INSTITUT FÜR INFORMATIONSSYSTEME
ARBEITSGRUPPE WISSENSBASIERTE SYSTEME

UPDATING ACTION DOMAIN DESCRIPTIONS

Thomas Eiter   Esra Erdem   Michael Fink   Ján Senko

INFSYS RESEARCH REPORT 1843-13-08
NOVEMBER 2008

Institut für Informationssysteme
AG Wissensbasierte Systeme
Technische Universität Wien
Favoritenstraße 9-11
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at

INFSYS RESEARCH REPORT

INFSYS RESEARCH REPORT 1843-13-08, NOVEMBER 2008

COMPARING ACTION DESCRIPTIONS BASED ON SEMANTIC PREFERENCES

(PRELIMINARY VERSION, JULY 1, 2006)

Thomas Eiter[1] and Esra Erdem[2] and Michael Fink[1] and Ján Senko[1]

Abstract. Incorporating new information into a knowledge base is an important problem which has been widely considered. In this paper, we study the problem in a formal framework for reasoning about action and change, in which action domains are described in an action language that has a transition-based semantics. Going beyond previous works, we consider (i) a richer action language that allows for non-deterministic, and concurrent actions, as well as the representation of indirect effects and dependencies between fluents, (ii) more general updates than elementary statements, and, most importantly, (iii) meta-level knowledge, such as observations, assertions, or general domain properties that remain invariant under change, expressed in an action query language. For this setting, we formalize a notion of update of an action domain description, relative to a generic preference relation on action domain descriptions that selects most preferred solutions. We study semantic and computational aspects of this notion, where we establish basic properties of updates and a decomposition result that gives rise to a divide and conquer approach to computing solutions under certain conditions. Furthermore, we study the computational complexity of decision problems around computing solutions, both for the generic setting and for two particular preference relations, viz. set-inclusion and weight-based preference. While deciding the existence of solutions and recognizing solutions are PSPACE-complete problems in general, the problems fall back into the polynomial hierarchy under restrictions on the meta-level conditions. We finally discuss methods to compute solutions and approximate solutions (which disregard preference). Our results provide a semantic and computational basis for developing systems that incorporate new information into action descriptions in an action language, in the presence of additional conditions at the meta-level.

[1]Institute of Information Systems, Knowledge-Based Systems Group, TU Vienna, Favoritenstraße 9-11, A-1040 Vienna, Austria. Email: (eiter | michael | jan)@kr.tuwien.ac.at

[2]Faculty of Engineering and Natural Sciences, Sabancı University, Orhanli, Tuzla Istanbul 34956, Turkey. Email: esraerdem@sabanciuniv.edu

Acknowledgements: This work was 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.

This paper is a revised and significantly extended version of a preliminary paper that appeared in: *Proc. 19th International Joint Conference on Artificial Intelligence (IJCAI 2005),* pp. 418–423.

Contents

1 Introduction

As we live in a world where knowledge and information is in flux, updating knowledge bases is an important issue that has been widely studied in the area of knowledge representation and reasoning, (see e.g. [61, 11, 18, 56] and references therein). However, the problem is far from trivial and many different methods have been proposed to incorporate new information, be it affirmative or prohibitive, which are based on different formal and philosophical underpinnings, cf. [61, 36, 52]. It appears that there is no general purpose method that would work well in all settings, which is partly due to the fact that an update method is also dependent to some extent on the application domain.

In particular, in reasoning about actions and change, the dynamicity of the world is a part of the domain theory, and requires special attention in update methods. For various approaches to formal action theories, including the prominent situation calculus, event calculus, and action languages that emerged from the research on non-monotonic reasoning, the problem of change has been widely studied and different methods have been proposed (see [58] for background and references, and Section 7.1 for a more detailed discussion).

To give a simple example, consider an agent having the following knowledge, K_{TV}, about a TV with remote control:

(TV1) If the power is off, pushing the power button on the TV turns the power on.

(TV2) If the power is on, pushing the power button on the TV turns the power off.

(TV3) The TV is on whenever the power is on.

(TV4) The TV is off whenever the power is off.

Now assume that the agent 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, K_{RC}, about remote controls:

(RC1) If the power is on and the TV is off, pushing the power button on the remote control turns the TV on.

(RC2) If the TV is on, pushing the power button on the remote control turns the TV off.

The task is now to incorporate this new knowledge into the current knowledge base K_{TV}. In this particular case, this seems unproblematic, as upon simply adding K_{RC} to K_{TV} the resulting stock of knowledge is consistent; in general, however, it might be inconsistent, and a major issue is how to overcome this inconsistency.

