Minimal Change and Maximal Coherence for Epistemic Logic Program Updates

Yan Zhang
School of Computing & IT
University of Western Sydney
Penrith South DC, NSW 1797, Australia
E-mail: yan@cit.uws.edu.au

Abstract

We consider the problem of updating nonmonotonic knowledge bases represented by epistemic logic programs where disjunctive information and notions of knowledge and beliefs can be explicitly expressed. We propose a formulation for epistemic logic program updates based on a principle called minimal change and maximal coherence. The central feature of our approach is that during an update procedure, contradictory information is removed on a basis of minimal change under the semantics of epistemic logic programs and then coherent information is maximally retained in the update result. By using our approach, we can characterize an update result in both semantic and syntactic forms. We show that our approach handles update sequences and satisfies the consistency requirement. We also investigate important semantic properties of our update approach such as reduction, persistence and preservation.

1 Introduction

Logic programming has been proved to be one of the most promising logic based formulations for problem solving, knowledge representation and reasoning, and reasoning about actions and plans. Recent research on logic program updates further shows that logic programming also provides a feasible framework for modeling agents' activities in dynamic environments [Alferes and *et al,* 2000; Eiter and *et al,* 2002; Sakama and Inoue, 1999; Zhang and Foo, 1998].

While all current approaches for logic program updates focus on the problem of updating extended logic programs or their variations, updating *epistemic logic programs,* however, has yet to be explored in the research. By combining *knowledge* and *belief* operators into logic rules, epistemic logic programming [Gelfond, 1994] is a powerful representation formalism in logic programming paradigm. It can deal with more difficult problems in reasoning with disjunctive information while traditional disjunctive extended logic programs fail to handle. Furthermore, epistemic logic programs seem more feasible for knowledge reasoning than many other autoepistemic logics [Gelfond, 1994] and has been used as a formal basis for modeling knowledge in action theories, e.g. [Lobo *et al,* 2001]. When we use an epistemic logic program to represent an agent's knowledge base, it is a nontrivial question how the agent's knowledge base (an epistemic logic program) can be updated when new information is received.

In this paper, we propose an approach for epistemic logic program updates. Contrary to other logic programs, notions of knowledge and beliefs in epistemic logic programs have strong semantic connections to the standard Kripke structures of modal logics. On the other hand, epistemic logic programs are also sensitive with various syntactic forms. Hence, we believe that a pure model-based or syntax-based approach will not be appropriate to handle epistemic logic program updates. Instead, we require our update formulation to meet three major criteria: (1) an update should be performed on a basis of minimal change semantics to remove contradictory information; (2) based on the minimal change semantics, the update result should have a clear syntactic representation and contain maximal consistent information from previous prograntS); and (3) the underlying update procedure should be consistent, that is, updating a consistent program by another consistent program (or a sequence of consistent programs) should generate a consistent result. Our main idea to accomplish these criteria is so called *minimal change and maximal coherence* which presents both semantic and syntactic features in an update procedure.

The paper is organized as follows. Section 2 presents a brief overview on epistemic logic programs. Section 3 develops a formulation for epistemic logic program updates, while section 4 extends this formulation to handle update sequences. Section 5 investigates important semantic properties for our update approach. Finally, section 6 concludes the paper with discussions on related work and future research.

2 Epistemic Logic Programs: An Overview

In this section, we present a general overview on epistemic logic programs. Gelfond extended the syntax and semantics of disjunctive logic programs to allow the correct representation of incomplete information (knowledge) in the presence of multiple extensions. Consider the following disjunctive program about the policy of offering scholarships in some university [Gelfond, 1994]:

V:
r_1: *eligible(x)* <- *highGPA(x),*
r_2: *eligible(x)* <- *minority(x)*, *fairGPA*(x),

r_3: $\neg eligible(x) \leftarrow \neg fairGPA(x),$
$\neg highGPA(x),$
r_4: $interview(x) \leftarrow not\ eligible(x),$
$not\ \neg eligible(x),$
r_5: $fairGPA(mike)\ or\ highGPA(mike) \leftarrow.$

Rule r_4 can be viewed as a formalization of the statement: "the sutdents whose eligibility is not decided by rules r_1, r_2 and r_3 should be interviewed by the committee". It is easy to see that V has two answer sets *{highGPA(mike), eligible (mike)}* and *{fair GPA(mike), interview(mike)}*. Therefore the answer to query *interview (mike)* is *unknown,* which seems too weak from our intuition. Epistemic logic programs will overcome this kind of difficulties in reasoning with incomplete information.

In epistemic logic programs, the language of (disjunctive) extended logic programs is expanded with two modal operators K and M. KF is read as "F is known to be true" and MF is read as "F may be believed to be true". For our purpose, in this paper we will only consider propositional epistemic logic programs where rules containing variables are viewed as the set of all ground rules by replacing these variables with all constants occurring in the language. The semantics for epistemic logic programs is defined by pairs $(\mathcal{A}, W)$, where $\mathcal{A}$ is a collection of sets of ground literals called the sets of *possible beliefs* of certain agent, while W is a set in $\mathcal{A}$ called the agent's *working set of beliefs.* The truth of a formula F in $(\mathcal{A}, W)$ is denoted by $(\mathcal{A}, W) \models F$ and the falsity by $(\mathcal{A}, W) =| F$, and are defined as follows:
$(\mathcal{A}, W) \models F$ iff $F \in W$ where F is a ground atom.
$(\mathcal{A}, W) \models KF$ iff $(\mathcal{A}, W_i) \models F$ for all $W_i \in \mathcal{A}$.
$(\mathcal{A}, W) \models MF$ iff $(\mathcal{A}, W_i) \models F$ for some $W_i \in \mathcal{A}$.
$(\mathcal{A}, W) \models F \wedge G$ iff $(\mathcal{A}, W) \models F$ and $(\mathcal{A}, W) \models G$.
$(\mathcal{A}, W) \models F\ or\ G$ iff $(\mathcal{A}, W) \models \neg(\neg F \wedge \neg G)$.
$(\mathcal{A}, W) \models \neg F$ iff $(\mathcal{A}, W) =| F$.
$(\mathcal{A}, W) =| F$ iff $\neg F \in W$ where F is a ground atom.
$(\mathcal{A}, W) =| KF$ iff $(\mathcal{A}, W) \not\models KF$.
$(\mathcal{A}, W) =| MF$ iff $(\mathcal{A}, W) \not\models MF$.
$(\mathcal{A}, W) =| F \wedge G$ iff $(\mathcal{A}, W) =| F$ or $(\mathcal{A}, W) =| G$.
$(\mathcal{A}, W) =| F\ or\ G$ iff $(\mathcal{A}, W) =| F$ and $(\mathcal{A}, W) =| G$.

