Updating Action Domain Descriptions*

Thomas Eiter, Esra Erdem, Michael Fink, and **Ján Senko**
Institute of Information Systems, Vienna University of Technology, Vienna, Austria
Email: (eiter | esra | michael | jan)@kr.tuwien.ac.at

Abstract

How can an intelligent agent update her knowledge base about an action domain, relative to some conditions (possibly obtained from earlier observations)? We study this question in a formal framework for reasoning about actions and change, in which the meaning of an action domain description can be represented by a directed graph whose nodes correspond to states and whose edges correspond to action occurrences. We define the update of an action domain description in this framework, and show among other results that a solution to this problem can be obtained by a divide-and-conquer approach in some cases. We also introduce methods to compute a solution and an approximate solution to this problem, and analyze the computational complexity of these problems. Finally, we discuss techniques to improve the quality of solutions.

1 Introduction

This paper discusses how an intelligent agent can modify her knowledge of the world, subject to some given conditions, in particular, when some new knowledge about the world is added. The given conditions can be domain-dependent (e.g., assertions obtained from earlier observations), or domain-independent (e.g., general properties of a knowledge base). The goal is to obtain a consistent knowledge base with minimal changes only to a variable part of the existing knowledge base such that the given conditions are satisfied.

We study the problem of knowledge base updates, briefly described above, in the context of reasoning about actions and change, so the agent's knowledge base is about domains of actions. For instance, consider an agent having the following knowledge, D, about a TV with remote control:

1. If the power is off, pushing the power button on the TV turns the power on.
2. If the power is on, pushing the power button on the TV turns the power off.
3. The TV is on whenever the power is on.
4. The TV is off whenever the power is off.

*Work supported by the Austrian Science Fund (FWF) grant P16536-N04 and the European Commission IST programme grants FET-2001-37004 WASP and IST-2001-33123 CologNeT.

Note that she does not know how a remote control works (e.g., she does not know the effect of pushing the power button on the remote control). Suppose that later she obtains the following information, Q, about remote controls:

1. If the power is on and the TV is off, pushing the power button on the remote control turns the TV on.
2. If the TV is on, pushing the power button on the remote control turns the TV off.

A desired condition, C, for the effective use of this TV system is the following:

> Pushing the power button on the remote control is always possible.

If Q is added to D, then condition C is not satisfied: when the power and the TV are on, pushing the power button on the remote control is not possible, since Item 2 of Q and Item 3 of D above contradict. The question is then how the agent can update D by Q relative to C.

We describe domains of actions, like D above, in a fragment of the action language $\mathcal{C}$ [Giunchiglia and Lifschitz, 1998], by "causal laws." For instance, the direct effect of the action $PushPB_{TV}$ stated in Item 1 of D in the example above is described by the causal law

$$\textbf{caused } PowerON \textbf{ after } PushPB_{TV} \wedge \neg PowerON,$$

which expresses that the action $PushPB_{TV}$ causes the value of the fluent $PowerON$ to change from false to true; the indirect effect of this action stated in Item 3 of D is described by the causal law

$$\textbf{caused } TvON \textbf{ if } PowerON, \tag{1}$$

which expresses that if the fluent $PowerON$ is caused to be true, then the fluent $TvON$ is caused to be true as well. An action (domain) description is partitioned into two: D_u (invariable) and D_m (variable). The meaning of an action description can be represented by a "transition diagram"—a directed graph whose nodes correspond to states and whose edges correspond to action occurrences.

We describe given conditions, like C above, in an action query language, similar to $\mathcal{R}$ [Gelfond and Lifschitz, 1998], by "queries". For instance, condition C above can be described by the query

$$\textbf{executable } \{PushPB_{RC}\}. \tag{2}$$

The main contributions of this paper briefly are as follows:

• Given an action description $D = (D_u, D_m)$, a set Q of causal laws, and a set C of conditions, what is an update of D by Q relative to C? We answer this question in Section 3.

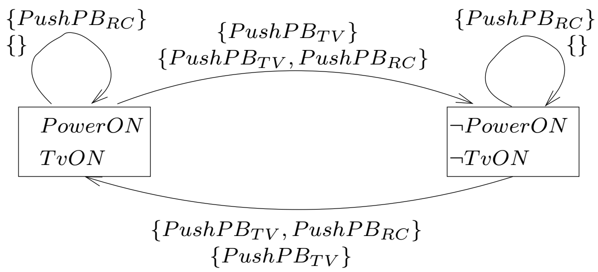

Figure 1: A transition diagram.

• Given an action description $D = (D_u, D_m)$, a set Q of causal laws, and a set C of conditions, consider the problem of computing an update of D by Q relative to C. We show that this problem is difficult in general (both Σ_3^P- and Π_3^P-hard), and that already checking a given solution is Π_3^P-complete. We also show that, under some conditions, these two problems are easier (Section 5).

