in $\mathcal{L}^P$, there is a corresponding predicate symbol $New\text{-}P$ in $\mathcal{L}^P_{new}$ with the same arity of P.

To simplifying our presentation, in $\mathcal{L}^P_{new}$ we use notation $New\text{-}L$ to denote the corresponding literal L in $\mathcal{L}$. For instance, if a literal L in $\mathcal{L}$ is $\neg P(x)$, then notation $New\text{-}L$ simply means $\neg New\text{-}P(x)$. We use Lit_{new} to denote the set of all ground literals of $\mathcal{L}^P_{new}$. Clearly, $Lit_{new} = Lit \cup \{New\text{-}L \mid L \in Lit\}$. Now we are ready to formalize our generalized rule-based update.

Definition 4 *Let $\mathcal{B}$, Π, $\mathcal{L}$, $\mathcal{L}^P$ and $\mathcal{L}^P_{new}$ be defined as above. The* specification of updating $\mathcal{B}$ with Π *is defined as a PLP of $\mathcal{L}^P_{new}$, denoted as $Update(\mathcal{B}, \Pi) = (\Pi^*, \mathcal{N}, <)$, as follows:*

1. Π^* *consists of following rules:*
   Initial knowledge rules*: for each L in $\mathcal{B}$, there is a rule $L \leftarrow$;*
   Inertia rules*: for each predicate symbol P in $\mathcal{L}$, there are two rules:*
   $New\text{-}P(x) \leftarrow P(x),\ not\ \neg New\text{-}P(x)$[2], *and*
   $\neg New\text{-}P(x) \leftarrow \neg P(x),\ not\ New\text{-}P(x)$,
   Update rules*: for each rule*
   $L_0 \leftarrow L_1, \cdots, L_m,\ not\ L_{m+1}, \cdots,\ not\ L_n$ *in* Π, *there is a rule*
   $New\text{-}L_0 \leftarrow New\text{-}L_1, \cdots,\ New\text{-}L_m,$
   $not\ New\text{-}L_{m+1}, \cdots,\ not\ New\text{-}L_n$;
2. *Naming function $\mathcal{N}$ assigns a unique name N for each rule in Π^*;*
3. *For any inertia rule with name N and update rule with name N', we specify $N < N'$.*

[2] x might be a tuple of variables.

An update specification $Update(\mathcal{B}, \Pi)$ *is* well defined *if it has answer set(s) and all its answer sets are consistent.*

It should be noted that in the above definition, we specify inertia rules to be more preferred than update rules in $Update(\mathcal{B}, \Pi)$. The intuitive idea behind this is that a preference ordering between an inertia rule and an update rule in $Update(\mathcal{B}, \Pi)$ will affect the evaluation of $Update(\mathcal{B}, \Pi)$ *only if* these two rules conflict with each other, eg. applying one rule causes the other inapplicable. On the other hand, a fact in the initial knowledge base $\mathcal{B}$ is always preferred to persist during an update whenever there is no violation of update rules[3]. Therefore, when conflicts occur between inertia and update rules, inertia rules should defeat the corresponding update rules. without conflicts between inertia and update rules, the preference ordering does not play any role in the evaluation of $Update(\mathcal{B}, \Pi)$. Also note that there will be at most $2k \cdot l$ $<$-relations in $Update(\mathcal{B}, \Pi)$, where k is the number of predicate symbols of $\mathcal{L}_P$ and l is the number of update rules in Π.

Finally, on the basis of Definition 4, we can formally define a knowledge base $\mathcal{B}'$ resulting from updating $\mathcal{B}$ with Π in a straightforward way.

Definition 5 *Let* $\mathcal{B}, \mathcal{P}, \mathcal{L}, \mathcal{L}^P$ *and* $\mathcal{L}^P_{new}$ *be specified as before, and* $Update(\mathcal{B}, \Pi)$ *the specification of updating* $\mathcal{B}$ *with* Π *as defined in Definition 4. A set of ground literals of* $\mathcal{L}$, $\mathcal{B}'$, *is called a* possible resulting knowledge base *with respect to the update specification* $Update(\mathcal{B}, \Pi)$, *iff* $\mathcal{B}'$ *satisfies the following conditions:*

1. *if* $Update(\mathcal{B}, \Pi)$ *has a consistent answer set, say* $Ans^P(Update(\mathcal{B}, \Pi))$, *then* $\mathcal{B}' = \{L \mid New\text{-}L \in Ans^P(Update(\mathcal{B}, \Pi))\}$.

2. *if* $Update(\mathcal{B}, \Pi)$ *is not well defined, then* $\mathcal{B}' = \mathcal{B}$.

