Updating Extended Logic Programs through Abduction

Chiaki Sakama[1] and Katsumi Inoue[2]

[1] Department of Computer and Communication Sciences
Wakayama University
Sakaedani, Wakayama 640 8510, Japan
sakama@sys.wakayama-u.ac.jp
[2] Department of Electrical and Electronics Engineering
Kobe University
Rokkodai, Nada, Kobe 657 8501, Japan
inoue@eedept.kobe-u.ac.jp

Abstract. This paper introduces techniques for updating knowledge bases represented in extended logic programs. Three different types of updates, *view updates*, *theory updates*, and *inconsistency removal*, are considered. We formulate these updates through *abduction*, and provide methods for computing them with *update programs*. An update program is an extended logic program which specifies changes on abductive hypotheses, then updates are computed by the *U-minimal answer sets* of an update program. The proposed technique provides a uniform framework for these different types of updates, and each update is computed using existing procedures of logic programming.

1 Introduction

A knowledge base must be updated when new information arrives. There are three cases in updating a knowledge base. The first one is that a knowledge base contains two different kinds of knowledge — variable knowledge and invariable knowledge. In this case, updates are permitted only on variable knowledge. Updates on the invariable part are translated into updates on the variable part. An example of this type of updates is a *view update* in deductive databases. The second one is that there is no such distinction and the whole knowledge base is subject to change. In this case, an update is done by directly introducing new information to the knowledge base. When there are conflicts between the current knowledge and the new knowledge, a higher priority is put on the new one to produce a consistent theory as a whole. We call this type of updates a *theory update*. As the third one, suppose a knowledge base containing inconsistent information. In this situation, a knowledge base must be updated to restore consistency by removing the source of inconsistency. This type of updates is called *inconsistency removal*. There are many studies which consider the update issue in logic programming and deductive databases, while they are rather individual techniques to realize one of these updates.

In this paper we propose a unified framework for realizing view updates, theory updates, and inconsistency removal in *extended logic programs*. We first formulate these different types of updates through *abduction*, then introduce an *update program* which is an extended logic program specifying changes on abductive hypotheses. We show that each update is realized by computing the *U-minimal answer sets* of an update program.

This paper is organized as follows. Section 2 introduces a theoretical framework used in this paper. Section 3 presents a technique for computing view updates using update programs. Section 4 provides methods for computing theory updates and inconsistency removal. Section 5 analyzes computational complexities. Section 6 presents related work, and Section 7 concludes the paper.

2 Preliminaries

A knowledge base is represented in an *extended logic program* (ELP) [9], which is a set of rules of the form:

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

where L_i's are literals and *not* is negation as failure. The literal L_0 is the *head* and the conjunction $L_1, \ldots, L_m, \mathit{not}\, L_{m+1}, \ldots, \mathit{not}\, L_n$ is the *body*. The head is possibly empty. A rule is called a *fact* if the body is empty. An ELP is a *normal logic program* (NLP) if every L_i is an atom. Throughout the paper, a *program* means an ELP unless stated otherwise.

The semantics of an ELP is defined by the *answer set semantics* [9]. Let $\mathcal{L}_P$ be the set of all ground literals in the language of a program P. The *answer sets* of an ELP are defined by the following two steps. First, let P be a *not*-free ELP (i.e., for each rule $m = n$) and $S \subseteq \mathcal{L}_P$. Then, S is an *answer set* of P if S is a minimal set satisfying the conditions:

1. For each ground rule $L_0 \leftarrow L_1, \ldots, L_m$ from P, $\{L_1, \ldots, L_m\} \subseteq S$ implies $L_0 \in S$. In particular, $\{L_1, \ldots, L_m\} \not\subseteq S$ if L_0 is empty.
2. If S contains a pair of complementary literals L and $\neg L$, then $S = \mathcal{L}_P$.

Second, let P be any ELP and $S \subseteq \mathcal{L}_P$. Then, define a *not*-free ELP P^S as follows: a rule $L_0 \leftarrow L_1, \ldots, L_m$ is in P^S iff there is a ground rule $L_0 \leftarrow L_1, \ldots, L_m, \mathit{not}\, L_{m+1}, \ldots, \mathit{not}\, L_n$ from P such that $\{ L_{m+1}, \ldots, L_n \} \cap S = \emptyset$. For programs of the form P^S, their answer sets have already been defined. Then, S is an *answer set* of P if S is an answer set of P^S.

An answer set is *consistent* if it is not $\mathcal{L}_P$. A program P is *consistent* if it has a consistent answer set; otherwise P is *inconsistent*. If a literal L is included in every answer set of P, it is written as $P \models L$; otherwise $P \not\models L$. When P is inconsistent, we define $P \models \mathit{false}$.

An abductive framework used in this paper is the *extended abduction* introduced by Inoue and Sakama [11]. An *abductive program* is a pair $\langle P, \mathcal{A} \rangle$ where P is an ELP and $\mathcal{A} \subseteq \mathcal{L}_P$ is a set of literals from the language of P called

abducibles.[1] Any instance A of an element from $\mathcal{A}$ is also called an abducible and is written as $A \in \mathcal{A}$. Without loss of generality, we assume that any rule from P having an abducible in its head is always a fact.[2]

Let $\langle P, \mathcal{A} \rangle$ be an abductive program and G a ground literal. A pair (E, F) is an *explanation* (resp. *anti-explanation*) of an *observation* G wrt $\langle P, \mathcal{A} \rangle$ if

1. $(P \cup E) \setminus F \models G$ (resp. $(P \cup E) \setminus F \not\models G$),
2. $(P \cup E) \setminus F$ is consistent,
3. $E \subseteq \mathcal{A} \setminus P$ and $F \subseteq \mathcal{A} \cap P$.

Thus, to explain observations, extended abduction can not only introduce hypotheses to a program but also remove them from it. On the other hand, when a given fact is not observed anymore, anti-explanations are used to *un*explain the observation. Note that traditional abduction considers only introducing hypotheses to explain an observation, i.e., $P \cup E \models G$ where $P \cup E$ is consistent. We call such traditional abduction *normal abduction* to distinguish it from the extended one.