It is clear that if a formula G is of the form $KF, \neg KF, MF$ or $\neg MF$, then its truth value in $(\mathcal{A}, W)$ will not depend on W and we call G a *subjective formula.* On the other hand, if G does not contain K or M, then its truth value in $(\mathcal{A}, W)$ will only depend on W and we call G an *objective formula.* In the case that G is subjective, we simply write $\mathcal{A} \models G$ instead of $(\mathcal{A}, W) \models G$, and $W \models G$ instead of $(\mathcal{A}, W) \models G$ in the case that G is objective.

An *epistemic logic program* is a finite set of rules of the form:

$$F \leftarrow G_1, \cdots, G_m, notG_{m+1}, \cdots, notG_n. \qquad (1)$$

In (1) F is of the form $F_1\ or\ \cdots\ or\ F_k$ and $F_1, \cdots, F_k$ are objective literals, $G_1, \cdots, G_m$ are objective or subjective literals, and $G_{m+1}, \cdots, G_n$ are objective literals. For an epistemic logic program $\mathcal{P}$, its semantics is given by its *world view* which is defined in the following steps:

Step 1. Let V be an epistemic logic program not containing modal operators K and M and negation as failure *not.* A set W of ground literals is called a *belief set* of V iff W is a minimal set of satisfying conditions: (i) for each rule $F \leftarrow G_1, \cdots, G_m$ from $\mathcal{P}$ such that $W \models G_1 \wedge \cdots \wedge G_m$ we have $W \models F$; and (ii) if W contains a pair of complementary literals then $W = Lit$, i.e. W is an inconsistent belief set.

Step 2. Let $\mathcal{P}$ be an epistemic logic program not containing modal operators K and M and W be a set of ground literals in the language of $\mathcal{P}$. By $\mathcal{P}_W$ we denote the result of (i) removing from $\mathcal{P}$ all the rules containing formulas of the form $notG$ such that $W \models G$ and (ii) removing from the rules in $\mathcal{P}$ all other occurrences of formulas of the form $notG$.

Step 3. Finally, let $\mathcal{P}$ be an arbitrary epistemic logic program and $\mathcal{A}$ a collection of sets of ground literals in its language. By $\mathcal{P}_{\mathcal{A}}$ we denote the epistemic logic program obtained from $\mathcal{P}$ by (i) removing from $\mathcal{P}$ all rules containing formulas of the form G such that G is subjective and $\mathcal{A} \not\models G$, and (ii) removing from rules in $\mathcal{P}$ all other occurrences of subjective formulas.

Now we define that a collection A of sets of ground literals is a *world view* of V if A is the collection of all belief sets of P_A* Consider program V about the eligibility of scholarship discussed earlier, if we replace rule r_4 with the following rule:

r_4': $interview(x) \leftarrow \neg Keligible(x),$
$\neg K \neg eligible(x),$

then the epistemic logic program that consists of rules r_1, r_2, r_3, r_4', and r_5 will have a unique world view $\{\{highGPA(mike), eligible(mike), interview(mike)\},$ $\{fairGPA(mike), interview(mike)\}\}$, which will result in a *yes* answer to the query $interview(mike)$.

3 Formalizing Epistemic Logic Program Updates

From this section, we start to develop a formulation for epistemic logic program updates. Consider the update of an epistemic logic program $V\backslash$ by another epistemic logic program P_2- Our approach consists of two stages: firstly, we update each world view of "P_1 by P_2 - this will remove contradictory information between P_1 and P_2 and ensure a minimal change for the underlying update semantics; and secondly, from the first stage result, we will derive a resulting program which retains the maximal consistent information represented by $V\backslash$.

3.1 Preliminaries

To begin with, we first introduce some useful notions. Let $\mathcal{A}$ be a collection of belief sets (i.e. sets of ground literals) and $\mathcal{P}$ an epistemic logic program. We call a pair $(\mathcal{A}, W)$ where $W \in \mathcal{A}$ an *epistemic model induced from* $\mathcal{A}$. If $\mathcal{A}$ is a world view of $\mathcal{P}$, then $\mathcal{M} = (\mathcal{A}, W)$ is also called an *epistemic model* of $\mathcal{P}$. We use $ind(\mathcal{A})$ to denote the set of all epistemic models induced from $\mathcal{A}$. Note $\|ind(\mathcal{A})\| = \|\mathcal{A}\|$.