Example 2 Let $\mathcal{B} = \{\neg A, B, C\}$ and $\Pi = \{\neg B \leftarrow not\ B,\ A \leftarrow C\}$. From Definition 4, the specification of updating $\mathcal{B}$ by Π, $Update(\mathcal{B}, \Pi)$, is as follows:

Initial knowledge rules:
$N_1 : \neg A \leftarrow$,
$N_2 : B \leftarrow$,
$N_3 : C \leftarrow$,
Inertia rules:
$N_4 : New\text{-}A \leftarrow A, not\ \neg New\text{-}A$,
$N_5 : New\text{-}B \leftarrow B, not\ \neg New\text{-}B$,
$N_6 : New\text{-}C \leftarrow C, not\ \neg New\text{-}C$,
$N_7 : \neg New\text{-}A \leftarrow \neg A, not\ New\text{-}A$,
$N_8 : \neg New\text{-}B \leftarrow \neg B, not\ New\text{-}B$,
$N_9 : \neg New\text{-}C \leftarrow \neg C, not\ New\text{-}C$,

[3]Note that an update rule in $Update(\mathcal{B}, \Pi)$ is defeasible if it contains a weak negation *not* in the body.

Update rules:

$N_{10} : \neg New\text{-}B \leftarrow not\ New\text{-}B,$

$N_{11} : New\text{-}A \leftarrow New\text{-}C,$

$<$:

$N_4 < N_{10}, N_5 < N_{10}, N_6 < N_{10},$

$N_7 < N_{10}, N_8 < N_{10}, N_9 < N_{10},$

$N_4 < N_{11}, N_5 < N_{11}, N_6 < N_{11},$

$N_7 < N_{11}, N_8 < N_{11}, N_9 < N_{11}.$

Now from Definitions 2 and 3, it is not difficult to see that $Update(\mathcal{B}, \Pi)$ has a unique answer set: $\{\neg A, B, C, New\text{-}A, New\text{-}B, New\text{-}C\}$. Note that in $Update(\mathcal{B}, \Pi)$, only $N_5 < N_{10}$ is used in $Update(\mathcal{B}, \Pi)$'s evaluation, while other $<$-relations are useless (see Definition 2). Hence, from Definition 5, the only resulting knowledge base $\mathcal{B}'$ after updating $\mathcal{B}$ with Π is: $\{A, B, C\}$ ■

It is worth to mention that the major difference between our approach and other formulations of rule based update is that our approach can deal with a more general case by allowing the set of update rules to have both classical negation and negation as failure, while in previous methods, an update rule can only have classical negation, e.g. [1]. It is also observed that by allowing an update rule to have two types of negations, previous methods may result in some unintuitive solution. For instance, in the case of Example 2 described above, Przymusinski and Turner's method will imply a solution that literal B may or may not be changed after updating $\{\neg A, B, C\}$ with $\{\neg B \leftarrow not\ B,\ A \leftarrow C\}$.

3 Restricted Monotonicity Properties

In this section, we examine basic properties, specifically, monotonicity properties related to the generalized rule based update described above. Firstly we show that the update specification $Update(\mathcal{B}, \Pi)$ in language $\mathcal{L}^P_{new}$ defined in Definition 4 can be simplified to a PLP in language $\mathcal{L}^P$.

Lemma 1 *Let $Update(\mathcal{B}, \Pi)$ be a well defined update specification as defined in Definition 4. $\mathcal{B}'$ is a resulting knowledge base after updating $\mathcal{B}$ with Π if and only if $\mathcal{B}'$ is an answer set of prioritized logic program $\mathcal{P} = (\Pi \cup \{L \leftarrow not\overline{L} \mid L \in \mathcal{B}\}, \mathcal{N}, <)$, where for each rule $r : L \leftarrow not\overline{L}$ with $L \in \mathcal{B}$, and each rule r' in Π, $\mathcal{N}(r) < \mathcal{N}(r')$*[4].

Consider a knowledge base $\mathcal{B}$ and a rule r with the form (1). Recall that $\mathcal{B}$ *satisfies* r iff if facts $L_1, \cdots, L_m$ are in $\mathcal{B}$ and facts $L_{m+1}, \cdots$, and L_n are not in $\mathcal{B}$, then fact L_0 is in $\mathcal{B}$. Let Π be a set of rules with the form (1). $\mathcal{B}$ satisfies Π if $\mathcal{B}$ satisfies each rule in Π. We show that after

[4] $\overline{L}$ stands for the complement of literal L.

updating knowledge base $\mathcal{B}$ with Π, every possible resulting knowledge base $\mathcal{B}'$ satisfies Π as stated in the following proposition.

Proposition 1 *Given a knowledge base $\mathcal{B}$ and an update program Π. Suppose the update specification $Update(\mathcal{B}, \Pi)$ is well defined. Let $\mathcal{B}'$ be a resulting knowledge base with respect to $Update(\mathcal{B}, \Pi)$. Then $\mathcal{B}'$ satisfies Π.*

Let $\mathcal{B}$ and $\mathcal{B}'$ be two knowledge bases. We use $Diff(\mathcal{B}, \mathcal{B}')$ to denote the symmetric set difference on ground atoms between $\mathcal{B}$ and $\mathcal{B}'$, i.e.

$$Diff(\mathcal{B}, \mathcal{B}') = \{|L| \mid L \in (\mathcal{B} - \mathcal{B}') \cup (\mathcal{B}' - \mathcal{B})\},$$

where notation $|L|$ indicates the corresponding ground atom of ground literal L, and $Min(\mathcal{B}, \Pi)$ to denote the set of all consistent knowledge bases satisfying Π but with minimal differences from $\mathcal{B}$, i.e.

$$Min(\mathcal{B}, \Pi) = \{\mathcal{B}' \mid \mathcal{B}' \text{ satisfies } \Pi \text{ and } Diff(\mathcal{B}, \mathcal{B}') \text{ is minimal with respect to set inclusion}\}.$$

Then we have the following minimal change theorem for update, which guarantees that our generalized rule based update satisfies the principle of minimal change.