• We provide conditions under which computing a solution to the problem above can be structurally decomposed. For instance, if the inputs of the problem satisfy the disjointness property, a divide-and-conquer approach is feasible (Section 4). We introduce methods to compute solutions and to compute "near-solutions" approximating them (Section 6).

Variations of the problem of updating knowledge bases have been studied in the AI literature, e.g., [Winslett, 1990; Katsuno and Mendelzon, 1991; Pereira *et al.*, 1991; Li and Pereira, 1996; Liberatore, 2000; Eiter *et al.*, 2002; Sakama and Inoue, 2003]. The problem addressed in this paper is more general (see Section 7 for a more detailed comparison).

2 Preliminaries

Transition diagrams. We start with a *(propositional) action signature* that consists of a set $\mathbf{F}$ of fluent names, and a set $\mathbf{A}$ of action names. An *action* is a truth-valued function on $\mathbf{A}$, denoted by the set of action names that are mapped to t.

A *(propositional) transition diagram* of an action signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$ consists of a set S of *states*, a function $V : \mathbf{F} \times S \rightarrow \{f, t\}$, and a subset $R \subseteq S \times 2^{\mathbf{A}} \times S$ of *transitions*. We say that $V(P, s)$ is the *value* of P in s. The states s' such that $\langle s, A, s' \rangle \in R$ are the possible *results of the execution* of the action A in the state s. We say that A is *executable* in s, if at least one such state s' exists.

A transition diagram can be thought of as a labeled directed graph. Every state s is represented by a vertex labeled with the function $P \mapsto V(P, s)$ from fluent names to truth values. Every triple $\langle s, A, s' \rangle \in R$ is represented by an edge leading from s to s' and labeled A. An example of a transition diagram is shown in Figure 1.

Action description languages. We consider a subset of the action description language $\mathcal{C}$ [Giunchiglia and Lifschitz, 1998] that consists of two kinds of expressions (called *causal laws*): *static laws* of the form

$$\textbf{caused } L \textbf{ if } G, \qquad (3)$$

where L is a literal (an expression of the form P or $\neg P$, where P is a fluent name) and G is a propositional combination of fluent names, and *dynamic laws* of the form

$$\textbf{caused } L \textbf{ if } G \textbf{ after } U, \qquad (4)$$

caused $PowerON$ **after** $PushPB_{TV} \wedge \neg PowerON$
caused $\neg PowerON$ **after** $PushPB_{TV} \wedge PowerON$
caused $TvON$ **if** $PowerON$
caused $\neg TvON$ **if** $\neg PowerON$
inertial $PowerON, \neg PowerON$
inertial $TvON, \neg TvON$

Figure 2: An action description.

where L and G are as above, and U is a propositional combination of fluent names and action names. In (3) and (4) the part **if** G can be dropped if G is $True$.

An *action description* is a set of causal laws. For instance, the knowledge base about a TV system, D, of the agent in the previous section, can be described by causal laws in Figure 2. Thereby, an expression of the form:

$$\textbf{inertial } P, \neg P,$$

where P is a fluent name stands for the dynamic laws

$$\textbf{caused } P \textbf{ if } P \textbf{ after } P,$$
$$\textbf{caused } \neg P \textbf{ if } \neg P \textbf{ after } \neg P,$$

describing that the value of the fluent P stays the same unless some action changes it.

The meaning of an action description can be represented by a transition diagram. Let D be an action description with a signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$. Then the transition diagram $\langle S, V, R \rangle$ *described* by D is defined as follows:

(*i*) S is the set of all interpretations s of $\mathbf{F}$ such that, for every static law (3) in D, s satisfies $G \supset L$,

(*ii*) $V(P, s) = s(P)$,

(*iii*) R is the set of all triples $\langle s, A, s' \rangle$ such that s' is the only interpretation of $\mathbf{F}$ which satisfies the heads L of all
   – static laws (3) in D for which s' satisfies G, and
   – dynamic laws (4) in D for which s' satisfies G and $s \cup A$ satisfies U.

The laws included in (*iii*) above are those that are *applicable* to the transition from s to s' caused by executing A. For instance, the transition diagram described by the action description of Figure 2 is presented in Figure 1. We say that an action description is *consistent* if it can be represented by a transition diagram with nonempty state set.

In the following we suppose that an action description D consists of two parts: D_u (unmodifiable causal laws) and D_m (modifiable causal laws). Therefore, we sometimes denote an action description D by the pair (D_u, D_m).

Action query languages. To talk about observations of the world, or assertions about the effects of the execution of actions, we use an action query language similar to $\mathcal{R}$ [Gelfond and Lifschitz, 1998]. We consider a language with *static queries* of one of the two forms:

$$\textbf{holds } F, \qquad (5)$$
$$\textbf{always } F, \qquad (6)$$

where F is a propositional combination of fluent names, and *dynamic queries* of the form:

$$\textbf{necessarily } F \textbf{ after } A_0; \ldots; A_n, \qquad (7)$$

where F is a propositional combination of fluent names, and each A_i ($0 \leq i \leq n$) is an action. A *query* is a propositional combination of both kinds of expressions.