An (anti-)explanation (E, F) of an observation G is called *minimal* if for any (anti-)explanation (E', F') of G, $E' \subseteq E$ and $F' \subseteq F$ imply $E' = E$ and $F' = F$.

Example 2.1. Let $\langle P, \mathcal{A} \rangle$ be an abductive program such that

$$
\begin{aligned}
P:\ & \mathit{flies}(x) \leftarrow \mathit{bird}(x), \mathit{not}\, \mathit{ab}(x)\,, \\
& \mathit{ab}(x) \leftarrow \mathit{broken\text{-}wing}(x)\,, \\
& \mathit{bird}(\mathit{tweety}) \leftarrow,\ \ \mathit{bird}(\mathit{opus}) \leftarrow, \\
& \mathit{broken\text{-}wing}(\mathit{tweety}) \leftarrow\ . \\
\mathcal{A}:\ & \mathit{broken\text{-}wing}(x)\,.
\end{aligned}
$$

Then, the observation $G = \mathit{flies}(\mathit{tweety})$ has the minimal explanation $(E, F) = (\emptyset, \{\mathit{broken\text{-}wing}(\mathit{tweety})\})$, while the observation $G = \mathit{flies}(\mathit{opus})$ has the minimal anti-explanation $(\{\mathit{broken\text{-}wing}(\mathit{opus})\}, \emptyset)$.

3 View Updates

3.1 Update Programs

Suppose a knowledge base which contains variable knowledge and invariable knowledge. When there is a request for inserting/deleting knowledge to/from the knowledge base, the update request on invariable knowledge is translated into updates on variable knowledge. This type of updates is called view updates.

[1] In [11], P and $\mathcal{A}$ are given as autoepistemic theories. Here we consider a formulation in logic programming.

[2] If there is a rule $A \leftarrow \Gamma$ with an abducible A and a non-empty body Γ, then A is made a non-abducible by introducing a rule $A \leftarrow A'$ with a new abducible A'.

Definition 3.1. Let P be a program, V the set of variable rules in the language of P, and G a literal. Then, a program P' accomplishes a *view update* for the *insertion* (resp. *deletion*) of G to/from P if

1. P' is consistent,
2. $P' \models G$ (resp. $P' \not\models G$),
3. $P' \setminus V = P \setminus V$,
4. there is no consistent program P'' s.t. $P'' \models G$ (resp. $P'' \not\models G$), $P'' \setminus V = P \setminus V$, and $[(P \cap V) \sim (P'' \cap V)] \subset [(P \cap V) \sim (P' \cap V)]$, where $P \sim Q = (P \setminus Q) \cup (Q \setminus P)$.

In the above, the third condition presents that the invariable part of P is unchanging, while the fourth condition presents that P' minimally changes the variable part. In particular, when $G \in V \setminus P$ (resp. $G \in V \cap P$), the insertion (resp. deletion) is done by directly introducing (resp. deleting) G to/from P.

In this section, we consider a knowledge base in which facts are variable and other rules are invariable. Also, an update request is either an insertion or deletion of a ground literal to/from a knowledge base.[3]

With this setting, the view update problem is naturally expressed by an abductive program $\langle P, \mathcal{A} \rangle$, where the program P represents a knowledge base and the abducibles $\mathcal{A}$ represent updatable literals.

Proposition 3.1 *[11] Let $\langle P, \mathcal{A} \rangle$ be an abductive program and G a ground literal. Then, $(P \cup E) \setminus F$ accomplishes a view update for inserting (resp. deleting) G iff (E, F) is a minimal explanation (resp. minimal anti-explanation) of the observation G wrt $\langle P, \mathcal{A} \rangle$.*

The goal of this section is to provide a computational method for the above characterized view updates. To this end, we introduce the notion of update programs.

Definition 3.2. Given an abductive program $\langle P, \mathcal{A} \rangle$, the set UR of *update rules* are defined as follows.

1. For any literal $a \in \mathcal{A}$, the following rules are in UR:

$$a \leftarrow not\, \overline{a},$$
$$\overline{a} \leftarrow not\, a,$$

   where $\overline{a}$ is a newly introduced atom uniquely associated with a. For notational convenience, the above pair of rules are expressed as $abd(a)$, hereafter.
2. For any literal $a \in \mathcal{A} \setminus P$, the following rule is in UR:[4]

$$+a \leftarrow a\,.$$

[3] When $G = (H \leftarrow B)$, inserting (resp. deleting) G to/from P is rephrased as inserting (resp. deleting) a literal G' to/from $P \cup \{ H \leftarrow B, G' \}$ (resp. $(P \setminus G) \cup \{ H \leftarrow B, G' \} \cup \{ G' \leftarrow \}$).

[4] When $p(x) \in \mathcal{A}$, $p(a) \in P$ and $p(t) \notin P$ for $t \neq a$, the rule precisely becomes $+p(t) \leftarrow p(t)$ for any $t \neq a$. In such a case, the rule is shortly written as $+p(x) \leftarrow p(x),\ x \neq a$. Generally, the rule becomes $+p(x) \leftarrow p(x),\ x \neq t_1, \ldots, x \neq t_n$ for n such instances.

3. For any literal $a \in \mathcal{A} \cap P$, the following rule is in UR:

$$-a \leftarrow \mathit{not}\, a\,.$$

Here, $+a$ and $-a$ are uniquely associated with any $a \in \mathcal{A}$ and are called *update atoms*.

The atom $\overline{a}$ becomes true iff a is not true. Then, a pair of rules in $\mathit{abd}(a)$ specify the situation that an abducible a is true or not [15, 10]. The rule $+a \leftarrow a$ derives the atom $+a$ if an abducible a which is not in P is to be true. In contrast, the rule $-a \leftarrow \mathit{not}\, a$ derives the atom $-a$ if an abducible a which is in P is not to be true. The set of all update atoms associated with $\mathcal{A}$ is denoted by $\mathcal{UA}$. We define that $\mathcal{UA} = \mathcal{UA}^+ \cup \mathcal{UA}^-$, where $\mathcal{UA}^+$ (resp. $\mathcal{UA}^-$) is the set of update atoms of the form $+a$ (resp. $-a$).