Theorem 1 *(**Minimal Change**) Let $\mathcal{B}$ be a knowledge base, Π an update program, and $Update(\mathcal{B}, \Pi)$ the well defined update specification as defined in Definition 4. If $\mathcal{B}'$ is a resulting knowledge base with respect to $Update(\mathcal{B}, \Pi)$, then $\mathcal{B}' \in Min(\mathcal{B}, \Pi)$.*

The above theorem guarantees that our generalized rule based update satisfies the principle of minimal change. However, it should be noted that not every element of $Min(\mathcal{B}, \Pi)$ could be a resulting knowledge base of updating $\mathcal{B}$ by $\mathcal{P}$.

It is not surprising that in general our rule based update is nonmonotonic in the sense that adding more facts into a knowledge base $\mathcal{B}$ or more update rules into an update program Π may destroy some facts obtained from the update of $\mathcal{B}$ with Π. Therefore, instead of obtaining a general monotonicity result, our goal here is to investigate some kinds of restricted monotonicity properties related to our rule based update. In particular, we are interested in two forms of restricted monotonicity properties: (1) under what conditions will the result of updating a knowledge base $\mathcal{B}$ with Π be preserved in the result of updating an expanded knowledge base $\mathcal{B} \cup \mathcal{B}'$ with Π? (2) under what conditions will the result of updating a knowledge base $\mathcal{B}$ with Π be preserved in the result of updating $\mathcal{B}$ with an expanded update program $\Pi \cup \Pi'$?

Before we present our restricted monotonicity theorems, we first introduce some useful notations. Let r be a rule in Π:

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

We use $pos(r)$ to denote the set of literals in the body of r without negation as failure $\{L_1, \cdots,$
$L_m\}$ in r, and $neg(r)$ the set of literals in the body of r with negation as failure $\{L_{m+1}, \cdots, L_n\}$ in r. 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 Π, e.g. $body(\Pi) = \bigcup_{r \in \Pi} body(r)$. Given a set of literals $\mathcal{B}$, we also use $\overline{\mathcal{B}}$ to denote the set of complement literals of $\mathcal{B}$ with respect to classical negation sign $\neg$.

Theorem 2 *(***Restricted Monotonicity Theorem 1***) Given two knowledge bases $\mathcal{B}_1$ and $\mathcal{B}_2$ where $\mathcal{B}_1 \subseteq \mathcal{B}_2$ and an update program Π. Suppose both update specifications $Update(\mathcal{B}, \Pi_1)$ and $Update(\mathcal{B}, \Pi_2)$ are well defined. Let $\mathcal{B}'_1$ be a resulting knowledge base with respect to $Update(\mathcal{B}_1, \Pi)$. Then there exists a resulting knowledge base $\mathcal{B}'_2$ with respect to $Update(\mathcal{B}_2, \Pi)$ such that $\mathcal{B}'_1 \subseteq \mathcal{B}'_2$ if $body(\Pi) \cap (\mathcal{B}_2 - \mathcal{B}_1) = \emptyset$. In this case, $\mathcal{B}'_2 = \mathcal{B}'_1 \cup \{L \mid L \in (\mathcal{B}_2 - \mathcal{B}_1)$ and $\overline{L} \notin \mathcal{B}'_1\}$.*

Theorem 3 *(***Restricted Monotonicity Theorem 2***) Given a knowledge base $\mathcal{B}$ and two update programs Π_1 and Π_2 where $\Pi_1 \subseteq \Pi_2$. Suppose both update specifications $Update(\mathcal{B}_1, \Pi)$ and $Update(\mathcal{B}_2, \Pi)$ are well defined. Let $\mathcal{B}'$ be a resulting knowledge base with respect to $Update(\mathcal{B}, \Pi_1)$. Then there exists a resulting knowledge base $\mathcal{B}''$ with respect to $Update(\mathcal{B}, \Pi_2)$ such that $\mathcal{B}' \subseteq \mathcal{B}''$ if $head(\Pi_2 - \Pi_1) \cap (\overline{\mathcal{B}} \cup body(\Pi_1)) = \emptyset$. In this case, $\mathcal{B}''$ is an answer set of program $\{L \leftarrow \mid L \in \mathcal{B}'\} \cup (\Pi_2 - \Pi_1)$.*

The intuitive idea behind Theorem 2 is described as follows. If a knowledge base $\mathcal{B}_1$ is expanded to $\mathcal{B}_2$ by adding some facts and all these added facts do not occur in the body of any rule in Π, then the result of updating $\mathcal{B}_1$ with Π is preserved in the result of updating $\mathcal{B}_2$ with Π. Furthermore, the latter can be simply computed from the result of updating $\mathcal{B}_1$ with Π. On the other hand, Theorem 3 says that if an update program Π_1 is expanded to Π_2 by adding some rules and the head of each added rule does not occur in the bodies of rules in update specification $Update(\mathcal{B}, \Pi_1)$, then the result of updating $\mathcal{B}$ with Π_1 is preserved in the result of updating $\mathcal{B}$ with Π_2, and the latter is reduced to an answer set of a corresponding extended logic program.