In order to formally study this problem, we describe domains of actions, like the TV domain above, in a fragment of the action language $\mathcal{C}$ [27], by "causal laws." For instance, the direct effect of the action of pushing the power button on the TV, stated in (TV1) is described by the causal law

$$\textbf{caused } PowerON \textbf{ after } PushPB_{TV} \wedge \neg PowerON, \tag{1}$$

which expresses that this action, represented by $PushPB_{TV}$, causes the value of the fluent $PowerON$ to change from false to true; the indirect effect of this action that is stated in (TV3) is described by the causal law

$$\textbf{caused } TvON \textbf{ if } PowerON, \tag{2}$$

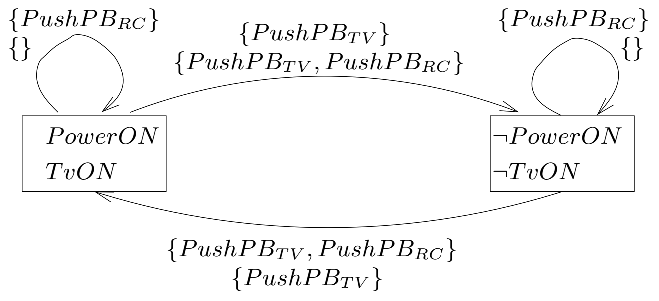

Figure 1: A transition diagram

which expresses that if the fluent $PowerON$ is caused to be true, then the fluent $TvON$ is caused to be true as well. Then, if we add the causal laws for K_{RC} to those for K_{TV}, the resulting action description is consistent according to the semantics of action language $\mathcal{C}$. (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; Figure 1 below shows an example.)

As far as action languages, as in [26], are concerned, the update problem was studied to a remarkably little extent. For the basic action language $\mathcal{A}$ (see [26]), which is far less expressive than $\mathcal{C}$, the update problem has been considered, e.g., in [41, 44]. Both works focused on updates that consist of elementary statements (i.e., essentially facts) over time, and presented specific update methods, focusing on the contents of the knowledge base.

In the present paper, we address the update problem from a more general perspective. For instance, action domains are described in a much richer language, and updates are represented in terms of a set of arbitrary causal laws. Furthermore, meta-level knowledge about the particular domain, such as assertions obtained from observations (as in [41, 44], simply facts, but possibly also of more complex form such as transitions), or general properties of the domain (integrity constraints) that might not be expressible in the formal action language per se, could be taken into account for updating the given action domain description.

For example, for the effective use of the TV system in the above scenario, the following condition is desired to hold:

(C) Pushing the power button on the remote control is always possible.

If K_{RC} is simply added to K_{TV}, 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 (RC2) and (TV3) contradict. The question is then how the agent can update K_{TV} by incorporating K_{RC} relative to (C). Furthermore, as condition (C) is not expressible in the action language $\mathcal{C}$, the question is also how to formally represent (C) and similar meaningful conditions.

Motivated by these questions, we consider a generic framework for incorporating new causal laws into an existing action description, that takes meta-level knowledge into account. Our main contributions can be summarized as follows:

(1) We introduce a formal notion of an *action update problem*, which is, given action descriptions D and I, and a set C of conditions, to determine a (possibly new) action description D' which incorporates I into D. While D and I are in (a canonical subset of) $\mathcal{C}$, we describe conditions like (C) by "queries" in an *action query language*, similar to the one in [26]. For instance, the condition (C) above can be described in this

language by the query

$$\textbf{ALWAYS executable } \{PushPB_{RC}\}. \quad (3)$$

In a more fine-grained treatment, D is split into an unmodifiable part, D_u, and a modifiable part, D_m, while C is split into obligatory conditions, C_o, (which must hold under all circumstances) and preference conditions, C_p, (which ideally should hold, but might be violated).

A solution to an action update problem is then defined in terms of an action description D' that consists of I and statements from D such that C_o is satisfied; as, in general, different candidates D' are possible, we use a (strict) *preference relation* $\sqsubset_C$ over action descriptions[1] in order to discriminate amongst alternatives and to single out a most preferable candidate D' as the result. Here the subscript $_C$ indicates that the preference relation is possibly dependent on the set C of conditions. Such a preference relation can be defined in different ways, in terms of syntactic conditions (e.g., the set of causal laws in an action description), or semantic conditions (e.g., number of conditions fulfilled by an action description).

(2) We investigate semantic properties of action updates, and establish some basic properties regarding solution preference, and special forms of updates, which serve as tests for the suitability of the notions proposed. We furthermore determine conditions under which computing a solution to an action update problem can be structurally decomposed, such that a divide-and-conquer approach becomes feasible. In particular, this is possible if the action description and the conditions can be split into disjoint parts that interfere in a benign way, and if the preference ordering can be gracefully decomposed along this split.