Definition 3.3. Given an abductive program $\langle P, \mathcal{A} \rangle$, its *update program* UP is defined as an ELP such that

$$UP = (P \setminus \mathcal{A}) \cup UR\,.$$

Definition 3.4. An answer set S of UP is called *U-minimal* if there is no answer set T of UP such that $T \cap \mathcal{UA} \subset S \cap \mathcal{UA}$.

A U-minimal answer set represents a minimal change in P. When there is no update request, the following property holds.

Proposition 3.2 *Let $\langle P, \mathcal{A} \rangle$ be an abductive program in which P is consistent, and UP its update program. If S is a U-minimal answer set of UP, it holds that $S \cap \mathcal{UA} = \emptyset$ and there is an answer set T of P s.t. $T = S \cap \mathcal{L}_P$. Conversely, if T is an answer set of P, there is a U-minimal answer set S of UP s.t. $S \cap \mathcal{UA} = \emptyset$ and $S \cap \mathcal{L}_P = T$.*

Proof. When S is a U-minimal answer set of UP, $a \in S \cap \mathcal{A}$ iff $a \in \mathcal{A} \cap P$; and $a \in \mathcal{A} \setminus S$ iff $a \in \mathcal{A} \setminus P$. Hence, $S \cap \mathcal{UA} = \emptyset$ holds. Put $T = S \cap \mathcal{L}_P$. Then, $P^T = \{\, H \leftarrow\ B \mid (H \leftarrow\ B) \in UP^S \ \text{ and } H \in \mathcal{L}_P \,\}$. As S is an answer set of UP^S, T becomes an answer set of P^T. Hence, T is an answer set of P.

In converse, let T be an answer set of P. Put $S = T \cup \{\, \overline{a} \mid a \in \mathcal{A} \setminus P \,\}$. Then, $UP^S = P^T \cup \{\, \overline{a} \leftarrow \mid\ \overline{a} \in S \,\}$. As T is an answer set of P^T, S becomes an answer set of UP. Here $S \cap \mathcal{UA} = \emptyset$ holds, hence S is also U-minimal. □

Example 3.1. Let $\langle P, \mathcal{A} \rangle$ be an abductive program such that

$$\begin{aligned} P:\ & p \leftarrow b\,, \\ & q \leftarrow a,\ \mathit{not}\, b\,, \\ & a \leftarrow\ . \\ \mathcal{A}:\ & a,\ b\,. \end{aligned}$$

Then, UP becomes

$$\begin{aligned} UP:\ & p \leftarrow b\,, \quad q \leftarrow a, \mathit{not}\, b\,, \\ & \mathit{abd}(a), \quad \mathit{abd}(b)\,, \\ & -a \leftarrow \mathit{not}\, a\,, \quad +b \leftarrow b\,. \end{aligned}$$

Here, UP has four answer sets: $S_1 = \{\, a, b, +b, p \,\}$, $S_2 = \{\, \overline{a}, b, -a, +b, p \,\}$, $S_3 = \{\, a, \overline{b}, q \,\}$, and $S_4 = \{\, \overline{a}, \overline{b}, -a \,\}$. Of these, S_3 is the U-minimal answer set and $S_3 \cap \mathcal{L}_P$ coincides with the answer set of P.

Next we consider an update request for inserting/deleting ground literals. An insertion of a literal G is represented as the rule

$$\leftarrow \mathit{not}\, G\,,$$

which represents the constraint that "G should be true". On the other hand, a deletion of a literal G is represented as the rule

$$\leftarrow G\,,$$

which represents the constraint that "G must not be true".

To perform the insertion of p in the program of Example 3.1, consider the program $UP \cup \{\, \leftarrow \mathit{not}\, p \,\}$. It has two answer sets: S_1 and S_2, of which S_1 is the U-minimal answer set. Observe that the observation p has the unique minimal explanation $(\{b\}, \emptyset)$ wrt $\langle\, P, \mathcal{A} \,\rangle$. The situation is expressed by the update atom $+b$ in S_1. On the other hand, to perform the deletion of q, consider the program $UP \cup \{\, \leftarrow q \,\}$. It has three answer sets: S_1, S_2, and S_4, of which S_1 and S_4 are the U-minimal answer sets. Here, the observation q has two minimal anti-explanations $(\{b\}, \emptyset)$ and $(\emptyset, \{a\})$ wrt $\langle\, P, \mathcal{A} \,\rangle$. The situations are respectively expressed by the update atom $+b$ in S_1 and $-a$ in S_4. Note that when the insertion of p and the deletion of q are requested at the same time, S_1 becomes the unique U-minimal answer set of $UP \cup \{\, \leftarrow \mathit{not}\, p \,\} \cup \{\, \leftarrow q \,\}$.[5]

Thus, the U-minimal answer sets are used to compute minimal explanations which realize view updates in a knowledge base. Note that the constraint $\leftarrow\ \mathit{not}\, G$ extracts answer sets in which G is true, but this does not imply that G is true in every answer set of $(P \cup E) \setminus F$ in general. To know that (E, F) is an explanation of G, we need an additional test for checking the entailment of G from $(P \cup E) \setminus F$.

A pair of abducibles (E, F) is called a *pre-explanation* of an observation G if $(P \cup E) \setminus F$ has a consistent answer set in which G is true. A pre-explanation of G is *minimal* if for any pre-explanation (E', F') of G, $E' \subseteq E$ and $F' \subseteq F$ imply $E' = E$ and $F' = F$.[6]

[5] Generally, when there are insertion requests of $p_1, \ldots, p_m$ and deletion requests of $q_1, \ldots, q_n$, instead of considering the $(m+n)$-goals $\leftarrow \mathit{not}\, p_i$ and $\leftarrow q_j$, the same effect is achieved by introducing the rule $g \leftarrow p_1, \ldots, p_m, \mathit{not}\, q_1, \ldots, \mathit{not}\, q_n$ to UP and considering the single goal $\leftarrow \mathit{not}\, g$.

[6] The term pre-explanation means that it does not explain G *skeptically* (as in the definition of explanations) but explains G *credulously*. Note that every skeptical explanation is a pre-explanation (credulous explanation) (cf. Proposition 3.3).