Given a knowledge base $\mathcal{B}$ and an update program Π, we can apply the restricted monotonicity theorems above to simplify the computation of $Update(\mathcal{B}, \Pi)$ when $\mathcal{B}$ or Π can be split into parts. The following example illustrates such application.

Example 3 Let $\mathcal{B} = \{A, B, C, D\}$, and $\Pi = \{\neg A \leftarrow B, \neg C \leftarrow B, \neg B \leftarrow not\ B\}$. We consider the update of $\mathcal{B}$ with Π. Since $body(\Pi) \cap \{A, C, D\} = \emptyset$, from Theorem 2, we can actually reduce the update of $\mathcal{B}$ with Π into the

update of $\{B\}$ with Π. It is clear that the unique result of $Update(\{B\}, \Pi)$ is $\{\neg A, B, \neg C\}$. So according to Theorem 2, the unique resulting knowledge base with respect to $Update(\mathcal{B}, \Pi)$ is $\{\neg A, B, \neg C, D\}$.

Let us consider another situation. Let $\mathcal{B} = \{A, B\}$, and $\Pi = \{\neg B \leftarrow not\ C, \neg C \leftarrow A\}$. Consider a rule $r : \neg C \leftarrow A$ in Π. Since $head(r) \cap (\overline{\mathcal{B}} \cup body(\neg B \leftarrow not\ C)) = \{\neg C\} \cap \{\neg A, \neg B, C\} = \emptyset$, according to Theorem 3, the update of $\mathcal{B}$ with Π can be reduced to the update of $\mathcal{B}$ with $\{\neg B \leftarrow not\ C\}$, which has the unique result $\{A, \neg B\}$. So the result of $Update(\mathcal{B}, \Pi)$ is an answer set of program $\{A \leftarrow, \neg B \leftarrow, \neg C \leftarrow A\}$, which has the unique answer set $\{A, \neg B, \neg C\}$. ■

4 A Generalization of Splitting Theorem

Our proofs of Theorems 2 and 3 described previously are based on a non-trivial generalization of Lifschitz-Turner's Splitting Set Theorem on extended logic programs [4]. To describe Lifschitz-Turner's Splitting Theorem , we first define the following transformation. Let X be a set of literals and Π an extended logic program. 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 *not* L in its body with $L \in X$; and
2. all formulas of form L in the bodies of the remaining rules with $L \in X$.

Consider an example with $X = \{C\}$ and Π is a set of the following rules:

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

Then it is clear that $e(\Pi, \{C\}) = \{B \leftarrow not\ A\}$. Intuitively, e-reduct of Π with respect to X can be viewed as a simplied program of Π given the fact that every literal in X is true. Now Lifschitz-Turner's Splitting Set Theorem on extended logic program can be presented as follows[5].

Lifschitz and Turner's Splitting Set Theorem [4]
Let $\Pi = \Pi_1 \cup \Pi_2$ be an extended logic program and $lit(\Pi_1) \cap head(\Pi_2) = \emptyset$. Then a set of literals $\mathcal{B}$ is a consistent answer set of Π if and only if $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}'$, where $\mathcal{B}_1$ is an answer set of Π_1, $\mathcal{B}'$ is an answer set of $e(\Pi_2, \mathcal{B}_1)$, and $\mathcal{B}_1 \cup \mathcal{B}'$ is consistent.

As illustrated by Lifschitz and Turner, with the splitting theorem, computing the answer set of an extended logic program can be simplified when the program is split into parts. In the above example, if we expand Π as $\Pi \cup \{C \leftarrow\}$, we can rewrite Π as $\Pi_1 \cup \Pi_2$, where $\Pi_1 = \{C \leftarrow\}$ and $\Pi_2 = \{A \leftarrow B,\ not\ C,\ B \leftarrow C,\ not\ A\}$. Clearly, $lit(\Pi_1) \cap head(\Pi_2) = \emptyset$. So

[5]For our purpose, we slightly reformulate this theorem according to Lifschitz and Turner's original form.

it is easy to see that $\{C, B\}$ is an answer set of $\Pi \cup \{C \leftarrow\}$. This splitting theorem, however, can be further generalized. Let us consider the following example.