(3) We study the computational complexity of the action update problem, where we consider the generic setting (making some assumptions about the cost of deciding whether the conditions C are satisfied by an action description D, denoted $D \models C$, and whether $D \sqsubset_C D'$ holds given D and D'), as well as some natural instances. Among the latter are those where the preference relation $\sqsubset_C$ is ordinary set-inclusion and where it is weight-based relative to satisfied conditions. Under the assumption that testing $D \models C$ and $D \sqsubset_C D'$ is feasible in polynomial space, deciding the existence of some solution to an action update problem turns out to be PSPACE-complete in general, and also verifying a given solution candidate has this complexity. However, the complexity of both problems falls back into the polynomial hierarchy, if deciding $D \models C$ and $D \sqsubset_C D'$ is located there, and is located at most one level higher up there; we recall here that deciding the consistency of an action description in $\mathcal{C}$ is intractable in general (and NP-complete for the canonical fragment of our concern). Given that the test $D \models C$ and $D \sqsubset_C D'$ is polynomial, deciding solution existence is NP-complete and thus not harder than the consistency problem, and recognizing a given solution is only mildly harder.

(4) Finally, we discuss methods for computing solutions and "near-solutions" which approximate them, by disregarding solution preference. As for solutions, we focus on set-inclusion and one particular weight-based comparison, as preference relations $\sqsubset_C$, which use an oracle for near-solutions. For near-solutions, we present a method that reduces the problem into reasoning over an action description that is constructed from the problem input; here, answering action queries can be exploited to test given candidates.

Our results go significantly beyond previous results in the literature (see Section 7.1), and provide a semantic and computational basis for developing systems that incorporate new information into action descriptions in an action language, in the presence of further conditions that can be formally expressed at

[1]That is, $\sqsubset_C$ is irreflexive and transitive.

the meta-level, and thus cleanly be separated from the action description at the object level. Our generic framework can be instantiated to different settings, which reflect different intuitions or criteria for solution preference. It thus provides a flexible tool for modeling action update. As a byproduct of our research, we also obtain decomposition results of action descriptions that emerge from special cases of action update instances, which are interesting in their own right.

The rest of this paper is structured as follows. In the next section, we provide preliminaries about transition diagrams, action languages, and actions queries as needed for the problem setting. After that, we define in Section 3 the update problem in a generic framework and briefly introduce a syntactic and a semantic instance of it. In Section 4, we study some semantic properties of updates, including possible decompositions. After that, we turn to computational issues. In Section 5, we characterize the computational complexity of problems around updates, and in Section 6 we provide algorithms for computing updates. After a discussion of related work and further aspects of the problem in Section 7, we conclude with a summary and issues for further research.

2 Preliminaries

2.1 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.

2.2 Action Description Languages

We consider the prime subset of the action description language $\mathcal{C}$ [27] that consists of two kinds of expressions (called *causal laws*): *static laws* of the form

$$\textbf{caused } L \textbf{ if } G, \tag{4}$$

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

$$\textbf{caused } L \textbf{ if } G \textbf{ after } H, \tag{5}$$

where L and G are as above, and H is a propositional combination of fluent names and action names. In (4) and (5) 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. An expression of the form

$$\textbf{inertial } L_1, \ldots, L_k$$

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

Figure 2: An action description for K_{TV}

$$\begin{array}{l}
\textbf{caused } \mathit{TvON} \textbf{ after } \mathit{PushB}_{RC} \wedge \mathit{PowerON} \wedge \neg \mathit{TvON} \\
\textbf{caused } \neg \mathit{TvON} \textbf{ after } \mathit{PushB}_{RC} \wedge \mathit{TvON}
\end{array}$$

Figure 3: Causal laws for K_{RC}

stands for the causal laws

$$\textbf{caused } L_i \textbf{ if } L_i \textbf{ after } L_i \qquad (1 \leq i \leq k)$$

describing that the value of the fluent L_i stays the same unless changed by an action.

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, denoted by $T(D)$, is defined as follows:

(i) S is the set of all interpretations s of $\mathbf{F}$ such that, for every static law (4) 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 (4) in D for which s' satisfies G, and
- dynamic laws (5) in D for which s' satisfies G and $s \cup A$ satisfies H.

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 for K_{TV} in 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 as $D_u \cup D_m$.

2.3 Action Queries

To talk about observations of the world, or assertions about the effects of the execution of actions, we use an action query language like in [26], consisting of queries described as follows [16].

A *basic query* is either *(a)* a *static query* of the form

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

where F is a fluent formula, or *(b)* a *dynamic query* of the form

$$\textbf{necessarily } Q \textbf{ after } A_1; \ldots; A_n, \tag{7}$$

where Q is a basic query and each A_i is an action; or *(c)* any propositional combination of basic queries. An *existential query* is an expression of the form

$$\textbf{SOMETIMES } Q, \tag{8}$$

where Q is a basic query; a *universal query* is of the form

$$\textbf{ALWAYS } Q, \tag{9}$$

where Q is a basic query. A *query* q is a propositional combination of existential queries and universal queries.