Consider a rule r of the form (1). We use $H(r)$ and $B(r)$ to denote the head and body parts of rule r respectively. For instance, for rule r: $a\ or \neg b \leftarrow c, Kd, not\ \neg e$, we have $H(r) = \{a, \neg b\}$, and $B(r) = \{c, Kd, not \neg e\}$. We say $H(r)$ is *satisfied* in an epistemic model $\mathcal{M}$, denoted as $\mathcal{M} \models H(r)$, iff for some $F \in H(r)$ $\mathcal{M} \models F$. We say $B(r)$ is *satisfied* in $\mathcal{M}$, denoted as $\mathcal{M} \models B(r)$, iff (1) for each objective or subjective literal $G \in B(r)$ $\mathcal{M} \models G$, and (2) for each $not\ G \in B(r)$ $\mathcal{M} \not\models G$. r *is satisfied* in $\mathcal{M}$, denoted

as $\mathcal{M} \models r$, if $\mathcal{M} \models B(r)$ implies $\mathcal{M} \models H(r)$. An epistemic logic program $\mathcal{P}$ is *satisfied* in $\mathcal{M}$ if each rule of $\mathcal{P}$ is satisfied in $\mathcal{M}$. $\mathcal{P}$ is *satisfied* in a collection of belief sets $\mathcal{A}$ if $\mathcal{P}$ is satisfied in all epistemic models induced from $\mathcal{A}$. A ground literal L *is derivable* from $\mathcal{P}$, denoted as $\mathcal{P} \models L$, if L is true in all world views of $\mathcal{P}$, i.e. L is true in all epistemic models induced from each world view of $\mathcal{P}$.

A collection of belief sets is *consistent* if each of its belief sets is consistent. An epistemic logic program is *consistent* if it has a world view and all of its world views are consistent. To simplify our presentation, in the rest of this paper, we will simply call an epistemic logic program *program*. By V(P) we denote the set of all collections of belief sets in which V is satisfied. We also denote the set of all world views of V as $\mathbf{A}(\mathcal{P})$. Clearly $\mathbf{A}(\mathcal{P}) \subseteq \mathbf{V}(\mathcal{P})$.

3.2 Minimal Change on World View Updates

Let W and W_1 be two belief sets. We define $Diff(W, W_1)$ to be the set $|(W \setminus W_1) \cup (W_1 \setminus W)|$[1]. Our method of updating world views shares a similar spirit of traditional model based update [Winslett, 1988] by defining a closeness relation between collections of belief sets, and after the update, the resulting collection of belief sets should be as close as possible to the original one.

Definition 1 **(Closeness)** *Let $\mathcal{A}$, $\mathcal{A}_1$ and $\mathcal{A}_2$ be three collections of belief sets. We say $\mathcal{A}_1$ is* as close to $\mathcal{A}$ as *$\mathcal{A}_2$, denoted as $\mathcal{A}_1 \leq_{\mathcal{A}} \mathcal{A}_2$, iff for any $W \in \mathcal{A}$ and $W_2 \in \mathcal{A}_2$, there exists a $W_1 \in \mathcal{A}_1$ such that $Diff(W, W_1) \subseteq Diff(W, W_2)$. We denote $\mathcal{A}_1 <_{\mathcal{A}} \mathcal{A}_2$ if $\mathcal{A}_1 \leq_{\mathcal{A}} \mathcal{A}_2$ and $\mathcal{A}_2 \not\leq_{\mathcal{A}} \mathcal{A}_1$.*

Proposition 1 *$\leq_{\mathcal{A}}$ is a partial ordering.*

Definition 2 **(Updating world views)** *Let $\mathcal{A}$ be a world view of some program and $\mathcal{P}$ a program. A collection of belief sets $\mathcal{A}'$ is a* possible result of updating $\mathcal{A}$ by $\mathcal{P}$, *if*

1. *$\mathcal{A}' \models \mathcal{P}$ (i.e. $\mathcal{A}' \in \mathbf{V}(\mathcal{P})$); and*
2. *there does not exist another collection of belief sets $\mathcal{A}''$ satisfying condition 1 and $\mathcal{A}'' <_{\mathcal{A}} \mathcal{A}'$.*

We use $Res(\mathcal{A}, \mathcal{P})$ to denote the set of all possible collections of belief sets after updating $\mathcal{A}$ by $\mathcal{P}$.

Example 1 Consider two epistemic logic programs $\mathcal{P}_1$ and $\mathcal{P}_2$ where

$\mathcal{P}_1$:
$a \leftarrow$,
$\neg b \leftarrow not\ c$,
$c \leftarrow not\ \neg b$, and
$\mathcal{P}_2$:
$b\ or\ c \leftarrow Ka$,

Clearly, $\mathcal{P}_1$ has one world view $\mathcal{A}_1 = \{\{a, \neg b\}, \{a, c\}\}$. Updating $\mathcal{A}_1$ by $\mathcal{P}_2$, according to Definition 2, we obtain a unique result $\mathcal{A}' = \{\{a, b\}, \{a, c\}\}$. Note that neither $\mathcal{A}'' = \{\{a, c\}\}$ nor $\mathcal{A}^* = \{\{a, \neg b, c\}, \{a, c\}\}$ is a result of updating $\mathcal{A}$ by $\mathcal{P}_2$. Although $\mathcal{A}'' \models \mathcal{P}_2$ and $\mathcal{A}^* \models \mathcal{P}_2$, we have $\mathcal{A}' <_{\mathcal{A}_1} \mathcal{A}''$ and $\mathcal{A}' <_{\mathcal{A}_1} \mathcal{A}^*$ according to Definition 1. ■

[1] For a set of ground literals W, $|W| = \{|l| \mid l \in W\}$ and here $|l|$ is l's corresponding propositional atom, i.e. $|l| = a$ if l is a or $\neg a$.