Example 4 Let Π be a program consisting of following rules:

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

It is not difficult to see that Π can not be split into two parts Π_1 and Π_2 such that $lit(\Pi_1) \cap head(\Pi_2) = \emptyset$ if we do not consider the trivial case by setting $\Pi_1 = \emptyset$ or $\Pi_2 = \emptyset$. Therefore, the above splitting theorem cannot be used to compute the answer set of Π. However, 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 $\{A\}$. ■

From the above observation, we can see that since the head of each rule in Π_2 does not occur in the body of each rule in Π_1, Π_2 actually does not play any role in affecting rules in Π_1 when we compute the answer set of Π. Therefore, Π can be still split into two parts to evaluate its answer set. In general, we have the following generalized splitting theorem.

Theorem 4 *(***Generalized Splitting Theorem***) Let $\Pi = \Pi_1 \cup \Pi_2$ be an extended logic program and $body(\Pi_1) \cap head(\Pi_2) = \emptyset$. Then a set of literals $\mathcal{B}$ is a consistent answer set of Π if and only if $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}'$, where $\mathcal{B}_1$ is an answer set of Π_1, $\mathcal{B}'$ is an answer set of $e(\Pi_2, \mathcal{B}_1)$, and $\mathcal{B}_1 \cup \mathcal{B}'$ is consistent.*

Proof: ($\Leftarrow$) Let $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}'$ and $\Pi = \Pi_1 \cup \Pi_2$, where $\mathcal{B}_1$ is an answer set of Π_1 and $\mathcal{B}'$ is an answer set of $e(\Pi_2, \mathcal{B}_1)$ and $\mathcal{B}_1 \cup \mathcal{B}'$ is consistent. Consider the Gelfond-Lifschitz transformation of Π in terms of $\mathcal{B}$, $\Pi^{\mathcal{B}}$. $\Pi^{\mathcal{B}}$ 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 $L \in \mathcal{B}$; 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 $L \in \mathcal{B}_1$, only rules with the form $L' \leftarrow \cdots$, *not* L in Π_1 or Π_2 will be deleted. On the other hand, for each literal $L \in \mathcal{B}'$, only rules with the form $L' \leftarrow \cdots$, *not* L in Π_2 will be deleted and no rules in Π_1 can be deleted. Therefore, we can denote $\Pi^{\mathcal{B}}$ as $\Pi_1' \cup \Pi_2'$, where Π_1' is obtained from Π_1 in terms of literals in

$\mathcal{B}_1$, and Π'_2 is obtained from Π_2 in terms of literals in $\mathcal{B}_1 \cup \mathcal{B}'$ during the above transformation procedure. Then it is easy to see that $\Pi'_1 = \Pi_1^{\mathcal{B}_1}$. So $\mathcal{B}_1$ is an answer set of Π'_1.

On the other hand, from the construction of $e(\Pi_2, \mathcal{B}_1)$, it is observed that there exists the following correspondence between Π'_2 and $e(\Pi_2, \mathcal{B}_1)$: for each rule $L_0 \leftarrow L_1, \cdots, L_k, L_{k+1}, \cdots, L_m$ in Π'_2, there is a rule with the form $L_0 \leftarrow L_1, \cdots, L_k$, *not* $L_{m+1}, \cdots$, *not* L_n in $e(\Pi_2, \mathcal{B}_1)$ such that $L_{k+1}, \cdots, L_m \in \mathcal{B}_1$ and $L_{m+1}, \cdots$, or $L_n \notin \mathcal{B}_1$; on the other hand, for each rule $L_0 \leftarrow L_1, \cdots, L_k$, *not* $L_{m+1}, \cdots$, *not* L_n in $e(\Pi_2, \mathcal{B}_1)$, if none of $L_{m+1}, \cdots, L_n$ is in $\mathcal{B}'$, then there exists a rule $L_0 \leftarrow L_1, \cdots, L_k, L_{k+1}, \cdots, L_m$ in Π'_2 such that $L_{k+1}, \cdots, L_m \in \mathcal{B}_1$. From this observation, it can be seen that there exists a subset Δ of $\mathcal{B}_1$ such that $\Delta \cup \mathcal{B}'$ is an answer set of Π'_2. This follows that $\mathcal{B}_1 \cup \mathcal{B}'$ is the smallest set such that for each rule $L_0 \leftarrow L_1, \cdots, L_m \in \Pi^{\mathcal{B}}$, $L_1, \cdots, L_m \in \mathcal{B}$ implies $L_0 \in \mathcal{B}$. That is, $\mathcal{B}$ is an answer set of $\Pi^{\mathcal{B}}$ and also an answer set of Π.