As for the semantics, let $T = \langle S, V, R\rangle$ be a transition diagram, with a set S of states, a value function V mapping, at each state s, every fluent P to a truth value, and a set R of transitions. A *history* of T of length n is a sequence

$$s_0, A_1, s_1, \ldots, s_{n-1}, A_n, s_n \tag{10}$$

where each $\langle s_i, A_{i+1}, s_{i+1}\rangle$ $(0 \leq i < n)$ is in R. We say that a state $s \in S$ *satisfies* a basic query Q' of form (6) (resp. (7)) relative to T (denoted $T, s \models Q'$), if the interpretation $P \mapsto V(P, s)$ satisfies F (resp. if, for every history $s = s_0, A_1, s_1, \ldots, s_{n-1}, A_n, s_n$ of T of length n, basic query Q is satisfied at state s_n). For other forms of basic queries Q, *satisfaction* is defined by the truth tables of propositional logic. If T is described by an action description D, then the satisfaction relation between s and a basic query Q can be denoted by $D, s \models Q$ as well.

Note that, for every state s and for every fluent formula F,

$$D, s \models \textbf{holds } F \iff D, s \models \neg\textbf{holds } \neg F.$$

For every state s, every fluent formula F, and every action sequence $A_1, \ldots, A_n$ $(n \geq 1)$, if

$$D, s \models \textbf{necessarily } (\textbf{holds } F) \textbf{ after } A_1; \ldots; A_n$$

then

$$D, s \models \neg\textbf{necessarily } (\neg\textbf{holds } F) \textbf{ after } A_1; \ldots; A_n.$$

We say that D *entails* a query q (denoted $D \models q$) if one of the following holds:

- q is an existential query (8) and $D, s \models Q$ for some state $s \in S$;
- q is a universal query (9) and $D, s \models Q$ for every state $s \in S$;
- $q = \neg q'$ and $D \not\models q'$;
- $q = q_1 \wedge q_2$ and $D \models q_1$ and $D \models q_2$; or
- $q = q_1 \vee q_2$ and $D \models q_1$ or $D \models q_2$.

For every basic query Q,

$$D \models \textbf{SOMETIMES } Q \text{ iff } D \models \neg\textbf{ALWAYS } \neg Q.$$

For a set C of queries, we say that D *entails* C (denoted $D{\models}C$) if D *entails* every query in C. Consider, e.g., the action description presented in Figure 2. It does not entail any set of queries containing

$$\textbf{ALWAYS necessarily } (\textbf{holds } \neg\, TvON) \textbf{ after } \{PushPB_{RC}\}$$

because this query is not satisfied at the state $\{TvON, PowerON\}$; but, it entails the queries:

$$\textbf{ALWAYS} \quad \textbf{holds } PowerON \equiv TvON, \tag{11}$$

$$\begin{aligned}\textbf{ALWAYS} \quad & \textbf{holds } PowerON \wedge TvON \supset \\ & \neg\textbf{necessarily } (\textbf{holds } TvON) \textbf{ after } \{PushPB_{TV}\}.\end{aligned} \tag{12}$$

In the rest of the paper, an expression of the form

$$\textbf{possibly } Q \textbf{ after } A_1; \ldots ; A_n,$$

where Q is a basic query and each A_i is an action, stands for the dynamic query

$$\neg\textbf{necessarily } \neg Q \textbf{ after } A_1; \ldots ; A_n;$$

an expression of the form

$$\textbf{evolves } F_0; A_1; F_1; \ldots ; F_{n-1}; A_n; F_n, \tag{13}$$

where each F_i is a fluent formula, and each A_i is an action, stands for

$$\textbf{holds } F_0 \ \wedge \ \textbf{possibly } (\textbf{holds } F_1 \wedge \textbf{possibly } (\textbf{holds } F_2 \wedge \ldots) \textbf{ after } A_2) \textbf{ after } A_1;$$

and

$$\textbf{executable } A_1; \ldots ; A_n$$

where each A_i is an action, stands for

$$\textbf{possibly } \mathit{True} \textbf{ after } A_1; \ldots ; A_n.$$

We sometimes drop **holds** from static queries appearing in dynamic queries.

2.3.1 Examples

To get a better intuition about the capability of the action query language, we give some examples of properties that can be expressed in it.