As for the semantics, let $T = \langle S, V, R \rangle$ be the transition diagram representing an action description D. A *history* of T of length n is a sequence

$$s_0, A_1, s_1, \ldots, s_{n-1}, A_n, s_n \quad (8)$$

where each $\langle s_i, A_{i+1}, s_{i+i} \rangle$ ($0 \leq i < n$) is a transition in R. The satisfaction relation between a state s in S and a query q relative to D (denoted $D, s \models q$) is defined as follows:

- (*i*) if q is a static query (5), it is *satisfied* at s if the interpretation $P \mapsto V(P, s)$ satisfies F;
- (*ii*) if q is a static query (6), it is *satisfied* at s if, for each state $s' \in S$, the interpretation $P \mapsto V(P, s')$ satisfies F;
- (*iii*) if q is a dynamic query (7), it is *satisfied* at s if, for every history $s = s_0, A_0, s_1, \ldots, s_{n-1}, A_n, s_n$ of T of length $n+1$, the interpretation $P \mapsto V(P, s_n)$ satisfies F;
- (*iv*) for any other q, *satisfaction* is defined by the truth tables of propositional logic.

We remark that **holds** $F \equiv \neg$**holds** $\neg F$, but not so for a static query of form (6), i.e., **always** $F \not\equiv \neg$**always** $\neg F$.

For a set C of queries, we denote by S^D_C the set of all states $s \in S$ such that $D, s \models q$ for every $q \in C$, and by $S^D_{\neg C}$ the set $S \setminus S^D_C$. We say that D *entails* C (denoted $D \models C$) if $S = S^D_C$. For instance, a set of queries containing the query

$$\textbf{necessarily}\ \neg\, TvON\ \textbf{after}\ \{PushPB_{RC}\}$$

is not entailed by the action description in Figure 2, because the query above is not satisfied at the state where both fluents $TvON$ and $PowerON$ are mapped to t. On the other hand, this action description entails the queries:

$$\textbf{always}\ PowerOn \equiv TvON, \quad (9)$$

$$\textbf{holds}\ PowerON \wedge TvON \supset \neg\textbf{necessarily}\ \neg\, TvON\ \textbf{after}\ \{PushPB_{TV}\}. \quad (10)$$

Queries allow us to express various pieces of knowledge about the domain (possibly acquired from earlier observations), like the following.

• **Existence of certain states:** There exists a state of the world where a formula F holds (assuming consistency of D):

$$\neg\textbf{always}\ \neg F.$$

• **Executability of actions:** An action sequence $A_0, \ldots, A_n$ is executable at every state:

$$\neg\textbf{necessarily}\ False\ \textbf{after}\ A_0; \ldots; A_n.$$

As in (2), this query can be abbreviated as

$$\textbf{executable}\ A_0; \ldots; A_n.$$

• **Mandatory effects of actions:** A sequence $A_0, \ldots, A_n$ of actions will always have certain effects in a given context:

$$\textbf{holds}\ G \supset \textbf{necessarily}\ F\ \textbf{after}\ A_0; \ldots; A_n$$

Here F describes the effects and G the context. For instance, the query

$$\textbf{holds}\ TvON \supset \textbf{necessarily}\ \neg\, TvON\ \textbf{after}\ \{PushPB_{RC}\}$$

expresses that the result of executing $\{PushPB_{RC}\}$ at any state where $TvON$ is mapped to t is a state where $TvON$ is mapped to f. Note that queries of this form allow us to express all possible executions of a conformant plan $\langle A_0, \ldots, A_n \rangle$ to reach a goal specified by F from an initial state described by G.

• **Possible effects of actions:** A sequence $A_0, \ldots, A_n$ of actions may have certain effects in a given context:

$$\textbf{holds}\ G \supset \neg\textbf{necessarily}\ \neg F\ \textbf{after}\ A_0; \ldots; A_n,$$

like in (10). Here F describes the effects and G the context. Queries of this form allow us to express, e.g., the observation that executing $\langle A_0, \ldots, A_n \rangle$ in some initial state described by G leads to a state where F holds (e.g., as for (10), that the TV is on). Formula G may describe a collection of states.

3 Problem Description

We define an *Action Description Update (ADU)* problem by an action description $D = (D_u, D_m)$, a set Q of causal laws, and a set C of queries, all with the same signature. We say that an action description D' *accomplishes an (action description) update* of D by Q relative to C, if

- (*i*) D' is consistent,
- (*ii*) $Q \cup D_u \subseteq D' \subseteq D \cup Q$,
- (*iii*) $D' \models C$,
- (*iv*) there is no consistent action description D'' such that $D' \subset D'' \subseteq D \cup Q$, and $D'' \models C$.