($\Rightarrow$) Let $\Pi = \Pi_1 \cup \Pi_2$ and $\mathcal{B}$ be a consistent answer set of Π. It is clear that for each literal $L \in \mathcal{B}$, there must be some rule with the form $L \leftarrow \cdots$ in Π. So we can write $\mathcal{B}$ as a form of $\mathcal{B}'_1 \cup \mathcal{B}'_2$ such that $\mathcal{B}'_1 \subseteq \mathit{head}(\Pi_1)$ and $\mathcal{B}'_2 \subseteq \mathit{head}(\Pi_2)$. Note that $\mathcal{B}'_1 \cap \mathcal{B}'_2$ may not be empty. Now we transfer set $\mathcal{B}'_1$ into $\mathcal{B}_1$ by the following step: if $\mathcal{B}'_1 \cap \mathcal{B}'_2 = \emptyset$, then $\mathcal{B}_1 = \mathcal{B}'_1$; otherwise, let $\mathcal{B}_1 = \mathcal{B}'_1 - \{L \mid L \in \mathcal{B}'_1 \cap \mathcal{B}'_2$, and for each rule $L \leftarrow L_1, \cdots, L_m$, $\mathit{not}L_{m+1}, \cdots$, $\mathit{not}L_n$ in Π_1, there exists some L_j $(1 \leq j \leq m) \notin \mathcal{B}'_1$ or L_j $(m+1 \leq j \leq n) \in \mathcal{B}'_1\}$. In above translation, since every L deleted from $\mathcal{B}_1$ is also in $\mathcal{B}'_2$, the answer set $\mathcal{B}$ of Π can then be expressed as $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}'_2$. An important fact is observed from the construction of $\mathcal{B}_1$:

Fact 1. $L \in \mathcal{B}_1$ iff there exists some rule in Π_1 with the form $L \leftarrow L_1, \cdots, L_m$, $\mathit{not}L_{m+1}, \cdots$, $\mathit{not}L_n$, such that $L_1, \cdots, L_m \in \mathcal{B}_1$ and $L_{m+1}, \cdots$, or $L_n \notin \mathcal{B}_1$.

Now we prove $\mathcal{B}_1$ is an answer set of Π_1. We do Gelfond-Lifschitz transformation on Π in terms of set $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}'_2$. After such transformation, we can write $\Pi^{\mathcal{B}}$ as form $\Pi'_1 \cup \Pi'_2$, where $\Pi'_1 \subseteq \Pi_1$ and $\Pi'_2 \subseteq \Pi_2$. As $\mathit{body}(\Pi_1) \cap \mathit{head}(\Pi_2) = \emptyset$, any literal in $\mathcal{B}'_2$ will not cause a deletion of a rule from Π_1 in the Gelfond-Lifschitz transformation. Then it is easy to see that $\Pi'_1 = \Pi_1^{\mathcal{B}_1}$. From **Fact 1**, it concludes that literal $L \in \mathcal{B}_1$ iff there is a rule $L \leftarrow L_1, \cdots L_m$ in $\Pi_1^{\mathcal{B}_1}$ and $L_1, \cdots, L_m \in \mathcal{B}_1$. This follows that $\mathcal{B}_1$ is an answer set of $\Pi_1^{\mathcal{B}_1}$, and then an answer set of Π_1.

Now we transfer $\mathcal{B}'_2$ into $\mathcal{B}_2$ by the following step: if $\mathcal{B}_1 \cap \mathcal{B}'_2 = \emptyset$, then $\mathcal{B}_2 = \mathcal{B}'_2$; otherwise, let $\mathcal{B}_2 = \mathcal{B}'_2 - \{L \mid L \in \mathcal{B}_1 \cap \mathcal{B}'_2$, and for each rule $L \leftarrow L_1, \cdots, L_m$, *not* $L_{m+1}, \cdots$, *not* L_n in Π_2, there exists some $L_j (1 \leq j \leq m) \notin \mathcal{B}_1 \cup \mathcal{B}'_2$, or $L_j (m+1 \leq j \leq n) \in \mathcal{B}_1 \cup \mathcal{B}'_2\}$. After this translation, $\mathcal{B}$ can be expressed as $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$. An important fact is also observed from the translation of $\mathcal{B}_2$:

Fact 2. $L \in \mathcal{B}_2$ iff there exists some rule in Π'_2 with the form $L \leftarrow L_1, \cdots, L_k, L_{k+1}, \cdots, L_m$ such that $L_1, \cdots, L_k \in \mathcal{B}_1$ and $L_{k+1}, \cdots, L_m \in \mathcal{B}_2$.

Now we prove $\mathcal{B}_2$ is an answer set of $e(\Pi_1, \mathcal{B}_1)$. Recall that $\Pi^{\mathcal{B}} = \Pi'_1 \cup \Pi'_2 = \Pi_1^{\mathcal{B}_1} \cup \Pi'_2$. From **Fact 2**, it is clear that there exists a subset Δ of $\mathcal{B}_1$ such that $\mathcal{B}_2$ is an answer set of $e(\Pi'_2, \Delta)$ and $e(\Pi'_2, \Delta)^{\mathcal{B}_2} = e(\Pi'_2, \Delta)$. On the other hand, from the construction of $e(\Pi_2, \mathcal{B}_1)$, it is easy to see that $e(\Pi_2, \mathcal{B}_1)^{\mathcal{B}_2} = e(\Pi'_2, \Delta) = e(\Pi'_2, \Delta)^{\mathcal{B}_2}$. So $\mathcal{B}_2$ is also an answer set of $e(\Pi_2, \mathcal{B}_1)$. ■

5 Proofs of Restricted Monotonicity Theorems

From the Generalized Splitting Theorem (Theorem 4), we can prove the following Lemma 2 which will be needed in the proof of Restricted Monotonicity Theorem 1 (Theorem 2), and from Lemma 3 we can prove Restricted Monotonicity Theorem 2 (Theorem 3). Due to a space limit, details of the proofs for these lemmas are referred to our full paper [9]. Here we only give proofs of Theorems 2 and 3.