Proposition 3.3 *Let $\langle P, \mathcal{A} \rangle$ be an abductive program and G an observation.*

1. *Any explanation (E, F) of G is a pre-explanation of G.*
2. *Any pre-explanation (E, F) of G is an explanation of G if $((P \cup E) \setminus F) \cup \{ \leftarrow G \}$ is inconsistent.*

Proof. 1. When (E, F) is an explanation of G in P, $(P \cup E) \setminus F \models G$ and $(P \cup E) \setminus F$ is consistent. Then, $(P \cup E) \setminus F$ has a consistent answer set in which G is true.

2. When (E, F) is a pre-explanation of G in P, $(P \cup E) \setminus F$ has a consistent answer set in which G is true. In addition, if $((P \cup E) \setminus F) \cup \{ \leftarrow G \}$ is inconsistent, every answer set of $(P \cup E) \setminus F$ contains G. Hence, $(P \cup E) \setminus F \models G$ and (E, F) is an explanation of G. □

In what follows, given sets $E \subseteq \mathcal{A}$ and $F \subseteq \mathcal{A}$, we define $E^+ = \{ +a \mid a \in E \}$ and $F^- = \{ -a \mid a \in F \}$. In converse, given sets $E^+ \subseteq \mathcal{UA}^+$ and $F^- \subseteq \mathcal{UA}^-$, define $E = \{ a \mid +a \in E^+ \}$ and $F = \{ a \mid -a \in F^- \}$.

Lemma 3.4 *Let $\langle P, \mathcal{A} \rangle$ be an abductive program, UP its update program, and G a ground literal. Then, there is a (minimal) pre-explanation (E, F) of G iff $UP \cup \{ \leftarrow not\, G \}$ has a consistent (U-minimal) answer set S s.t. $E^+ = S \cap \mathcal{UA}^+$ and $F^- = S \cap \mathcal{UA}^-$.*

Proof. Let S be a consistent answer set of $UP \cup \{ \leftarrow not\, G \}$ s.t. $E^+ = S \cap \mathcal{UA}^+$ and $F^- = S \cap \mathcal{UA}^-$. Then, for each $+a \in E^+$ and $-b \in F^-$, $a \leftarrow$ and $\overline{b} \leftarrow$ are produced by $abd(a)$ and $abd(b)$ in UP^S. This means that to make G true in S, $a \in \mathcal{A} \setminus P$ is introduced to P and $b \in \mathcal{A} \cap P$ is deleted from P. In this case, UP^S contains a rule $H \leftarrow B$ with $H \in \mathcal{L}_P$ iff $((P \cup E) \setminus F)^S$ has the same rule. Put $T = S \cap \mathcal{L}_P$. Then, T is a consistent answer set of $(P \cup E) \setminus F$ in which G is true. When S is U-minimal, suppose that the pair (E, F) is not minimal. Then there is a pair (E', F') s.t. $E' \subseteq E$, $F' \subseteq F$ and ($E' \neq E$ or $F' \neq F$), and $(P \cup E') \setminus F'$ has an answer set T' in which G is true. In this case, there is an answer set S' of $UP \cup \{ \leftarrow not\, G \}$ s.t. $T' = S' \cap \mathcal{L}_P$ and $S' \cap \mathcal{UA} \subset S \cap \mathcal{UA}$. This contradicts the assumption that S is U-minimal.

In converse, for a pair (E, F) of abducibles, let T be a consistent answer set of $(P \cup E) \setminus F$ in which G is true. Then, there is a program UP^T in which $a \leftarrow$ and $\overline{b} \leftarrow$ are produced by $abd(a)$ and $abd(b)$ for each $a \in E$ and $b \in F$. In this case, UP^T contains a rule $H \leftarrow B$ with $H \in \mathcal{L}_P$ iff $((P \cup E) \setminus F)^T$ has the same rule. Put $S = T \cup \{ +a \mid a \in E \} \cup \{ -b, \overline{b} \mid b \in F \}$. Then, S is a consistent answer set of UP in which G is true. When the pair (E, F) is minimal, it is shown in a similar way as above that S is also U-minimal. □

Theorem 3.5. *Let $\langle P, \mathcal{A} \rangle$ be an abductive program, UP its update program, and G a ground literal. Then, $(P \cup E) \setminus F$ accomplishes an insertion of G iff*

1. *S is a consistent answer set of $UP \cup \{ \leftarrow not\, G \}$ s.t. $E^+ = S \cap \mathcal{UA}^+$, $F^- = S \cap \mathcal{UA}^-$, and $((P \cup E) \setminus F) \cup \{ \leftarrow G \}$ is inconsistent; and*
2. *S is U-minimal among those satisfying the condition 1.*

Proof. Suppose that S satisfies the conditions. By the first condition, (E, F) is a pre-explanation of G (Lemma 3.4), and also an explanation of G (Proposition 3.3). By the second condition, (E, F) is a minimal explanation of G. Then, $(P \cup E) \setminus F$ accomplishes the insertion of G. In converse, suppose that $(P \cup E) \setminus F$ accomplishes the insertion of G. By Proposition 3.3 and Lemma 3.4, there is a consistent answer set S of $UP \cup \{ \leftarrow not\, G \}$ s.t. $E^+ = S \cap \mathcal{UA}^+$, $F^- = S \cap \mathcal{UA}^-$, and $((P \cup E) \setminus F) \cup \{ \leftarrow G \}$ is inconsistent. Thus, the first condition holds. To see that S is U-minimal among those satisfying the first condition, suppose that there is an answer set S' s.t. $S' \cap \mathcal{UA} \subset S \cap \mathcal{UA}$ and satisfying the first condition. Put $J^+ = S' \cap \mathcal{UA}^+$ and $K^- = S' \cap \mathcal{UA}^-$. Then, $J^+ \subset E^+$ or $K^- \subset F^-$, and $(P \cup J) \setminus K \models G$ holds by Proposition 3.3 and Lemma 3.4. This contradicts the assumption that (E, F) is minimal. □

In the above theorem, the first condition selects (possibly non-minimal) explanations from pre-explanations of G which are computed by the consistent answer sets of $UP \cup \{ \leftarrow not\, G \}$ (Lemma 3.4). Then, the second condition selects minimal ones from those explanations.