- *Existence of certain states, transitions, and histories:* For instance, we can express the existence of states where a formula F holds by means of the query

$$\textbf{SOMETIMES holds } F.$$

Similarly, we can express the existence of a transition from some state where a formula F holds to another state where a formula F' holds, by the execution of an action A:

$$\textbf{SOMETIMES holds } F \wedge \textbf{possibly } F' \textbf{ after } A.$$

In general, the existence of a history (10) such that, for each s_i of the history, the interpretation $P \mapsto V(P, s_i)$ satisfies some formula F_i is expressed by the query:

$$\textbf{SOMETIMES evolves } F_0; A_1; F_1; \ldots; F_{n-1}; A_n; F_n. \tag{14}$$

For instance, the query

$$\begin{array}{ll}\textbf{SOMETIMES evolves} & \mathit{PowerON}; \{\mathit{PushPB}_{TV}\}; \\ & \neg\mathit{PowerON}; \{\mathit{PushPB}_{TV}\}; \mathit{PowerON}.\end{array} \tag{15}$$

describes the presence of the following history in Fig. 1:

$$\begin{array}{l}\{\mathit{PowerON}, \mathit{TvON}\}, \{\mathit{PushPB}_{TV}\}, \\ \{\neg\mathit{PowerON}, \neg\mathit{TvON}\}, \{\mathit{PushPB}_{TV}\}, \{\mathit{PowerON}, \mathit{TvON}\}.\end{array} \tag{16}$$

- *(Non-)executability of an action:* Like in [14], executability of an action sequence $A_1, \ldots, A_n$ ($n \geq 1$) at every state can be described by

  $$\textbf{ALWAYS executable } A_1; \ldots; A_n.$$

  That no action is possible at a state where formula F holds is expressed by

  $$\textbf{SOMETIMES holds } F \wedge \bigwedge_{A \in 2^{\mathbf{A}}} \textbf{necessarily } \mathit{False} \textbf{ after } A.$$

- *Mandatory and possible effects of actions:* Like in [14], mandatory effects of a sequence $A_1, \ldots, A_n$ ($n \geq 1$) of actions in a given context are described by

  $$\textbf{ALWAYS holds } G \supset \textbf{necessarily } F \textbf{ after } A_1; \ldots; A_n;$$

  and possible effects of a sequence of actions in a context by

  $$\textbf{ALWAYS holds } G \supset \textbf{possibly } F \textbf{ after } A_1; \ldots; A_n.$$

  In these queries, F describes the effects and G the context.

3 Problem Description

In this section, we provide a formal description of the update problem, and its solution, as well as a weaker form of solution, called near-solution.

Informally, we define an *Action Description Update (ADU)* problem by an action description $D = D_u \cup D_m$, a set I of causal laws, a set $C = C_o \cup C_p$ of queries, and a preference relation $\sqsubset_C$ over action descriptions. Here D_u and D_m are the unmodifiable (protected) and the modifiable part of D, respectively, and I is the update that has to be incorporated. The queries in C_o are "hard (obligatory) constraints" that have to be satisfied in an acceptable action description, while the queries in C_p are "soft (preference) constraints" that might be accounted for by the preference relation $\sqsubset_C$. In the latter, $D \sqsubset_C D'$ expresses that D is less preferable compared to D'.

Definition 1 (Action Description Update) *Given an action description $D = D_u \cup D_m$, a set I of causal laws, a set $C = C_o \cup C_p$ of queries, and a preference relation $\sqsubset_C$ over action descriptions, all over the same signature $\mathcal{L}$, an action description D'* accomplishes an (action description) update *of D by I relative to C, if*

(i) D' *is consistent,*

(ii) $D_u \cup I \subseteq D' \subseteq D \cup I$,

(iii) $D' \models C_o$,

(iv) there is no consistent action description D'' such that $D_u \cup I \subseteq D'' \subseteq D \cup I$, $D'' \models C_o$, and $D' \sqsubset_C D''$.

Such a D' is called a solution *to the ADU problem $(D, I, C, \sqsubset_C)$. If an action description D' satisfies (i)–(iii), then we call D' a* near-solution *to the ADU problem $(D, I, C, \sqsubset_C)$.*

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 the causal laws in the variable part are considered to be either "correct" or "wrong", and in the latter case simply disposed.

Condition *(iv)* 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 (as will be discussed below).

Finally, Condition *(iv)* picks the most preferred action description among the ones for which Conditions *(i)–(iii)* are satisfied.

In an ADU problem, the preference relation can be described in various ways. For instance, it can be defined in terms of syntactic conditions, like simple set inclusion. If we define $\sqsubset_C$ to be $\subset$, then an action description D is less preferable than an action description D' if $D \subset D'$. Alternatively, the preference relation $\sqsubset_C$ can be defined in terms of semantic conditions. For instance, once a weight is assigned to each action description with respect to some semantic measure (e.g., the number of queries in C_p entailed by the description) by a function $weight$, we can take $\sqsubset_C$ to be an operator $<_{\mathit{weight}}$ comparing the weights of the action descriptions; then an action description D is less preferable than an action description D' if $D <_{\mathit{weight}} D'$.

In the literature, two kinds of changes that incorporate new information into a knowledge base have been identified, viz. revision (which adds more precise knowledge about the domain) and update (which is a change of the world per se) [60], which should be governed by different sets of postulates in axiomatic approaches like the AGM theory [1] and the KM theory [36]. Our notion of ADU has more of a revision flavor, but we do not govern it with AGM or KM postulates, as the formalism is non-monotonic; see Section 7.2 for more discussion. However, the conditions C can be adjusted if the nature of the change I is known. In case of a revision, C should reasonably contain all conditions corresponding to observations made about the domain, while other conditions may be kept or dropped; on the other hand, if I is a change of the world per se, then conditions corresponding to observations might be dropped.