Such a D' is called a *solution* to the ADU problem (D, Q, C).

Condition (*i*) expresses that an action description update modeling a dynamic domain, such as the TV system in Section 1, must have a state. According to Condition (*ii*), new knowledge about the world and the invariable part of the existing action description are kept, and causal laws in the variable part are considered to be either "correct" or "wrong", and in the latter case simply disposed.

Condition (*iii*) imposes semantical constraints C on D', which comprise further knowledge about the action domain gained, e.g., from experience. It is important to note that C can be modified later for another action description update. If Q is a revision (containing more precise knowledge about the domain), then reasonably C should contain all conditions corresponding to observations made about the domain, while other conditions may be kept or dropped; on the other hand, if Q is a change of the world per se, then conditions corresponding to observations might be dropped.

Finally, Condition (*iv*) applies Occam's Razor to select solutions as simple as possible. Simplicity is measured by a smallest set (in terms of inclusion) of causal laws which need to be dropped.

We call any D' which satisfies (*i*)-(*iii*) a *near-solution* of the ADU problem (D, Q, C).

Example 1 For instance, take D the action description in Figure 2, with D_m containing the first four causal laws; consider Q to be the set containing the following two causal laws:

$$\textbf{caused}\ TvON\ \textbf{after}\ PushPB_{RC} \wedge PowerOn \wedge \neg\, TvON,$$
$$\textbf{caused}\ \neg\, TvON\ \textbf{after}\ PushPB_{RC} \wedge TvON,$$

and let C be the set containing queries (2), (10), and

$$\textbf{executable}\ \{PushPB_{TV}\}.$$

The transition diagram described by $D \cup Q$ is shown in Figure 3. Here we can see that at the state where both $PowerON$ and $TvON$ are mapped to t, the action $PushPB_{RC}$ is not executable. Therefore, $D \cup Q$ is not a solution. In fact, a solution to the ADU problem (D, Q, C) is obtained by dropping the static law (1) from $D \cup Q$. □

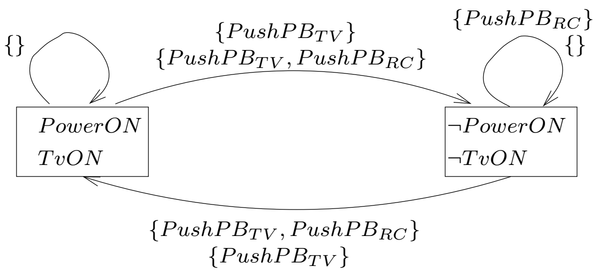

Figure 3: Transition diagram described by $D \cup Q$ of Ex. 1.

4 Properties of Updates

We first consider some basic properties of solutions to an ADU problem. To this end, we define the subsumption relation between an action description D and a causal law, both with the same signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$, as follows:

• A static law (3) is *subsumed* by D if, for every state s in the transition diagram described by D, the interpretation of $\mathbf{F}$ describing the state s satisfies $G \supset L$;

• a dynamic law (4) is *subsumed* by D if, for every transition $\langle s, A, s' \rangle$ in the transition diagram described by D, the following holds: if the interpretation of $\mathbf{F} \cup \mathbf{A}$ describing the state s and the action A satisfies U, then the interpretation of $\mathbf{F}$ describing s' satisfies $G \supset L$.

A set Q of causal laws is *subsumed* by an action description D, if every law in Q is subsumed by D. A causal law is *tautological*, if it is subsumed by every action description D.

Proposition 1 (Subsumption) *Let (D, Q, C) be an ADU problem, such that D is consistent and $D \models C$. If D subsumes Q then $D \cup Q$ is a solution to (D, Q, C).*

From this result, we obtain the following corollaries.

Corollary 1 (Void Update) *Let $(D, \emptyset, C)$ be an ADU problem. If D is consistent and $D \models C$, then D is the unique solution to $(D, \emptyset, C)$.*

Corollary 2 (Idempotence) *Let (D, D, C) be an ADU problem, such that D is consistent and $D \models C$, then D is a solution to (D, D, C).*

Corollary 3 (Addition of Tautologies) *Let (D, Q, C) be an ADU problem, such that D is consistent and $D \models C$. If Q is tautological, i.e., every causal law of Q is a tautology, then $D \cup Q$ is a solution to (D, Q, C).*

Notice that a similar property fails for logic programming updates as in [Alferes *et al.*, 2000; Eiter *et al.*, 2002].

We next consider a structural property of solutions and near-solutions, which can be exploited for a syntactical decomposition of an ADU problem, in a divide-and-conquer manner. Because of the involved semantics of transitions and causation, in general some prerequisites are needed. For any action signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$, we denote by $\mathcal{L}_X$ the part of it which appears in X. We say that an action description D *permits NOP*, if there is some transition $\langle s, \emptyset, s' \rangle$ in the transition diagram described by D. A query F occurs *positively* (resp. *negatively*) in a set C of queries if F occurs in the scope of an even (resp. odd) number of negations, $\neg$, in a query in C.