Proposition 2 *Given a collection of belief sets $\mathcal{A}$ and a program $\mathcal{P}$. Then $Res(\mathcal{A}, \mathcal{P}) = Min(\mathbf{V}(\mathcal{P}), \leq_{\mathcal{A}})$, where $Min(\mathbf{V}(\mathcal{P}), \leq_{\mathcal{A}})$ is the subset of $\mathbf{V}(\mathcal{P})$ containing all minimal elements with respect to ordering $\leq_{\mathcal{A}}$.*

3.3 Maximal Coherence and Resulting Programs

As discussed earlier, during the second stage of an update procedure, we need to derive a resulting program which should contain the maximal consistent information represented by the initial program in syntactic forms. This is achieved by introducing the concept of *coherence*. Let $\mathcal{M} = (\mathcal{A}, W)$ be an epistemic model and r a rule of the form (1) and $\mathcal{M} \models r$. We define $\mathcal{S}(\mathcal{M}, H(r)) = \{l \mid l \in H(r) \cap W \text{ and } \mathcal{M} \models B(r)\}$. Intuitively, if $\mathcal{M} \models B(r)$, then $\mathcal{S}(\mathcal{M}, H(r))$ presents all literals occurring in $H(r)$ that are true in W. For example, let $\mathcal{M} = (\{\{a, b\}, \{a, c\}, \{a, b, c\}\}, \{a, b, c\})$ and $r : b\ or\ c \leftarrow Ka$, then $\mathcal{S}(\mathcal{M}, H(r)) = \{b, c\}$. In the case $\mathcal{M} \not\models B(r)$, we have $\mathcal{S}(\mathcal{M}, H(r)) = \emptyset$.

Definition 3 **(Subsumption)** *Given two collections of belief sets $\mathcal{A}_1$ and $\mathcal{A}_2$ and a program $\mathcal{P}$ where $\mathcal{A}_1 \in \mathbf{V}(\mathcal{P})$. $\mathcal{A}_2$ is* subsumed by $\mathcal{A}_1$ *with respect to $\mathcal{P}$, denoted as $\mathcal{A}_2 \subseteq_{\mathcal{P}} \mathcal{A}_1$, iff $\mathcal{A}_2 \in \mathbf{V}(\mathcal{P})$ and for each $\mathcal{M}_2 \in ind(\mathcal{A}_2)$ there exists a $\mathcal{M}_1 \in ind(\mathcal{A}_1)$ such that for each $r \in \mathcal{P}$, $\mathcal{S}(\mathcal{M}_2, H(r)) \subseteq \mathcal{S}(\mathcal{M}_1, H(r_2))$.*

Given $\mathcal{A}_1 \models \mathcal{P}$, if $\mathcal{A}_2$ is subsumed by $\mathcal{A}_1$ with respect to $\mathcal{P}$, then each rule r in $\mathcal{P}$ is also satisfied in $\mathcal{A}_2$ without increasing the satisfied literals of $H(r)$ in epistemic models induced from $\mathcal{A}_2$. For instance, let $\mathcal{P} = \{r : a\ or\ b \leftarrow Kc\}$, $\mathcal{A}_1 = \{\{a, b, c, \}, \{b, c, d\}\}$ and $\mathcal{A}_2 = \{\{a, c\}, \{b, c, e\}\}$, then $\mathcal{A}_2 \subseteq_{\mathcal{P}} \mathcal{A}_1$. Note that belief set $\{a, c\}$ in $\mathcal{A}_2$ only contains one literal of $H(r)$ where belief set $\{a, b, c\}$ in $\mathcal{A}_1$ contains both literals of $H(r)$.

Definition 4 **(Coherence)** *Let $\mathcal{P}$ and $\mathcal{P}'$ be two programs and $\mathcal{A} \in \mathbf{V}(\mathcal{P})$. $\mathcal{P}'$ is* coherent with $\mathcal{P}$ *with respect to $\mathcal{A}$ if for each $\mathcal{A}' \in Res(\mathcal{A}, \mathcal{P}')$, $\mathcal{A}'$ is consistent and $\mathcal{A}' \subseteq_{\mathcal{P}} \mathcal{A}$.*

The intuitive meaning of coherence is explained as follows. Given a collection of belief sets $\mathcal{A}$ and programs $\mathcal{P}$ and $\mathcal{P}'$ where $\mathcal{A} \models \mathcal{P}$. Updating $\mathcal{A}$ by $\mathcal{P}'$ will change $\mathcal{A}$ to other collections of belief sets that satisfy $\mathcal{P}'$. Coherence ensures that such changes do not violate the satisfaction of $\mathcal{P}$, i.e. for each $\mathcal{A}' \in Res(\mathcal{A}, \mathcal{P}')$ $\mathcal{A}' \models \mathcal{P}$. Furthermore, it also ensures that the result $\mathcal{A}'$ is not larger than $\mathcal{A}$ in terms of $\mathcal{P}$'s satisfaction. Note that coherence only associates with consistent programs.

Proposition 3 *Let $\mathcal{P}$ and $\mathcal{P}'$ be two programs and $\mathcal{A} \in \mathbf{V}(\mathcal{P})$. If $\mathcal{P}'$ is coherent with $\mathcal{P}$ with respect to $\mathcal{A}$, then $\mathcal{P} \cup \mathcal{P}'$ is a consistent program.*

Proof: Since $\mathcal{P}'$ is coherent with $\mathcal{P}$ with respect to $\mathcal{A}$, it means that for each $\mathcal{A}' \in Res(\mathcal{A}, \mathcal{P}')$ $\mathcal{A}'$ is consistent, $\mathcal{A}' \models \mathcal{P}'$ and $\mathcal{A}' \models \mathcal{P}$. It also follows $\mathcal{A}' \in \mathbf{V}(\mathcal{P} \cup \mathcal{P}')$. From Lemma 1 in section 5, we know that $\mathcal{P} \cup \mathcal{P}'$ is a consistent program. ■

Now the resulting program can be specified by the following definition.