When a program is a *locally stratified* NLP [14], it has at most one answer set (called a *perfect model*). Then, Theorem 3.5 is simplified as follows.

Corollary 3.6 *Let $\langle P, \mathcal{A} \rangle$ be an abductive program in which P is a locally stratified NLP, and UP its update program. Given a ground atom G, $(P \cup E) \setminus F$ accomplishes the insertion of G iff the program $UP \cup \{ \leftarrow not\, G \}$ has a consistent U-minimal answer set S s.t. $E^+ = S \cap \mathcal{UA}^+$ and $F^- = S \cap \mathcal{UA}^-$.*

Proof. When P is a locally stratified NLP, so is $(P \cup E) \setminus F$. Then $(P \cup E) \setminus F$ has at most one answer set. Hence, the result holds by Lemma 3.4. □

For deletion, the next result holds for any abductive program.

Theorem 3.7. *Let $\langle P, \mathcal{A} \rangle$ be an abductive program, UP its update program, and G a ground literal. Then, $(P \cup E) \setminus F$ accomplishes the deletion of G iff $UP \cup \{ \leftarrow G \}$ has a consistent U-minimal answer set S s.t. $E^+ = S \cap \mathcal{UA}^+$ and $F^- = S \cap \mathcal{UA}^-$.*

Proof. Deletion is done as $(P \cup E) \setminus F \not\models G$
iff (E, F) is a minimal anti-explanation of G wrt $\langle P, \mathcal{A} \rangle$
iff (E, F) is a minimal pre-explanation of G' wrt $\langle P \cup \{ G' \leftarrow not\, G \}, \mathcal{A} \rangle$ where G' is an atom appearing nowhere in P
iff $UP \cup \{ G' \leftarrow not\, G \} \cup \{ \leftarrow not\, G' \}$ has a consistent U-minimal answer set $S \cup \{ G' \}$ s.t. $E^+ = S \cap \mathcal{UA}^+$ and $F^- = S \cap \mathcal{UA}^-$ (by Lemma 3.4)
iff $UP \cup \{ \leftarrow G \}$ has a consistent U-minimal answer set S s.t. $E^+ = S \cap \mathcal{UA}^+$ and $F^- = S \cap \mathcal{UA}^-$. □

3.2 Updates with Rules

In view updates, if one wants to insert/delete not only facts but also rules, it is done in the following manner.

Let $\langle P, \mathcal{A} \rangle$ be an abductive program, in which both P and $\mathcal{A}$ are ELPs. The rules in $\mathcal{A}$, called *abducible rules*, are hypothetical rules that are used for abducing an observation together with the background knowledge from P [10]. Then, abductive programs introduced in Section 2 are considered as a special case where each rule in $\mathcal{A}$ is a fact.

In this extended framework, an (anti-)explanation for an observation is defined as in Section 2 with the only difference that it is given as a pair (E, F) where E and F are sets of abducible rules.

Example 3.2. Let $\langle P, \mathcal{A} \rangle$ be an abductive program such that

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

Then, the observation $G = \neg flies(tweety)$ has an explanation
$(E, F) = (\{\, \neg flies(x) \leftarrow penguin(x) \,\}, \{\, flies(x) \leftarrow bird(x) \,\})$.

An abductive program with abducible rules is transferable to an abductive program with abducible facts [10]. Given an abductive program with abducible rules $\langle P, \mathcal{A} \rangle$, let $\mathcal{R} = \{\, H \leftarrow B \mid (H \leftarrow B) \in \mathcal{A} \text{ and } B \neq \emptyset \,\}$. Then, define

$$\begin{aligned} P' &= (P \setminus \mathcal{R}) \cup \{\, H \leftarrow B, \gamma_R, \quad \gamma_R \leftarrow \mid R = (H \leftarrow B) \in \mathcal{R} \cap P \,\}, \\ &\qquad \cup \{\, H \leftarrow B, \gamma_R \mid R = (H \leftarrow B) \in \mathcal{R} \setminus P \,\}, \\ \mathcal{A}' &= (\mathcal{A} \setminus \mathcal{R}) \cup \{\, \gamma_R \mid R \in \mathcal{R} \,\}, \end{aligned}$$

where γ_R is a newly introduced atom uniquely associated with each abducible rule R in $\mathcal{R}$. The produced abductive program $\langle P', \mathcal{A}' \rangle$ is semantically equivalent to the original abductive program $\langle P, \mathcal{A} \rangle$.

Example 3.3. The abductive program of Example 3.2 is transformed to an abductive program $\langle P', \mathcal{A}' \rangle$ where

$$\begin{aligned} P':\ & flies(x) \leftarrow bird(x),\ \gamma_1(x), \\ & bird(x) \leftarrow penguin(x), \\ & \neg flies(x) \leftarrow penguin(x),\ \gamma_2(x). \\ & penguin(tweety) \leftarrow, \quad \gamma_1(x) \leftarrow, \\ \mathcal{A}':\ & \gamma_1(x),\ \gamma_2(x). \end{aligned}$$

Here, $\gamma_1(x)$ and $\gamma_2(x)$ are newly introduced abducibles associated with the rules $flies(x) \leftarrow bird(x)$ and $\neg flies(x) \leftarrow penguin(x)$, respectively. In this program, $G = \neg flies(tweety)$ has an explanation $(\{\, \gamma_2(x) \,\}, \{\, \gamma_1(x) \,\})$, which corresponds to the explanation (E, F) of Example 3.2.

With this transformation, the technique of view updates in Section 3.1 is directly applied to view updates with abducible rules.

4 Theory Updates

4.1 Update with Programs

Next we consider the situation that new information arrives at a knowledge base in which the whole knowledge is subject to change. Given a program P which represents the current knowledge base and another program Q which represents new information, a theory update should satisfy the following conditions.

Definition 4.1. Given programs P and Q, P' accomplishes a *theory update* of P by Q if

1. P' is consistent,
2. $Q \subseteq P' \subseteq P \cup Q$,
3. there is no consistent program P'' s.t. $P' \subset P'' \subseteq P \cup Q$.