Theorem 1 (Disjointness) *Let (D, Q, C) be an ADU problem with signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$. Let $\{D^0, D^1\}$, $\{Q^0, Q^1\}$, and $\{C^0, C^1\}$ be partitionings of D, Q, and C, respectively, such that (i) $\mathcal{L}_{D^0} \cap \mathcal{L}_{D^1} = \emptyset$, (ii) $\mathcal{L}_{Q^0} \subseteq \mathcal{L}_{D^0} \wedge \mathcal{L}_{Q^1} \subseteq \mathcal{L}_{D^1}$, and (iii) $\mathcal{L}_{C^0} \subseteq \mathcal{L}_{D^0} \wedge \mathcal{L}_{C^1} \subseteq \mathcal{L}_{D^1}$.*

(i) Let X^i be a near-solution to (D^i, Q^i, C^i), for $i = 0, 1$. Suppose that, for $i = 0, 1$, X^i permits NOP if some dynamic query occurs negatively in C^{1-i}. Then $X^0 \cup X^1$ is a near-solution to (D, Q, C).

(ii) Let X be a near-solution to (D, Q, C), and let $\{X^0, X^1\}$ be a partitioning of X such that $X^0 \subseteq D^0$ and $X^1 \subseteq D^1$. Suppose that, for $i = 0, 1$, X^i permits NOP if some dynamic query occurs positively in C^{1-i}. Then, for $i = 0, 1$, X^i is a near-solution to (D^i, Q^i, C^i).

Informally, the permission of NOP in Theorem 1 is needed to ensure that the transition diagrams of near-solutions to the sub-problems can be "combined". Here are two useful consequences of Proposition 1:

Corollary 4 *Let (D, Q, C) be an ADU problem, and let $\{D^0, D^1\}$, $\{Q^0, Q^1\}$, $\{C^0, C^1\}$ be partitionings of D, Q, C, respectively, satisfying Conditions (i)–(iii) of Theorem 1.*

(i) Suppose that no dynamic query occurs in C. If, for $i = 0, 1$, X^i is a solution to (D^i, Q^i, C^i), then $X^0 \cup X^1$ is a solution to (D, Q, C). Furthermore, every solution to (D, Q, C) can be represented in this form.

(ii) If no dynamic query occurs in C^1, and X^1 is a solution to (D^1, Q^1, C^1) permitting NOP, then, for every solution X^0 to (D^0, Q^0, C^0), $X^0 \cup X^1$ is a solution to (D, Q, C).

Example 2 Consider the ADU problem $(D \cup D', Q, C)$, with D, Q, and C as in Example 1, and D' consisting of the causal laws:

$$\textbf{inertial } \mathit{LightON}, \neg \mathit{LightON}, \quad (11)$$
$$\textbf{caused } \mathit{LightON} \textbf{ after } \mathit{SwitchLight} \wedge \neg \mathit{LightON}, \quad (12)$$
$$\textbf{caused } \neg \mathit{LightON} \textbf{ after } \mathit{SwitchLight} \wedge \mathit{LightON}. \quad (13)$$

Since $X^0 = D \cup Q \setminus \{(1)\}$ is a solution to (D, Q, C) (Example 1), $X^1 = D'$ is a solution to $(D', \emptyset, \emptyset)$, and D' permits NOP, by Corollary 4 (*ii*) $X^0 \cup X^1 = (D \cup D' \cup Q) \setminus \{(1)\}$ is a solution to $(D \cup D', Q, C)$. □

5 Computational Complexity

An important subtask in solving ADU problems is checking whether a set of queries is entailed by an action description.

Theorem 2 *Given an action description D and a set C of queries, deciding $D \models C$ is (i) Π_2^P-complete in general, and (ii) $\mathrm{P}_{\parallel}^{\mathrm{NP}}$-complete if C does not involve dynamic queries.*

Here $\mathrm{P}_{\parallel}^{\mathrm{NP}}$ means polynomial-time with a single parallel evaluation of calls to an NP oracle.

Proof. (Sketch) W.l.o.g., C contains a single query c. As for the membership part of (*ii*), deciding whether $D \models a$ for a static query a of form (5) is polynomial and of form (6) is in coNP. Hence, for a Boolean combination c of such queries $a_1, \ldots, a_n$, deciding whether $D, s \models c$ is feasible in polynomial time with a single parallel evaluation of NP oracles. For (*i*), deciding whether $D, s \not\models d$ for a given state s and a dynamic query d of form (7) is in NP. Thus $D, s \not\models d$ for a general query d is decidable in polynomial time with an NP oracle, which means deciding $D \not\models d$ is in $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2^P$.