Lemma 2 *Let Π_1 and Π_2 be two extended logic programs, where each rule in Π_2 has form $L \leftarrow not\overline{L}$, $head(\Pi_2) \cap body(\Pi_2) = \emptyset$ and $body(\Pi_1) \cap head(\Pi_2) = \emptyset$. Suppose both Π_1 and $\Pi_1 \cup \Pi_2$ have consistent answer sets. Then $\mathcal{B}$ is a consistent answer set of $\Pi_1 \cup \Pi_2$ if and only if $\mathcal{B} = \mathcal{B}' \cup \{L \mid L \leftarrow not\overline{L} \in \Pi_2$ and $\overline{L} \notin \mathcal{B}'\}$, where $\mathcal{B}'$ is a consistent answer set of Π_1.*

Lemma 3 *Let $\mathcal{P} = (\Pi, \mathcal{N}, <)$ be a PLP which has consistent answer set(s). Then each answer set of $\mathcal{P}$ is also an answer set of Π.*

Proof of Restricted Monotonicity Theorem 1
From Lemma 1, $Update(\mathcal{B}_1, \Pi)$ and $Update(\mathcal{B}_2, \Pi)$ are equivalent to following two PLPs respectively: $\mathcal{P}_1 = (\Pi \cup \{L \leftarrow not\overline{L} \mid L \in \mathcal{B}_1\}, \mathcal{N}, <)$ and $\mathcal{P}_2 = (\Pi \cup \{L \leftarrow not\overline{L} \mid L \in \mathcal{B}_2\}, \mathcal{N}, <)$, where $<$ relations in $\mathcal{P}_1$ are specified as stated in Lemma 1 respectively. Let $\mathcal{B}'_1$ be a consistent answer set of $\mathcal{P}_1$. We assume $\mathcal{B}_1 = \{L_1, \cdots, L_k\}$, and $\mathcal{B}_2 = \mathcal{B}_1 \cup \{L_{k+1}, \cdots, L_m\}$. Observing the construction of $\mathcal{P}_2$, besides rules in $\mathcal{P}_1$, $\mathcal{P}_2$ also contains following rules: $r_{k+1} : L_{k+1} \leftarrow not\overline{L_{k+1}}$, $\cdots$, $r_m : L_m \leftarrow not\overline{L_m}$. Let $\Pi' \cup \Pi''$ be a reduct of $\mathcal{P}_2$, where each rule in Π' is also in $\mathcal{P}_1$, and $\Pi'' = \{r_{k+1}, \cdots, r_m\}$. Note that since for each rule $r \in \Pi''$, there is no other rule r^* such that $\mathcal{N}(r^*) < \mathcal{N}(r)$, all rules $r_{k+1}, \cdots, r_m$ will be included in each reduct of $\mathcal{P}_2$. Now we show $\mathcal{B}'_2$ is an answer set of $\Pi' \cup \Pi''$. If Π' is a reduct of $\mathcal{P}_1$, then $\mathcal{B}'_1$ is an answer set of Π' and the result is true from Lemma 2.

Now suppose Π' is a proper subset of some reduct Π^* of $\mathcal{P}_1$, where $\Pi' = \Pi^* - \{r_p, \cdots, r_q\}$. Clearly, all rules $r_p, \cdots, r_q$ are eliminated from Π^* due to additional rules $r_{k+1}, \cdots, r_m$ are added into $\mathcal{P}_1$ to form $\mathcal{P}_2$. Therefore, we can assume that in the evaluation of reduct $\Pi' \cup \Pi''$ of $\mathcal{P}_2$, there exists some integer h such that

$$\Pi_0 = \Pi \cup \{L \leftarrow not\overline{L} \mid L \in \mathcal{B}_2\},$$
$$\cdots,$$

$$\Pi_h = \Pi^* \cup \Pi'',$$
$$\Pi_{h+1} = \Pi_h - \{r_i, \cdots, r_j \mid \text{there exists some } r \in \Pi'' \text{ such that } \mathcal{N}(r) < \mathcal{N}(r_i), \cdots, \mathcal{N}(r) < \mathcal{N}(r_j) \text{ and } r_i, \cdots, r_j \text{ are defeated by } \Pi_h - \{r_i, \cdots, r_j\}\},$$
$$\cdots,$$