By definition, the updated program P' is defined as the union of the new information Q and a maximal subset of the original program P which is consistent with Q. The first condition also implies that new information Q should be consistent, namely, updating with inconsistent information makes no sense.

To realize theory updates, an abductive framework is used for specifying priorities between the current knowledge and the new knowledge. Consider the abductive program $\langle\, P \cup Q, P \setminus Q \,\rangle$, where a program is given as $P \cup Q$ and any rule in the original program P other than the new information Q is specified as variable abducible rules.

Proposition 4.1 *Let $\langle\, P \cup Q, P \setminus Q \,\rangle$ be an abductive program. Then, P' accomplishes a theory update of P by Q iff $P' = (P \cup Q) \setminus F$ where $(\emptyset, F)$ is a minimal anti-explanation of the observation $G = false$ wrt $\langle\, P \cup Q, P \setminus Q \,\rangle$.*

Proof. P' accomplishes a theory update of P by Q
iff $P' = (P \cup Q) \setminus F$ where F is a minimal set s.t. $F \subseteq P \setminus Q$ and $(P \cup Q) \setminus F \not\models false$
iff $P' = (P \cup Q) \setminus F$ where $(\emptyset, F)$ is a minimal anti-explanation of the observation $G = false$ wrt $\langle\, P \cup Q, P \setminus Q \,\rangle$. □

The abductive program $\langle\, P \cup Q, P \setminus Q \,\rangle$ is transformed to an abductive program with abducible facts using the naming technique of Section 3.2. Then, we can compute theory updates using update programs and U-minimal answer sets introduced in Section 3.1. When UP is an update program of $\langle\, P \cup Q, P \setminus Q \,\rangle$, a minimal anti-explanation of $G = false$ is computed by a consistent U-minimal answer set of $UP \cup \{ \leftarrow false \}$ (Theorem 3.7). Here, the constraint $\leftarrow false$ imposes consistency on UP, then the above problem is equivalent to computing a consistent U-minimal answer set of UP.

Example 4.1. [2] Given the current knowledge base

$$\begin{aligned} P_1 : \; & sleep \leftarrow not\ tv\_on\,, \\ & watch\_tv \leftarrow tv\_on\,, \\ & tv\_on \leftarrow\,, \end{aligned}$$

we want to update P_1 with[7]

$$P_2:\ \neg tv\_on \leftarrow power\_failure\,,$$
$$power\_failure \leftarrow\ .$$

The situation is expressed by the abductive program $\langle\, P_1 \cup P_2, P_1 \setminus P_2 \,\rangle$, The update program UP of $\langle\, P_1 \cup P_2, P_1 \setminus P_2 \,\rangle$ then becomes

$$\begin{aligned} UP:\ & \neg tv\_on \leftarrow power\_failure\,,\\ & power\_failure \leftarrow,\\ & sleep \leftarrow not\ tv\_on,\ \gamma_1\,,\quad watch\_tv \leftarrow tv\_on,\ \gamma_2\,,\\ & abd(tv\_on),\quad abd(\gamma_1),\quad abd(\gamma_2),\\ & -tv\_on \leftarrow not\ tv\_on\,,\quad -\gamma_1 \leftarrow not\ \gamma_1\,,\quad -\gamma_2 \leftarrow not\ \gamma_2\,. \end{aligned}$$

where γ_1 and γ_2 are the names of the abducible rules in P_1. Then, UP has the unique U-minimal answer set $\{\, power\_failure,\ \neg tv\_on,\ sleep,\ \overline{tv\_on},\ -tv\_on,\ \gamma_1,\ \gamma_2 \,\}$, which represents the deletion of the fact tv_on from $P_1 \cup P_2$. As a result, the theory update of P_1 by P_2 becomes $P_3 = (P_1 \cup P_2) \setminus \{\, tv\_on \,\}$.

Next, suppose that another update

$$P_4:\ \neg power\_failure \leftarrow$$

is given to P_3 which states that power is back again. The situation is expressed by the abductive program $\langle\, P_3 \cup P_4, P_3 \setminus P_4 \,\rangle$, and its update program becomes

$$\begin{aligned} UP:\ & \neg power\_failure \leftarrow,\\ & sleep \leftarrow not\ tv\_on\,, \gamma_1\,,\quad watch\_tv \leftarrow tv\_on\,, \gamma_2\,,\\ & \neg tv\_on \leftarrow power\_failure,\ \gamma_3\,,\\ & abd(power\_failure),\quad abd(\gamma_i)\ (i=1,2,3),\\ & -power\_failure \leftarrow not\ power\_failure,\quad -\gamma_i \leftarrow not\ \gamma_i\ (i=1,2,3)\,. \end{aligned}$$

Then, UP has the unique U-minimal answer set $\{\, \neg power\_failure,\ sleep,\ \gamma_1,\ \gamma_2,\ \gamma_3,\ \overline{power\_failure},\ -power\_failure \,\}$, which implies that the result of an update is $(P_3 \cup P_4) \setminus \{\, power\_failure \,\}$.[8]

Note that in $\langle\, P \cup Q, P \setminus Q \,\rangle$ it holds that $(P \setminus Q) \setminus (P \cup Q) = \emptyset$, so UP contains no rule of the form $+a \leftarrow a$ of Definition 3.2(2). The above observation is formally stated as follows.

Theorem 4.2. *Let P and Q be programs, and UP the update program of $\langle\, P \cup Q, P \setminus Q \,\rangle$. Then, $(P \cup Q) \setminus F$ accomplishes a theory update of P by Q iff UP has a consistent U-minimal answer set S and F is the set of rules whose names are in $S \cap \mathcal{UA}$.* [9]

Proof. The result holds by Proposition 4.1 and Theorem 3.7. □

[7] In [2] the rule $\neg tv\_on \leftarrow power\_failure$ is given as $not\ tv\_on \leftarrow power\_failure$.

[8] If the rule $tv\_on \leftarrow not\ power\_failure$ is in the program, tv_on becomes true as in [2].

[9] For convenience, we consider that an abducible fact $a \leftarrow$ has the name a.

4.2 Inconsistency Removal

When a program contains inconsistent information, it must be updated to recover consistency. This type of updates is called an inconsistency removal.