Hence, deciding $D \models d$ is in Π_2^P = co-Σ_2^P. The hardness results follow from transformations from deciding whether (*i*) a Boolean combination of Quantified Boolean Formulas (QBFs) of the form $\exists X E_j$, $1 \le j \le m$, is true and (*ii*) a QBF of the form $\forall X \exists Y E$ is true, respectively. □

As for finding solutions, we note that deciding whether $D \cup Q$ is a solution to an ADU problem (D, Q, C) has the same complexity as deciding $D \models C$ in general. Deciding the existence of an arbitrary solution is harder.

Theorem 3 *Deciding if a given ADU problem (D, Q, C) has a solution (or a near-solution) is (i) Σ_3^P-complete in general, (ii) Σ_2^P-complete if C does not involve dynamic queries, and (iii) NP-complete if $C = \emptyset$.*

Proof. (Sketch) Let $D = D_u \cup D_m$. We can guess a near-solution D' such that $D_u \cup Q \subseteq D' \subseteq D_u \cup Q$, along with a state s for D' (to witness consistency), and check $D \models C$ for (*i*) and (*ii*) in polynomial time with the help of (*i*) a Σ_2^P-oracle and (*ii*) an NP-oracle respectively. This proves membership in (*i*) Σ_3^P, (*ii*) Σ_2^P, and (*iii*) NP, respectively. The hardness results follow from transformations from deciding whether (*i*) a QBF of the form $\exists X F$ is true, (*ii*) a QBF of form $\exists Z \forall X F$ is true, and (*iii*) a QBF of the form $\exists Z \forall X \exists Y F$ is true. □

Similarly, testing arbitrary solution candidates has higher complexity than testing $D \cup Q$.

Theorem 4 *Given an ADU problem (D, Q, C) and an action description D', deciding whether D' is a solution for it is (i) Π_3^P-complete in general, (ii) Π_2^P-complete if C does not involve dynamic queries, and (iii) coNP-complete if $C = \emptyset$.*

Hence, even recognizing solutions is quite hard. However, recognizing near-solutions is easier (Π_2^P-complete in general, P^{NP}-complete if C has no dynamic queries, and polynomial if $C = \emptyset$). This follows easily from Theorem 2.

6 Computing Solutions

With an oracle for near-solutions, we can incrementally compute a solution to an ADU problem (D, Q, C) where $D = (D_u, D_m)$, in polynomial time as follows:

1. If $((D_u, D_m), Q, C)$ has a near-solution
   then $D' := \emptyset$ else halt;
2. For each $q_i \in D_m = \{q_1, \ldots, q_m\}$ do
   $D_u^i = D_u \cup D' \cup \{q_i\}$; $D_m^i = \{q_{i+1}, \ldots, q_m\}$;
   if $((D_u^i, D_m^i), Q, C)$ has a near-solution then
   $D' := D' \cup \{q_i\}$;
3. Output $D_u \cup D'$. □

By virtue of Theorems 3 and 4, no substantially better algorithm of this type is likely to exist, since computing a solution is both Σ_3^P- and Π_3^P-hard, and thus needs the power of a Σ_3^P oracle. If a near-solution D_n for (D, Q, C) is known, we may compute, starting from it, a solution in the manner above (assuming $D_n = \{q_1, \ldots, q_k\}$, set $D' = D_n$ and $i = k + 1$).

Near-solutions to a given ADU problem may be nondeterministically computed as in the membership part of Theorem 3, or obtained from a QBF encoding using a QBF solver. We present here a different computation method, using "update descriptions" and "update fluent sets."

Let $D = (D_u, D_m)$ be an action description with signature $\langle \mathbf{F}, \mathbf{A} \rangle$. The *update description* of D is the action description obtained from D as follows:

1. Extend $\langle \mathbf{F}, \mathbf{A} \rangle$ by a set $\mathbf{H}$ of $k = |D_m|$ new fluents (called *update fluents*) $H_1, \ldots, H_k$;
2. label each static law (3) in D_m with a fluent $H_i \in \mathbf{H}$:
   $$\textbf{caused } L \textbf{ if } G \wedge H_i, \tag{14}$$
   and each dynamic law (4) in D_m with a fluent $H_i \in \mathbf{H}$:
   $$\textbf{caused } L \textbf{ if } G \textbf{ after } U \wedge H_i, \tag{15}$$
   such that no two laws are labeled by the same fluent H_i;
3. for each H_i labeling a law, add the dynamic law:
   $$\textbf{inertial } H_i, \neg H_i. \tag{16}$$

Let $D = (D_u, D_m)$ be an action description and C be a set of queries with the same signature. Let E be the update description of D, with a set $\mathbf{H}$ of update fluents. Let $\langle S^E, V^E, R^E \rangle$ be the transition diagram described by E. An *update (fluent) set* for E relative to C is a subset $\mathbf{M}$ of $\mathbf{H}$ such that

(*i*) $\mathbf{M}$ is a hitting set for $S_{\neg C}^E$ (i.e., it contains for each state $s \in S_{\neg C}^E$ some fluent which is true at s);

(*ii*) $\mathbf{H} \setminus \mathbf{M}$ is contained in some state in S_C^E.