3.1 Examples

The following is an example of an ADU problem with the syntax-based preference relation above.

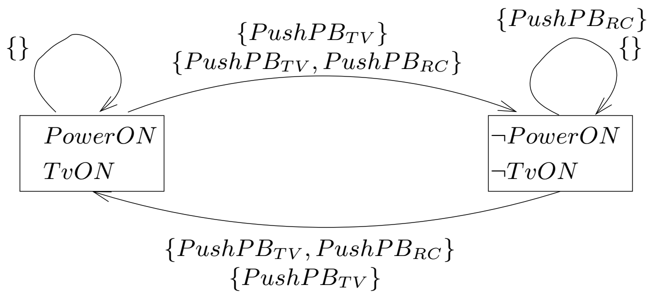

Figure 4: Transition diagram described by $D \cup I$ of Ex. 1.

Example 1 Let D be the action description for K_{TV} in Figure 2, with D_m containing the third and the fourth causal laws (i.e., $D_m = \{$**caused** $TvON$ **if** $PowerON$, **caused** $\neg TvON$ **if** $\neg PowerON\}$, and let I be the set of the causal laws for K_{RC} in Figure 3. Let $C = C_o$ contain the besides the query (3) also the queries (12) and

$$\textbf{ALWAYS executable } \{PushPB_{TV}\}, \tag{17}$$

and take (strict) set-inclusion ($\subset$) as the preference relation $\sqsubset_C$. The transition diagram described by $D \cup I$ is shown in Figure 4. 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, $D \cup I$ is not a solution to the ADU problem $(D, I, C, \sqsubset_C)$. In fact, a solution is obtained by dropping the static law (2) from $D \cup I$. $\Box$

For an instance of a semantic definition of $\sqsubset_C$, consider the following setting based on weights that are assigned to queries on C (i.e., *weighted queries* in [16]). We define the weight of an action description D relative to a set C of queries, and a weight function $f : C \rightarrow \mathbb{R}$ mapping each query in C to a real number by

$$weight_q(D) = \sum\nolimits_{c \in C, D \models c} f(c).$$

Intuitively, the weight of an action description defined relative to the weights of queries encodes to what extent the set C of given preferable queries is satisfied. (Note that f can easily express a threshold function as well.) With this definition, the more the highly preferred queries are satisfied, the more preferred the action description is.

Example 2 Reconsider our previous example where C_p consists of the query (12) with weight 1. Suppose that the preference relation $\sqsubset_C$ is defined in terms of a weight function on queries (i.e., $\sqsubset_C = <_{weight_q}$). Then, the action descriptions $D' = (D \cup I) \setminus \{$**caused** $TvON$ **if** $PowerON\}$ and $D'' = D_u \cup I$ satisfy C_o and thus are near-solutions. However, D'' does not satisfy C_p, which implies $weight_q(D'') = 0$, whereas $weight_q(D') = 1$, and hence $D'' \sqsubset_C D'$.

For further details on comparing action description by means of weighted queries and other semantic preferences, we refer the reader to [16].

In the rest of the paper, we will study ADU problems at an abstract level, leaving the preference relation undefined. For some problems, we will provide more concrete results by instantiating the preference relation: we will take $\sqsubset_C$ as $\subset$ (and $C_p = \emptyset$, thus $C = C_o$) for an instance of a syntax-based relation, and we consider $\sqsubset_C$ = $<_{weight_q}$ as a representative of the semantic-based approaches.

4 Properties of Updates

In this section, we study some basic properties of solutions to an ADU problem. To this end, we first introduce a subsumption relation between action descriptions, and then show that solutions to an ADU problem fulfill some desired properties regarding special updates, provided that the preference relation $\sqsubseteq_C$ obeys some natural conditions. We then consider the structure of solutions and near-solutions, and establish a disjoint factorization result that allows for decomposing an ADU into smaller parts.

4.1 Basic Update Properties

We define subsumption of causal laws by an action description as follows.

Definition 2 (Subsumption) *Let D be an action description over a signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$. Then,*

- *a static law (4) over $\mathcal{L}$ is* subsumed *by D, if for every state s in $T(D)$, the interpretation of $\mathbf{F}$ describing s satisfies $G \supset L$;*
- *a dynamic law (5) over $\mathcal{L}$ is* subsumed *by D, if for every transition $\langle s, A, s' \rangle$ in $T(D)$, the following holds: if the interpretation of $\mathbf{F} \cup \mathbf{A}$ describing s and A satisfies H, then the interpretation of $\mathbf{F}$ describing s' satisfies $G \supset L$.*

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

Furthermore, we build on the properties of a preference relation $\sqsubset_C$ introduced next.