Definition 5 ***(Resulting programs)*** *Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be two programs and $\mathcal{A}_1 \in \mathbf{A}(\mathcal{P}_1)$. A program $\mathcal{P}'$ is a* possible resulting program *after updating $\mathcal{P}_1$ by $\mathcal{P}_2$, iff $\mathcal{P}' = \mathcal{P}_1^* \cup \mathcal{P}_2$, where $\mathcal{P}_1^*$ is a maximal subset of $\mathcal{P}_1$ such that for each $\mathcal{A}' \in Res(\mathcal{A}_1, \mathcal{P}_2)$ $\mathcal{P}_1^*$ is coherent with $\mathcal{P}_2$ with respect to $\mathcal{A}'$.*

Example 2 Example 1 continued. Note that $\mathcal{A}_1 = \{\{a, \neg b\}, \{a, c\}\}$ is the unique world view of $\mathcal{P}_1$. After updating $\mathcal{A}_1$ by $\mathcal{P}_2$, we have $Res(\mathcal{A}_1, \mathcal{P}_2) = \{\{\{a, b\}, \{a, c\}\}\}$ as shown in Example 1. Then it can be verified that $\{a \leftarrow\}$ is the only maximal subset of $\mathcal{P}_1$ that is coherent with $\mathcal{P}_2$ with respect to $\{\{a, b\}, \{a, c\}\}$. Therefore, from Definition 5, we have

$\mathcal{P}'$:
$a \leftarrow$,
b *or* $c \leftarrow Ka$,

from which, we conclude that $\mathcal{P}'$ has a unique world view $\{\{a, b\}, \{a, c\}\}$. ■

4 Handling Update Sequences

In this section, we extend our previous formulation to handle update sequences where more than two programs are involved. Let $\mathcal{P}_1, \cdots, \mathcal{P}_k$ be consistent programs. $\mathbf{P} = (\mathcal{P}_1, \cdots, \mathcal{P}_k)$ is called an *update sequence*. Informally performing this update sequence means that program $\mathcal{P}_1$ is sequentially updated by $\mathcal{P}_2$, $\cdots$, by $\mathcal{P}_k$. From our intuition, during this process we would like to assign $\mathcal{P}_j$ a higher priority of persistence than $\mathcal{P}_i$ where $j > i$ since $\mathcal{P}_j$ represents the agent's newly received knowledge comparing to $\mathcal{P}_i$. Therefore, after the performance of this update sequence, it is desirable to achieve a result which maximally contains information represented by $\mathcal{P}_k$, $\mathcal{P}_{k-1}$, $\cdots$, $\mathcal{P}_1$ with *progressively decreasing priorities*. Similarly to our previous update formulation, we will formalize this principle from both semantic and syntactic considerations. We first illustrate our idea by the following example.

Example 3 Consider an update sequence $\mathbf{P} = (\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3)$, where

$\mathcal{P}_1$:
r_1: $b \leftarrow Ka$,
r_2: $\neg b \leftarrow Ka$,
$\mathcal{P}_2$:
r_3: $a \leftarrow$,
$\mathcal{P}_3$:
r_4: $\neg a$ *or* $\neg b \leftarrow$.

To perform update sequence $\mathbf{P}$, we first update $\mathcal{P}_1$ by $\mathcal{P}_2$, which, according to the approach developed earlier, will generate two resulting programs: $\mathcal{P}' = \{r_1, r_3\}$ and $\mathcal{P}'' = \{r_2, r_3\}$. Now we consider to update $\mathcal{P}'$ and $\mathcal{P}''$ by $\mathcal{P}_3$ respectively. However, this time, we cannot simply apply our previous update approach because both $\mathcal{P}'$ and $\mathcal{P}''$ contain rules from $\mathcal{P}_1$ and $\mathcal{P}_2$ where rules from $\mathcal{P}_2$ should have a higher priority to be maintained during the change. For instance, consider the update of $\mathcal{P}'$ by $\mathcal{P}_3$. $\mathcal{P}'$ has one world view $\mathcal{A}' = \{\{a, b\}\}$. Then updating $\mathcal{A}'$ by $\mathcal{P}_3$ will generate a result $\mathcal{A}'' = \{\{\neg a, b\}, \{a, \neg b\}\}$. Clearly, both $\{r_1\}$ and $\{r_3\}$ are the maximal subsets of $\mathcal{P}'$ that are coherent with $\mathcal{P}_3$ with respect to $\mathcal{A}''$. However, as we discussed above, we should only have a final resulting program $\{r_3\} \cup \mathcal{P}_3$ because r_3 has a higher priority than r_1 to be persistent during the update. Updating $\mathcal{P}''$ by $\mathcal{P}_3$, on the other hand, will result in a resulting program $\mathcal{P}'' \cup \mathcal{P}_3$ as there is no any contradiction between $\mathcal{P}''$ and $\mathcal{P}_3$. ■

Let $\mathbf{P} = (\mathcal{P}_1, \cdots, \mathcal{P}_k)$ be an update sequence and $\mathcal{P}'$ a subset of $\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_k$. We denote $\mathcal{P}'[\mathbf{P}, i] = \{r \mid r \in \mathcal{P}_i \cap \mathcal{P}'\}$ $(1 \leq i \leq k)$. Note that $\mathcal{P}'[\mathbf{P}, i] \subseteq \mathcal{P}_i$. Now the following two definitions formalize our idea of performing update sequences as discussed above.

Definition 6 ***(Preferred subsets)*** *Let $\mathbf{P} = (\mathcal{P}_1, \cdots, \mathcal{P}_k)$ be an update sequence and $\mathcal{P}'$ and $\mathcal{P}''$ two subsets of $\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_k$. We say that $\mathcal{P}'$ is* more preferred than *$\mathcal{P}''$ with respect to $\mathbf{P}$, denoted as $\mathcal{P}' \ll_{\mathbf{P}} \mathcal{P}''$, iff (i) $\mathcal{P}'' \subset \mathcal{P}'$; or (ii) there exist some i and j where $1 \leq i < j \leq k$ such that for all l $(j \leq l \leq k)$, $\mathcal{P}''[\mathbf{P}, l] \subseteq \mathcal{P}'[\mathbf{P}, l]$, for some l' $(j \leq l' \leq k)$ $\mathcal{P}''[\mathbf{P}, l'] \subset \mathcal{P}'[\mathbf{P}, l']$, and $\mathcal{P}'[\mathbf{P}, i] \subset \mathcal{P}''[\mathbf{P}, i]$.*