With the notions above, we can find a near-solution to an ADU problem (D, Q, C), where $D = (D_u, D_m)$, as follows:

1. If $D \cup Q$ is consistent and $D \cup Q \models C$, output $D \cup Q$.
2. Otherwise, construct the update description E of $D \cup Q$ = $(D_u \cup Q, D_m)$.
3. Take a minimal update set $\mathbf{M}$ for E relative to C, if one exists; otherwise, take $\mathbf{M} = \mathbf{H}$.
4. Identify the set W of causal laws in D_m labeled by the elements of $\mathbf{M}$.
5. Output $D_u \cup (D_m \setminus W) \cup Q$.

Proposition 2 *Let (D, Q, C) be an ADU problem, with $D = (D_u, D_m)$. If there is a nonempty update set for the update description of $D \cup Q = (D_u \cup Q, D_m)$ relative to C, then the method above generates a near-solution to (D, Q, C).*

Example 3 Consider the ADU problem (D, Q, C) presented in Example 1. Note that $D \cup Q \not\models C$ (as explained in Example 1). We obtain the following update description E of $(D_u \cup Q, D_m)$, which contains $D_u \cup Q$ and the laws:

$$\begin{array}{l}
\textbf{caused } \mathit{PowerON} \textbf{ after } \mathit{PushPB}_{TV} \wedge \neg \mathit{PowerON}, H_1, \\
\textbf{caused } \neg \mathit{PowerON} \textbf{ after } \mathit{PushPB}_{TV} \wedge \mathit{PowerON}, H_2, \\
\textbf{caused } \mathit{TvON} \textbf{ if } \mathit{PowerON}, H_3, \\
\textbf{caused } \neg \mathit{TvON} \textbf{ if } \neg \mathit{PowerON}, H_4, \\
\textbf{inertial } H_i, \neg H_i \qquad (1 \le i \le 4).
\end{array}$$

According to the transition diagram described by E,

$$\begin{array}{l}
S_{\neg C}^E = \{\mathit{PowerON}, \mathit{TvON}, H_3\} \times 2^{\mathbf{H}}, \\
S_C^E = \{\{H_1, H_2, H_4\}, \ldots\}.
\end{array}$$

The only minimal update set for E relative to C is $\{H_3\}$. We thus remove from $D \cup Q$ the law labeled by H_3, i.e., (1). The result is a near-solution and also a solution to the problem. □

Example 4 Take D to be the action description of Example 1, with the following causal laws added to D_m:

$$\begin{array}{l}
\textbf{caused } \mathit{TvON} \textbf{ after } \mathit{PushPB}_{TV} \wedge \neg \mathit{PowerON}, \\
\textbf{caused } \neg \mathit{TvON} \textbf{ after } \mathit{PushPB}_{TV} \wedge \mathit{PowerON}.
\end{array}$$

Take Q and C as in Example 1. The transition diagram described by $D \cup Q$ is shown in Figure 3. Due to the same reasons mentioned in Example 1, $D \cup Q \not\models C$. The update description E of $(D_u \cup Q, D_m)$ consists of the update description presented in Example 3 and the causal laws:

caused $TvON$ **after** $PushPB_{TV} \wedge \neg PowerON, H_5$,
caused $\neg TvON$ **after** $PushPB_{TV} \wedge PowerON, H_6$,
inertial $H_i, \neg H_i \qquad (5 \leq i \leq 6)$.

According to the transition diagram described by E,

$$\begin{aligned} S^E_{\neg C} = &\quad \{PowerON, TvON, H_3\} \times 2^{\mathbf{H}} \cup \\ &\quad \{H_4, H_5\} \times 2^{\mathbf{H} \setminus \{H_1\}}, \\ S^E_C = &\quad \{\{H_1, H_2, H_4, H_5, H_6\}, \ldots\}. \end{aligned}$$

The minimal update sets for E relative to C are $\{H_3, H_4\}$ and $\{H_3, H_5\}$. We may choose either one and, by removing from $D \cup Q$ the corresponding two causal laws, we get a near-solution to the problem. None of the near-solutions is however a solution, as removing (1) is sufficient. $\square$

7 Discussion

Related work. Despite conceptional similarities to model based diagnosis, for space reasons, we focus our discussion on more closely related approaches. Updating knowledge bases has been studied extensively in the context of both databases and AI, with different approaches, and in various representation frameworks, e.g., [Winslett, 1990; Katsuno and Mendelzon, 1991; Pereira *et al.*, 1991; Eiter *et al.*, 2002; Sakama and Inoue, 2003; Li and Pereira, 1996; Liberatore, 2000]. The last three are related more closely.