where $r_i, \cdots, r_j$ are the first set of rules eliminated from the reduct Π^* of $\mathcal{P}_1$ due to preferences $\mathcal{N}(r) < \mathcal{N}(r_i)$, $\cdots$, $\mathcal{N}(r) < \mathcal{N}(r_j)$ for some $r \in \Pi''$. Note that since Π^* is a reduct of $\mathcal{P}_1$, no any other rules in Π^* can be further eliminated from preferences between rules in $\{L \leftarrow not \neg L \mid L \in \mathcal{B}_1\}$ and rules in Π. On the other hand, since for any rule $r' \in \{L \leftarrow not\, \overline{L} \mid L \in \mathcal{B}_1\}$, $\mathcal{N}(r') < \mathcal{N}(r_i)$, $\cdots$, $\mathcal{N}(r') < \mathcal{N}(r_j)$, Π_{h+1} can be also specified as

$$\Pi_{h+1} = \Pi_h - \{r_i, \cdots, r_j \mid \text{there exists some } r' \in \{L \leftarrow not \overline{L} \mid L \in \mathcal{B}_1\} \text{ such that } \mathcal{N}(r') < \mathcal{N}(r_i), \cdots, \mathcal{N}(r') < \mathcal{N}(r_j) \text{ and } r_i, \cdots, r_j \text{ are defeated by } \Pi_h - \{r_i, \cdots, r_j\}\}.$$

But this contradicts the fact that Π^* is a reduct of $\mathcal{P}_1$ where no any other rules can be further eliminated from preferences between rule r' and rules $r_i, \cdots, r_j$. So Π' must be a reduct of $\mathcal{P}_1$. ■

Proof of Restricted Monotonicity Theorem 2
From Lemma 1, we know that $Update(\mathcal{B}, \Pi_2)$ is equivalent to a PLP $\mathcal{P} = (\{L \leftarrow not \overline{L} \mid L \in \mathcal{B}\} \cup \Pi_2, \mathcal{N}, <)$, where for each rule $r \in \{L \leftarrow not \overline{L} \mid L \in \mathcal{B}\}$ and each rule and each rule r' in Π_2, $N(r) < N(r')$. Then from Lemma 3, we know that each answer set of $\mathcal{P}$ is also an answer set of extended logic program $\{L \leftarrow not \overline{L} \mid L \in \mathcal{B}\} \cup \Pi_2$. Again from Lemmas 1 and 3, a resulting knowledge base with respect to $Update(\mathcal{B}, \Pi_1)$ can be viewed as an answer set of program $\{L \leftarrow not \overline{L} \mid L \in \mathcal{B}\} \cup \Pi_1$. Therefore, it is sufficient to prove that each consistent answer set $\mathcal{B}''$ of Π', where $\Pi' = \{L \leftarrow not \overline{L} \mid L \in \mathcal{B}\} \cup \Pi_2$, is also an answer set of $\{L \leftarrow \mid L \in \mathcal{B}'\} \cup (\Pi_2 - \Pi_1)$, where $\mathcal{B}'$ is an answer set of $\{L \leftarrow not \overline{L} \mid L \in \mathcal{B}\} \cup \Pi_1$. From condition $head(\Pi_2 - \Pi_1) \cap (\overline{\mathcal{B}} \cup body(\Pi_1)) = \emptyset$, this is simply followed from the Generalized Splitting Theorem (Theorem 4). ■

6 Conclusion

In this paper we analyzed restricted monotonicity properties for rule based update. In particular, we provided two forms of restricted monotonicity theorems. Example 3 presented in section 3 illustrated an application of how these results are used to simplify update evaluations. In general, given an update specification $Update(\mathcal{B}, \Pi)$, two restricted monotonicity theorems can be applied alternatively and sequentially to split both $\mathcal{B}$ and Π into smaller and smaller parts such that the evaluation of $Update(\mathcal{B}, \Pi)$ can be significantly simplified (details have been shown in [9]).

Another contribution of this paper is that we proved a generalization of Lifschitz-Turner's Splitting Set Theorem. As Lifschitz-Turner's Splitting Set Theorem has illustrated its broad applications in exploring declarative semantics of logic programs, logic program based knowledge representation, and reasoning about action, e.g. [7], it is clear that our generalized splitting theorem will enhance these applications under an even weaker condition.

Acknowledgement

The author thanks Vladimir Lifschitz for examining the author's early version of the proof of the generalized splitting theorem. Thanks are also due to four anonymous referees for their criticisms and useful comments.

References

[1] C. Baral, Rule based updates on simple knowledge bases. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-94)*, pp 136-141. AAAI Press, 1994.

[2] J. Engelfriet, Monotonicity and persistence in preferential logics. *Journal of Artificial Intelligence Research*, **8**: 1-12, 1998.

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

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

[5] V.W. Marek and M. Truszczyński, Update by means of inference rules. In *Proceedings of JELIA'94, Lecture Notes in Artificial Intelligence*, 1994.

[6] T.C. Przymusinski and H. Turner, Update by means of inference rules. In *Proceedings of LPNMR'95, Lecture Notes in Artificial Intelligence*, pp 156-174. Springer-Verlag, 1995.

[7] H. Turner, Representing actions in logic programs and default theories: A situation calculus approach. *Journal of Logic Programming*, **31**, 1997.

[8] Y. 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

[9] Y. Zhang, A framework of rule based update. Manuscript, April, 1999.