Definition 7 ***(Update selection functions)*** *Let $\mathbb{P}$ be the collection of all epistemic logic programs. For each $i \geq 1$, we define Π_i to be a i-ary* update selection function *if Π_i is a mapping:*

$$\Pi_i : \underbrace{\mathbb{P} \times \cdots \times \mathbb{P}}_{i} \longrightarrow 2^{\mathbb{P}}$$

where the following conditions hold:

1. *if $i = 1$, then for any update sequence $\mathbf{P} = (\mathcal{P}_1)$, $\Pi_1(\mathbf{P}) = \{\mathcal{P}_1\}$;*
2. *if $i = 2$, then for any update sequence $\mathbf{P} = (\mathcal{P}_1, \mathcal{P}_2)$, $\Pi_2(\mathbf{P}) = \{\mathcal{P}' \mid$ where $\mathcal{P}'$ is a resulting program as described in Definition 5$\}$;*
3. *if $i = k$ and $k > 2$, then for any update sequence $\mathbf{P} = (\mathcal{P}_1, \cdots, \mathcal{P}_k)$, $\mathcal{P} \in \Pi_k(\mathbf{P})$ iff $\mathcal{P} = \mathcal{P}^* \cup \mathcal{P}_k$, where $\mathcal{P}^*$ is obtained as follows:*
   - *(a) let $\mathbf{P}_1 = (\mathcal{P}_1, \cdots, \mathcal{P}_{k-1})$ and $\mathcal{P}' \in \Pi_{k-1}(\mathbf{P}_1)$;*
   - *(b) let $\mathcal{A}$ be a world view of $\mathcal{P}'$, and $\mathcal{P}^*$ is a maximal subset of $\mathcal{P}'$ such that for all $\mathcal{A}' \in Res(\mathcal{A}, \mathcal{P}_k)$ $\mathcal{P}^*$ is coherent with $\mathcal{P}_k$ with respect to $\mathcal{A}'$;*
   - *(c) there does not exist another maximal subset $\mathcal{P}^\dagger$ of $\mathcal{P}'$ that satisfies condition (b) and $\mathcal{P}^\dagger \ll_{\mathbf{P}_1} \mathcal{P}^*$[2].*

Example 4 Consider an irrigation system which has the following general rules to decide whether the plants should be watered:

> *If there is no evidence showing that it will not be raining next day, then we do not need to water the plants;* and
> *If there is no evidence showing that the soil is dry, then we do not need to water the plants.*

It is also assumed that the soil is dry or it will not be raining next day. This scenario can be represented by the following program P_1:

[2]Note that here we refer to the same world view $\mathcal{A}$ of $\mathcal{P}'$ as in (b) when we say $\mathcal{P}^\dagger$ is coherent with $\mathcal{P}_k$ with respect to all $\mathcal{A}'$ in $Res(\mathcal{A}, \mathcal{P}_k)$.

$\mathcal{P}_1$:
r_1: $\neg watering \leftarrow not\ \neg to\_be\_raining$,
r_2: $\neg watering \leftarrow not\ dry$,
r_3: $dry\ or\ \neg to\_be\_raining \leftarrow$.

$\mathcal{P}_1$ has one world view:

$$\{\{dry, \neg watering\}, \{\neg to\_be\_raining, \neg watering\}\},$$

from which it is concluded that we do not need to water the plants. However, from a conservative viewpoint for plants' growth, this result is rather optimistic because r_3 does not represent an exclusive disjunctive information. Therefore, we consider to update P_1 by P_2

$\mathcal{P}_2$:
r_4: $watering \leftarrow \neg K \neg dry, M \neg to\_be\_raining$,
r_5: $\neg watering \leftarrow \neg dry$,
r_6: $\neg watering \leftarrow to\_be\_raining$,

which says that if it is not known that the soil is dry and it may be believed that it will not be raining next day, then we water the plants; and we do not need to water the plants if the soil is not dry or it will be raining next day. After a period of time, suppose new information is further received that is represented by P_3 as follows:

$\mathcal{P}_3$:
r_7: $\neg dry\ or\ to\_be\_raining \leftarrow$.

Now we consider the update sequence $\mathbf{P} = (\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3)$. Let $\mathbf{P}_1 = (\mathcal{P}_1, \mathcal{P}_2)$. Then according to Definition 7, We have $\Pi_2(\mathbf{P}_1) = \{\mathcal{P}'\}$, where $\mathcal{P}' = \{r_3, r_4, r_5, r_6\}$ (note that rules r_1 and r_2 are removed from the initial program $\mathcal{P}_1$). $\mathcal{P}'$ has one world view $\{\{dry, watering\}, \{\neg to\_be\_raining, watering\}\}$, from which it is concluded that we need to water the plants. Finally, we have a unique resulting program $\Pi_3(\mathbf{P}) = \{\mathcal{P}''\}$ where $\mathcal{P}'' = \{r_4, r_5, r_6, r_7\}$, from which it is concluded that we no longer need to water the plants, i.e. $\mathcal{P}'' \models \neg watering$. ■

5 Semantic Characterizations

In this section, we study important semantic properties of our approach for epistemic logic program updates. Let $\mathbf{P} = (\mathcal{P}_1, \cdots, \mathcal{P}_k)$ be an update sequence and $\mathcal{P}$ a program, by $(\mathbf{P}, \mathcal{P})$ we denote the update sequence $(\mathcal{P}_1, \cdots, \mathcal{P}_k, \mathcal{P})$. By $Body(\mathcal{P})$ and $Head(\mathcal{P})$ we denote the sets of all objective literals occurring in the bodies and heads of rules in $\mathcal{P}$ respectively[3]. To simplify our presentation, we also use notion $\pi_k(\mathbf{P})$ to denote an arbitrary resulting program in $\Pi_k(\mathbf{P})$ whenever there is no confusion in the context. For instance, by $\pi_k(\mathbf{P}) \models L$, we mean that for any $\mathcal{P} \in \Pi_k(\mathbf{P})$, $\mathcal{P} \models L$; and by $\pi_k(\mathbf{P}) = \pi_{k+1}(\mathbf{P}, \mathcal{P})$, we mean that any resulting program in $\Pi_k(\mathbf{P})$ is also a resulting program in $\Pi_{k+1}(\mathbf{P}, \mathcal{P})$, and *vice versa*, etc..