[Sakama and Inoue, 2003] is similar to our work in that it also studies update problems in a nonmonotonic framework (yet in logic programming) and considers the same criterion of minimal change. It deals with three kinds of updates to a knowledge base D: theory update of D by some new information Q, inconsistency removal from D, and view update of $D = (D_u, D_m)$ by some new information Q. In the context of reasoning about actions and change, these kinds of updates are expressible as ADU problems $(D, Q, \emptyset)$, $(D, \emptyset, \emptyset)$, and $((D_u,\, D_m \cup Q), \emptyset, \emptyset)$. Sakama and Inoue show that checking for solution existence is NP-hard for each problem; this complies with Theorem 3 (*iii*). An important difference to [Sakama and Inoue, 2003] is that in an ADU problem (D, Q, C), conditions C may not be directly expressed in D. Moreover, the semantics of an action description D in $\mathcal{C}$ is a transition diagram, and only captured by *all* answer sets of a logic program corresponding to D by known transformations.

[Li and Pereira, 1996] and [Liberatore, 2000] study, like we do, theory update problems in the context of reasoning about actions and change, based on an action language (but language $\mathcal{A}$ instead of $\mathcal{C}$). New information, Q, contains facts describing observations over time (e.g., the action $PushPB_{RC}$ occurs at time stamp 0). The action language $\mathcal{C}$ we use is more expressive than $\mathcal{A}$ in that it accommodates nondeterminism and concurrency, and the changes in the world are not only due to direct effects of actions. To formulate temporal observations, we can extend our query language by queries of the forms

$$E \textbf{ occurs at } t_i, \tag{17}$$

$$P \textbf{ holds at } t_i, \tag{18}$$

where E is an action name, P is a fluent name, and t_i is a time stamp; a state s satisfies a query (17) resp. (18) if, for some history (8) such that $s = s_0$, E is in A_{i+1} resp. s_i satisfies P.

Our notion of consistency of an action description D (in essence, the existence of a state) is different from that of Zhang *et al.* [2002]. They describe action domains in propositional dynamic logic, and require for consistency the existence of some model of an action description. Different from the setting here, conflicting action effects may prevent any model. With the extension of our query language discussed above, other forms of consistency studied in [Zhang *et al.*, 2002] can be achieved in our framework, by describing possible scenarios or formulas as queries.

Repair of action descriptions. We can sometimes improve solutions (and near-solutions) to an ADU problem (D, Q, C) by considering a slightly different version of the problem. We may take the view that a causal law is not completely wrong, and for instance holds in certain contexts. Suppose that Q is a dynamic law of the form:

caused L' **after** $A' \wedge G'$,

where L' is a literal, G' is a propositional combination of fluents, and A' is an action. We can obtain an action description D^* from D, which describes the same transition diagram as D, by replacing each dynamic law (4) in D_m with:

caused L **if** F **after** $U \wedge G'$,
caused L **if** F **after** $U \wedge \neg G'$.

Here, we can observe that for each solution D' to (D, Q, C) some solution ${D^*}'$ to (D^*, Q, C) exists which contains D'. Therefore, such a method can be useful to prevent "complete removal" of some laws from the given action description. Similar methods are also useful for repairing an action description, e.g., if some dynamic laws (4) in the action description have missing formulas in U. The study of such techniques is a part of future work.

References

[Alferes *et al.*, 2000] J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinska, and T. C. Przymusinski. Dynamic Updates of Non-Monotonic Knowledge Bases. *Journal of Logic Programming*, 45(1–3):43–70, 2000.

[Eiter *et al.*, 2002] T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. On properties of update sequences based on causal rejection. *TPLP*, 2(6):721–777, 2002.

[Gelfond and Lifschitz, 1998] M. Gelfond and V. Lifschitz. Action languages. *ETAI*, 3:195–210, 1998.

[Giunchiglia and Lifschitz, 1998] E. Giunchiglia and V. Lifschitz. An action language based on causal explanation: Preliminary report. In *Proc. AAAI*, pp. 623–630, 1998.

[Katsuno and Mendelzon, 1991] H. Katsuno and A. O. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proc. KR*, pp. 387–394, 1991.

[Li and Pereira, 1996] R. Li and L. M. Pereira. What is believed is what is explained. In *Proc. AAAI*, pp. 550–555.

[Liberatore, 2000] P. Liberatore. A framework for belief update. In *Proc. JELIA*, pp. 361–375, 2000.

[Pereira *et al.*, 1991] L. M. Pereira, J. J. Alferes, and J. N. Aparicio. Contradiction removal within well-founded semantics. In *Proc. LPNMR*, pp. 105–119, 1991.

[Sakama and Inoue, 2003] C. Sakama and K. Inoue. An abductive framework for computing knowledge base updates. *TPLP*, 3(6):671–713, 2003.

[Winslett, 1990] M. Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.

[Zhang *et al.*, 2002] D. Zhang, S. Chopra, and N. Foo. Consistency of action descriptions. In *Proc. PRICAI*, 2002.