In the following, for an action description D and a set C of queries, let us denote by C_D the set $\{c \in C \mid D \models c\}$.

Definition 3 *Given a set of queries C over a signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$, a preference relation $\sqsubset_C$ over a $\mathcal{L}$ is called*

- monotone with respect to C*, if for any two action descriptions D and D' in $\mathcal{L}$, $C_{D'} \subseteq C_D$ implies $D \not\sqsubset_C D'$, and* strongly monotone with respect to C*, if additionally $C_{D'} \subset C_D$ implies $D' \sqsubset_C D$;*
- monotone with respect to $\mathcal{L}$*, if for any two action descriptions D and D' in $\mathcal{L}$, $D' \subseteq D$ implies $D \not\sqsubset_C D'$, and* strongly monotone with respect to $\mathcal{L}$*, if additionally $D' \subset D$ implies $D' \sqsubset_C D$;*
- non-minimizing with respect to $\mathcal{L}$*, if for any action description D in $\mathcal{L}$, $D \models C$ implies $D \not\sqsubset_C D'$ for all $D' \subseteq D$, and* strongly non-minimizing with respect to $\mathcal{L}$*, if additionally $D \models C$ implies $D' \sqsubset_C D$ for all $D' \subset D$.*

We say that $\sqsubset_C$ is *monotone*, if it is either monotone with respect to C or monotone with respect to $\mathcal{L}$ (or both).

Monotonicity is an intuitive potential requirement one might have on a preference relation: monotonicity with respect to C encodes the semantically motivated preference of satisfying preferable queries as much as possible, whereas monotonicity with respect to $\mathcal{L}$ expresses a more syntactic view of retaining as many causal laws as possible. This is reflected in our representative preference relations. Notice that $\subset$ is strongly monotone with respect to $\mathcal{L}$ (but not necessarily with respect to C), whereas $<_{weight_q}$ is monotone with respect to C if, for instance, all weights are nonnegative (but not necessarily with respect to $\mathcal{L}$).

Obviously, any monotone preference relation is also non-minimizing with respect to $\mathcal{L}$, and strong monotonicity with respect to $\mathcal{L}$ implies that $\sqsubset_C$ is also strongly non-minimizing with respect to $\mathcal{L}$. Intuitively, a non-minimizing preference relation with respect to $\mathcal{L}$ ensures that syntactically smaller (with respect to subset inclusion) action descriptions cannot prevent an action description that satisfies all queries from being a solution, while the respective strong property explicitly excludes syntactically smaller action descriptions as solutions in this case (note that the additional condition implies the condition of non-minimizing, and could serve as a definition alone). This intuition motivates basic properties of solutions to an ADU problem as follows.

Proposition 1 (Subsumption) *Let $(D, I, C, \sqsubset_C)$ be an ADU problem, such that $\sqsubset_C$ is non-minimizing with respect to $\mathcal{L}$, D is consistent and $D \models C$. If D subsumes I, then $D \cup I$ is a solution to $(D, I, C, \sqsubset_C)$. Moreover if $\sqsubset_C$ is strongly non-minimizing with respect to $\mathcal{L}$, then $D \cup I$ is the unique solution.*

Proof. Let $D = D_u \cup D_m$ and let $T(D) = \langle S, V, R\rangle$. Since $D \cup I = D_u \cup I \cup D_m$ trivially satisfies *(ii)* of our definition of update accomplishment, it remains to show: *(i)* $D \cup I$ is consistent, *(iii)* $D \cup I \models C_o$, and *(iv)* $D_u \cup I \subseteq D' \subseteq D \cup I$ and $D' \models C_o$ implies $D \cup I \not\sqsubset_C D'$.

Let $T(D \cup I) = \langle S', V', R'\rangle$. In the following we prove that $T(D \cup I) = T(D)$.

$S' = S$: Since $D \subseteq D \cup I$, we get $S' \subseteq S$. Furthermore, D subsumes I and, hence, every $s \in S$ satisfies $G \supset L$ for all static laws of form (4) in I, i.e., $S \subseteq S'$.

$V' = V$: Follows from $S' = S$ and our labeling convention for states.

$R' = R$: Let $\langle s, A, s'\rangle$ be a *candidate* for a transition relation, R, of an action description, D, if *(a)* s' satisfies the heads L of all static laws of form (4) in D, for which s' satisfies G, and *(b)* s' satisfies the heads L of all dynamic laws of form (5) in D, for which s' satisfies G and $s \cup A$ satisfies H. Furthermore, let s' be a *determined successor* of s w.r.t. A, if the set of heads of all laws applicable to $\langle s, A, s'\rangle$ uniquely determines s', i.e., it contains (at least) one fluent literal for every fluent in **F**. Then, $\langle s, A, s'\rangle \in R$ iff it is a candidate for R and s' is a determined successor of s with respect to A. Since $D \subseteq D \cup I$, every candidate $\langle s, A, s'\rangle$ for R' is a candidate for R. Moreover, that D subsumes I implies that every candidate $\langle s, A, s'\rangle$ for R is a candidate for R' as well. As $\langle s, A, s'\rangle$ is neither in R nor in R', if s' is not a determined successor of s with respect to A it follows that $R' = R$.