Lemma 1 *A program $\mathcal{P}$ is consistent iff $\mathbf{V}(\mathcal{P}) \neq \emptyset$ and there is a consistent collection of belief sets in $\mathbf{V}(\mathcal{P})$.*

Theorem 1 *Let* P *be an update sequence with a length of k and V a program. Then the following properties hold:*

1. *$\pi_k(\mathbf{P})$ is a consistent program* (**consistency property**);
2. *$\mathbf{V}(\pi_{k+1}(\mathbf{P}, \mathcal{P})) \subseteq \mathbf{V}(\mathcal{P})$;*
3. *if $\mathbf{V}(\pi_k(\mathbf{P})) \subseteq \mathbf{V}(\mathcal{P})$, then $\pi_{k+1}(\mathbf{P}, \mathcal{P}) = \pi_k(\mathbf{P}) \cup \mathcal{P}$.*

The following theorem provides a sufficient condition under which the computation of a $(k+1)$-length update sequence can be reduced to the computation on a k-length update sequence.

Theorem 2 (**Reduction property**) *Let $\mathbf{P}$ be an update sequence with a length of k, and $\mathcal{P}$ and $\mathcal{P}'$ two programs. If $\mathbf{V}(\pi_k(\mathbf{P})) \subseteq \mathbf{V}(\mathcal{P}) \subseteq \mathbf{V}(\mathcal{P}')$, then $\pi_{k+2}(\mathbf{P}, \mathcal{P}, \mathcal{P}') = \pi_{k+1}(\mathbf{P}, \mathcal{P} \cup \mathcal{P}')$.*

Now we investigate two specific properties called *persistence* and *preservation* for epistemic logic program updates. Informally, the persistence property ensures that if a literal is derivable from a program, then after updating this program, this literal is still derivable from the resulting program. The preservation property, on the other hand, says that if a literal is derivable from a program, then updating other program by this program will still preserve this literal's derivability from the resulting program. While the persistence property is usually not valid for classical belief revision and update due to their nonmonotonicity, the preservation property, nevertheless, indeed holds for classical belief revision and update. It is not difficult to observe that generally none of these two properties holds for extended logic program updates or epistemic logic program updates. However, it is always worthwhile to explore their restricted forms because under certain conditions, these properties may significantly simplify the computation of a query to the update result. We first present the following lemma.

Lemma 2 *Let $\mathcal{P}$ be a program and L a ground literal. Suppose $\mathcal{P}'$ is a subset of $\mathcal{P}$ such that $\mathcal{P}' \models L$ and $Body(\mathcal{P}') \cap Head(\mathcal{P} \setminus \mathcal{P}') = \emptyset$. Then $\mathcal{P} \models L$.*

Proof: (*Proof Sketch*) The proof for this lemma is rather technical and involves the proof of a result so called *splitting theorem* for epistemic logic programs which inherits a similar feature of the splitting theorem for extended logic programs [Lifschitz and Turner, 1994]. Basically, the splitting theorem for epistemic logic programs states that if $Body(\mathcal{P}') \cap Head(\mathcal{P} \setminus \mathcal{P}') = \emptyset$ (here $\mathcal{P}$ is a consistent program), then for any world view $\mathcal{A}$ of $\mathcal{P}$ and every belief set W in $\mathcal{A}$, there exists a belief set W' that is in some world view of $\mathcal{P}'$ such that $W' \subseteq W$. Due to a space limit, here we have to omit the formal proof of the splitting theorem of epistemic logic programs [Zhang, 2003][4]. Based on the splitting theorem, this lemma is proved as follows. Consider program $\mathcal{P}'$. Since $\mathcal{P}' \models L$, for each world view $\mathcal{A}'$ of $\mathcal{P}'$, $\mathcal{A}' \models L$. That is, for each $W' \in \mathcal{A}'$, $L \in W'$. Suppose $\mathcal{P} \not\models L$. Then there exists a world view $\mathcal{A}$ of $\mathcal{P}$ such that $\mathcal{A} \not\models L$. Then there must be some belief set $W \in \mathcal{A}$ such that $L \notin W$. However, since $Body(\mathcal{P}') \cap Head(\mathcal{P} \setminus \mathcal{P}') = \emptyset$,

[3]Note that $Body(\mathcal{P})$ is different from $B(r)$ while $Head(\mathcal{P}) = \cup_{r \in \mathcal{P}} H(r)$. See section 3.1 for definitions on $B(r)$ and $H(r)$.

[4]One referee pointed that as a general extension of Lifschitz and Turner's result, Watson also proposed a splitting set theorem for epistemic logic programs [Watson, 2000]. It appears that our splitting theorem has a different feature from Watson's though the later may be also used for this proof.

from the splitting theorem, there exists a world view $\mathcal{A}^*$ of $\mathcal{P}'$ such that for some $W^* \in \mathcal{A}^*$ we have $W^* \subseteq W$. Since $\mathcal{A}^* \models L$, it follows $L \in W^*$ and then $L \in W$. Obviously this contradicts the fact $L \notin W$. So we have $\mathcal{P} \models L$. ■

From Lemma 2, we can prove the following two properties.