Definition 4.2. Let P be an inconsistent program. Then, a program P' accomplishes an *inconsistency removal* of P if

1. P' is consistent,
2. $P' \subseteq P$,
3. there is no consistent program P'' s.t. $P' \subset P'' \subset P$.

By definition, inconsistency removal is captured as a special case of theory updates where P is inconsistent and Q is empty in Definition 4.1. By putting $Q = \emptyset$ in $\langle P \cup Q, P \setminus Q \rangle$, inconsistency removal is characterized by the abductive program $\langle P, P \rangle$. The next proposition directly follows from Proposition 4.1.

Proposition 4.3 *Let $\langle P, P \rangle$ be an abductive program. Then, P' accomplishes an inconsistency removal of P iff $P' = P \setminus F$ where $(\emptyset, F)$ is a minimal anti-explanation of the observation $G = false$ wrt $\langle P, P \rangle$.*

Example 4.2. Let $P = \{ p \leftarrow not\, p, \quad q \leftarrow \}$. Then, $G = false$ has the minimal anti-explanation $(E, F) = (\emptyset, \{ p \leftarrow not\, p \})$ wrt $\langle P, P \rangle$. As a result, $P' = \{ q \leftarrow \}$ accomplishes an inconsistency removal of P.

The following result holds by Theorem 4.2.

Theorem 4.4. *Let P be an inconsistent program and UP the update program of $\langle P, P \rangle$. Then, $P \setminus F$ accomplishes inconsistency removal of P iff UP has a consistent U-minimal answer set S and F is the set of rules whose names are in $S \cap \mathcal{UA}$.*

Example 4.3. Let P be the program

$$
\begin{aligned}
& pacifist \leftarrow quaker\,, \\
& \neg pacifist \leftarrow republican\,, \\
& quaker \leftarrow, \quad republican \leftarrow,
\end{aligned}
$$

which has the answer set $\mathcal{L}_P$. Then, consider the update program UP of an abductive program $\langle P, P \rangle$:

$$
\begin{aligned}
UP: \; & pacifist \leftarrow quaker, \gamma_1\,, \\
& \neg pacifist \leftarrow republican, \gamma_2\,, \\
& abd(\gamma_1), \;\; abd(\gamma_2), \;\; abd(quaker), \;\; abd(republican), \\
& -\gamma_1 \leftarrow not\, \gamma_1, \quad -\gamma_2 \leftarrow not\, \gamma_2, \\
& -quaker \leftarrow not\, quaker, \quad -republican \leftarrow not\, republican\,.
\end{aligned}
$$

Then, UP has four U-minimal answer sets: $\{ quaker, republican, pacifist, \gamma_1, \overline{\gamma_2}, -\gamma_2 \}$, $\{ quaker, republican, \neg pacifist, \overline{\gamma_1}, \gamma_2, -\gamma_1 \}$, $\{ \overline{quaker}, republican, \gamma_1, \gamma_2, \neg pacifist, -quaker \}$, $\{ quaker, \overline{republican}, pacifist, \gamma_1, \gamma_2, -republican \}$, which represent that deletion of one of the rules (or facts) from P makes the program consistent.

5 Computational Complexity

In this section, we analyze the computational complexity of updating ELPs. Throughout the section, we consider propositional abductive programs, i.e., an abductive program $\langle P, \mathcal{A} \rangle$ where P is a finite ELP containing no variable and $\mathcal{A}$ is a finite set of ground literals.

We first present the complexity of extended abduction. To this end, we introduce transformations between extended abduction and normal abduction. Let $\langle P, \mathcal{A} \rangle$ be an abductive program, G an observation, $E \subseteq \mathcal{A} \setminus P$ and $F \subseteq \mathcal{A} \cap P$. It holds that $(P \cup E) \setminus F \models G$ iff $(P \setminus \mathcal{A}) \cup E \cup ((P \cap \mathcal{A}) \setminus F) \models G$. Then, G has an explanation (E, F) wrt $\langle P, \mathcal{A} \rangle$ (under extended abduction) iff G has an explanation $H = E \cup ((P \cap \mathcal{A}) \setminus F)$ wrt $\langle P \setminus \mathcal{A}, \mathcal{A} \rangle$ (under normal abduction). Here, (E, F) is easily extracted from H as $E = H \cap (\mathcal{A} \setminus P)$ and $F = (P \cap \mathcal{A}) \setminus H$.

In converse, put $P' = P \cup \{ \leftarrow \mathit{not}\, a \mid a \in \mathcal{A} \cap P \}$. Then, G has an explanation E wrt $\langle P, \mathcal{A} \rangle$ (under normal abduction) iff G has an explanation $(E, \emptyset)$ wrt $\langle P', \mathcal{A} \rangle$ (under extended abduction).

On the other hand, G has an anti-explanation (E, F) wrt $\langle P, \mathcal{A} \rangle$ iff an observation G' has a pre-explanation (E, F) wrt $\langle P \cup \{ G' \leftarrow \mathit{not}\, G \}, \mathcal{A} \rangle$ (Theorem 3.7). Using the above transformations between extended abduction and normal abduction, it is shown that any pre-explanation (E, F) of G' is efficiently computed by normal abduction, and vice-versa.

In what follows, the problem of *finding* an (anti-)explanation (resp. minimal (anti-)explanation) of an observation G means that producing a pair (E, F) of abducibles and checking whether it is an (anti-)explanation (resp. minimal (anti-)explanation) of G.

Theorem 5.1. *Given a propositional abductive program $\langle P, \mathcal{A} \rangle$ and a ground literal G:*

(a) Finding an explanation (resp. minimal explanation) of an observation G is Σ_2^P-complete (resp. Σ_3^P-complete).
(b) Finding an anti-explanation (resp. minimal anti-explanation) of an observation G is NP-complete (resp. Σ_2^P-complete).

Proof. By the polynomial-time transformations between normal abduction and extended abduction, the complexity class of a decision problem under extended abduction is the same as that of the corresponding problem under normal abduction. Then the results of (a) and (b) follow from the complexity results of normal abduction [7]. □

The above results imply the complexity of updating ELPs.

Theorem 5.2. *(a) Finding a program which accomplishes a given view update is Σ_3^P-complete for insertion and Σ_2^P-complete for deletion.*
(b) Finding a program which accomplishes a given theory update is Σ_2^P-complete.
(c) Finding a program which accomplishes inconsistency removal is Σ_2^P-complete.

Proof. The problem of finding a program which accomplishes the view update for inserting (resp. deleting) G is equivalent to the problem of finding a minimal explanation (resp. minimal anti-explanation) of G. On the other hand, the problem of finding a program which accomplishes a theory update or inconsistent removal is equivalent to the problem of finding a minimal anti-explanation of $G = false$. Then, the results hold by Theorem 5.1. □

To conclude, in view update insertion is generally harder than deletion by one level of the polynomial hierarchy, while theory updates and inconsistency removal are as hard as deletion in view updates.

6 Related Work

It is well-known that abduction is used for view updates in deductive databases. Kakas and Mancarella [13] characterize view updates through abduction and use Eshghi and Kowalski's abductive procedure for computation. Decker [6] introduces an abductive procedure for view updates and integrity maintenance. These procedures are top-down and the correctness is guaranteed for locally stratified programs. Generally, view updates based on an SLDNF-like top-down procedure have restrictions on the program syntax. By contrast, the proposed technique is applicable to any ELP and updates are computed by a procedure for computing answer sets of ELPs with the additional task of selecting the U-minimal answer sets. Bry [3] characterizes *intensional updates* through abduction and specifies update procedures in a meta-program that is computed in a bottom-up manner. Console *et al.* [4] provide an abductive procedure for view updates based on Clark's completion. They realize view updates based on traditional abduction but do not consider theory updates. Update programs can compute explanations for extended abduction. Inoue and Sakama [12] introduce *transaction programs* for computing extended abduction, while its application is limited to acyclic abductive programs.

Fagin *et al.* [8] discuss view updates and theory updates in the context of first-order theories. Inoue and Sakama [11] characterize view updates in normal logic programs and theory updates of [8] in terms of extended abduction. Update programs proposed in this paper provide methods for computing them. Further, theory updates considered in this paper are more general than those of Fagin *et al.* in the sense that they consider updating a theory with a single formula. Alferes and Pereira [1] introduce update programs for updating normal and extended logic programs. Alferes *et al.* [2] propose a framework of *dynamic logic programming* which performs theory updates for *generalized logic programs* containing negation as failure in the head. Our approach is different from these two works on the point that we formulate updates in terms of abductive programs and update programs are used for computing view updates as well as theory updates. Theory updates with ELPs are also discussed in [16]. They associate priorities to resolve conflicts between rules, which is different from our abductive approach. Damásio and Pereira [5] resolve inconsistency in ELPs by changing the truth-value of abducibles under the three-valued well-founded semantics.

7 Summary

This paper introduced methods for computing updates in extended logic programs. Three different types of updates, view updates, theory updates, and inconsistency removal were characterized in terms of abductive programs. All these updates were computed using update programs. The results of this paper show that abduction can play a fundamental role in various update problems. In future study we will extend the techniques to knowledge bases containing disjunctions.

References

1. J. J. Alferes and L. M. Pereira, Update programs can update programs, *Nonmonotonic Extensions of Logic Programming, Lecture Notes in Artificial Intelligence* 1216, pages 110–131, Springer, 1997.
2. J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinska, and T. Przymusinski. Dynamic logic programming. In: *Proc. 6th Int'l Conf. Principles of Knowledge Representation and Reasoning*, pages 98–109, Morgan Kaufmann, 1998.
3. F. Bry. Intensional updates: abduction via deduction. In: *Proc. 7th Int'l Conf. Logic Programming*, pages 561–575, MIT Press, 1990.
4. L. Console, M. L. Sapino, and D. T. Dupré. The role of abduction in database view updating. *J. Intelligent Information Systems* 4:261–280, 1995.
5. C. V. Damásio and L. M. Pereira. Abduction over 3-valued extended logic programs. In: *Proc. LPNMR'95, Lecture Notes in Artificial Intelligence* 928, pages 29–42.
6. H. Decker. An extension of SLD by abduction and integrity maintenance for view updating in deductive databases. In: *Proc. 1996 Joint Int'l Conf.& Symp. Logic Programming*, pages 157–169, MIT Press.
7. T. Eiter, G. Gottlob, and N. Leone, Abduction from logic programs: semantics and complexity, *Theoretical Computer Science* 189(1-2):129–177, 1997.
8. R. Fagin, J. D. Ullman, and M. Y. Vardi. On the semantics of updates in databases (preliminary report). In: *Proc. 2nd ACM SIGACT-SIGMOD Symp. Principles of Database Systems*, pages 352–365, 1983.
9. M. Gelfond and V. Lifschitz. Logic programs with classical negation. In: *Proc. 7th Int'l Conf. Logic Programming*, pages 579–597, MIT Press, 1990.
10. K. Inoue. Hypothetical reasoning in logic programs. *J. Logic Programming* 18:191-227, 1994.
11. K. Inoue and C. Sakama. Abductive framework for nonmonotonic theory change. In: *Proc. IJCAI-95*, pages 204–210, Morgan Kaufmann.
12. K. Inoue and C. Sakama. Specifying transactions for extended abduction. In: *Proc. 6th Int'l Conf. Principles of Knowledge Representation and Reasoning*, pages 394–405, Morgan Kaufmann, 1998.
13. A. C. Kakas and P. Mancarella. Database updates through abduction. In: *Proc. 16th Int'l Conf. Very Large Databases*, pages 650–661, Morgan Kaufmann, 1990.
14. T. C. Przymusinski. On the declarative semantics of deductive databases and logic programs. In: J. Minker, (ed.), *Foundations of Deductive Databases and Logic Programming*, pages 193–216, Morgan Kaufmann, 1998.
15. K. Satoh and K. Iwayama. Computing abduction by using the TMS. In: *Proc. 8th Int'l Conf. Logic Programming*, pages 505–518, MIT Press, 1991.
16. Y. Zhang and N. Y. Foo. Updating logic programs. In: *Proc. 13th European Conf. Artificial Intelligence*, pages 403-407, Wiley, 1998.