Theorem 3 *(**Persistence property**) Let $\mathbf{P}$ be an update sequence with a length of k, $\mathcal{P}$ a program, and L a ground literal. Suppose $\pi_k(\mathbf{P}) \models L$. Then $\pi_{k+1}(\mathbf{P}, \mathcal{P}) \models L$ if $\mathbf{V}(\pi_k(\mathbf{P})) \subseteq \mathbf{V}(\mathcal{P})$ and $Body(\pi_k(\mathbf{P})) \cap Head(\mathcal{P}) = \emptyset$.*

Proof: Since $\mathbf{V}(\pi_k(\mathbf{P})) \subseteq \mathbf{V}(\mathcal{P})$, from Theorem 1, we have $\pi_{k+1}(\mathbf{P}, \mathcal{P}) = \pi_k(\mathbf{P}) \cup \mathcal{P}$. On the other hand, from condition $Body(\pi_k(\mathbf{P})) \cap Head(\mathcal{P}) = \emptyset$ and Lemma 2, it follows $\pi_{k+1}(\mathbf{P}, \mathcal{P}) \models L$. ■

Theorem 4 *(**Preservation property**) Let $\mathbf{P} = (\mathcal{P}_1, \cdots, \mathcal{P}_k)$ be an update sequence with a length of k, $\mathcal{P}$ a program, and L a ground literal. Suppose $\mathcal{P} \models L$. Then $\pi_{k+1}(\mathbf{P}, \mathcal{P}) \models L$ if $Head(\bigcup_{i=1}^{k} \mathcal{P}_i) \cap Body(\mathcal{P}) = \emptyset$.*

Proof: Clearly $\mathcal{P} \subseteq \pi_{k+1}(\mathbf{P}, \mathcal{P})$. On the other hand, we have $\pi_{k+1}(\mathbf{P}, \mathcal{P}) = \mathcal{P}^* \cup \mathcal{P}$, where $\mathcal{P}^* \subseteq \bigcup_{i=1}^{k} \mathcal{P}_i$ and satisfies conditions (b) and (c) in Definition 7. So we have $Head(\mathcal{P}^*) \cap Body(\mathcal{P}) = \emptyset$. Then directly from Lemma 2, we have $\pi_{k+1}(\mathbf{P}, \mathcal{P}) \models L$. ■

6 Concluding Remarks

In this paper, we proposed a formulation for epistemic logic program updates. Our update approach was developed based on the principle of minimal change and maximal coherence. By using our approach, not only a minimal change semantics is embedded into the underlying update procedure, but also a maximal syntactic coherence is achieved after the update. We also investigated important semantic properties of our update approach. This work can be viewed as a further development on knowledge update [Baral and Zhang, 2001]. Although all current approaches of logic program updates have their own features, it is not clear yet whether they are suitable to handle epistemic logic program updates. For instance, in [Zhang, 2003] we demonstrated that a straightforward extension of Alferes *et al's* approach [Alferes and *el* a/, 2000J or Eiter *et* ai.'s approach [Liter and *ct al,* 2002] to epistemic logic program updates may generate incorrect solutions, while the proposed generic framework in [Eiter and *et* a/, 2001] seems not applicable to our case either. On the other hand, a syntax-based approach, e.g. [Sakama and Inoue, 1999], cannot characterize the semantics of epistemic logic program updates, and some approaches, e.g. [Eiter and *ct al,* 20021, do not obey the consistency requirement that is believed to be one of the essential requirements for any revision and update systems [Katsuno and Mendelzon, 1991].

Several issues remain open for our future research. First, in our update approach, we did not consider the issue of preference over different resulting programs. In practice, it is possible that one resulting program is more preferred than the other in terms of the domain semantics. This problem involves conflict resolution which is a difficult issue in logic program updates [Zhang and Foo, 1998]. Second, as world view semantics can be viewed as a generalization of answer set semantics, our update approach is also applicable for extended logic program updates. Therefore, it is important to characterize similarities and differences between our approach and other logic program update approaches from a semantic viewpoint. Finally, since epistemic logic programs have been used as a main component in knowledge based action theories, e.g. [Lobo *et* a/., 2001], we expect that these theories may be significantly enhanced for representing interactions between actions and knowledge by applying our update approach.

References

[Alferes and *et* a/, 2000] J. Alferes and *et ah* Dynamic updates of non-monotonic knowledge bases. *Journal of Logic Programming,* 45(1 -2):43-70,2000.

[Baral and Zhang, 2001] C. Baral and Y. Zhang. On the semantics of knowledge update. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01),* pages 97-102. Morgan Kaufmann Publishers, Inc., 2001.

[Eiter and *ct ah* 2001] T Eiter and *et al. A* framework for declarative update specification in logic programs. In *Proceedings of IJCAI-Ol,* pages 649-654, 2001.

[Eiter and *ct ah* 2002] T. Eiter and *et al.* On properties of update sequences based on causal rejection. *Theory and Practice of Logic Programming,* 2:711-767, 2002.

[Gelfond, 1994] M. Gelfond. Logic programming and reasoning with incomplete information. *Annals of Mathematics and Artificial Intelligence,* 12:98-116, 1994.

[Katsuno and Mendelzon, 1991] H. Katsuno and A. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proceedings of of KR-91,* pages 387-394, 1991.

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

[Lobo *et al,* 2001] J. Lobo, G. Mendez, and S.R. Taylor. Knowledge and the action description language -I. *Theory and Practice of Logic Programming,* 1:129-184, 2001.

[Sakama and Inoue, 1999] C. Sakama and K. Inoue. Updating extended logic programs through abduction. In *Proceedings of LPNMR'99,* pages 147-161. LNAI, Vol 1730, Springer, 1999.

[Watson, 2000] R. Watson. A splitting set theorem for epistemic specifications. In *Proceedings of NMR 2000,* 2000.

[Winslett, 1988] M. Winslett. Reasoning about action using a possible models approach. In *Proceedings of AAAI-88,* pages 89-93, 1988.

[Zhang and Foo, 1998] Y. Zhang and N.Y. Foo. Updating logic programs. In *Proceedings of ECAI-98,* pages 403-407. John Wiley & Sons, Inc., 1998.

[Zhang, 2003] Y. Zhang. Minimal change and maximal coherence for epistemic logic program updates. In *Manuscript,* 2003.