Given that D is consistent and that $D \models C$, $T(D \cup I) = T(D)$ proves *(i)* and *(iii)*. As for *(iv)*, $D \models C$ and $T' = T$ implies $D \cup I \models C$. Since $\sqsubset_C$ is non-minimizing with respect to $\mathcal{L}$, it follows for all $D_u \cup I \subseteq D' \subseteq D \cup I$, that $D \cup I \not\sqsubset_C D'$, which proves *(iv)*. Therefore, $D \cup I$ is a solution to $(D, I, C, \sqsubset_C)$. Moreover, if $\sqsubset_C$ is strongly non-minimizing with respect to $\mathcal{L}$, then $D' \sqsubset_C D \cup I$ holds for all $D_u \cup I \subseteq D' \subseteq D \cup I$. This implies that $D \cup I$ is the unique solution to $(D, I, C, \sqsubset_C)$ in this case. $\Box$

From this result, we obtain the following corollaries telling us that the solution to an ADU is as we would expect in some extremal cases, that correspond to cases that were considered for nonmonotonic logic programming updates [4, 18].

Corollary 1 (Void Update) *Let $(D, \emptyset, C, \sqsubset_C)$ be an ADU problem. If $\sqsubset_C$ is non-minimizing with respect to $\mathcal{L}$, D is consistent, and $D \models C$, then D is a solution to $(D, \emptyset, C, \sqsubset_C)$. If $\sqsubset_C$ is strongly non-minimizing with respect to $\mathcal{L}$, then D is the unique solution.*

Corollary 2 (Idempotence) *Let $(D, D, C, \sqsubset_C)$ be an ADU problem, such that $\sqsubset_C$ is non-minimizing with respect to $\mathcal{L}$, D is consistent, and $D \models C$, then D is the unique solution to $(D, D, C, \sqsubset_C)$.*

Let us call a causal law *tautological*, if it is subsumed by every action description D. Informally, such a causal law has no logical content, and updating with it should not lead to any change. In fact we have the following property.

Corollary 3 (Addition of Tautologies) *Let $(D, I, C, \sqsubset_C)$ be an ADU problem, such that $\sqsubset_C$ is non-minimizing with respect to $\mathcal{L}$, D is consistent, and $D \models C$. If I consists of tautological causal laws, then $D \cup I$ is a solution to $(D, I, C, \sqsubset_C)$. If $\sqsubset_C$ is strongly non-minimizing with respect to $\mathcal{L}$, then $D \cup I$ is the unique solution.*

Notice that a similar property fails for logic programming updates as in [4, 18].

Example 3 Consider an action description D that has the following causal laws:

$$\textbf{inertial } \mathit{LightON}, \neg \mathit{LightON}, \tag{18}$$
$$\textbf{caused } \mathit{LightON} \textbf{ after } \mathit{SwitchLight} \wedge \neg \mathit{LightON}, \tag{19}$$
$$\textbf{caused } \neg \mathit{LightON} \textbf{ after } \mathit{SwitchLight} \wedge \mathit{LightON}. \tag{20}$$

Since D is consistent and $\subset$ is strongly non-minimizing, we can state for any set C of queries, such that $D \models C$: D is the unique solution to $(D, \emptyset, C, \subset)$ (void update), as well as to $(D, D, C, \subset)$ (idempotence), and to $(D, D', C, \subset)$ for any tautological action description D' (addition of tautologies).

Considering $<_{\mathit{weight}_q}$ with nonnegative weights for any query $c \in C$ instead of $\subset$ as a preference relation (which is non-minimizing), we can still infer that D' is a solution, in general however, it need not be unique. $\Box$

4.2 Disjoint Factorization

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.

Definition 4 (NOP) *We say that an action description D* has NOP, *if $T(D)$ has either (i) a transition $\langle s, \emptyset, s\rangle$ for some state s, or (ii) for every state s some transition $\langle s, \emptyset, s'\rangle$.*

Notice that NOP is a very natural property that often applies, in particular for *time-driven* domains, where passage of time causes $\langle s, \emptyset, s\rangle$ by inertia, usually for all states s.

The following lemma is the key for our disjoint factorization result. For any action signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A}\rangle$, we denote by $\mathcal{L}_D$ the part of it which appears in any action description D.

Lemma 1 *Let $T(D^i) = \langle S^i, V^i, R^i\rangle$ for action descriptions D^i, $i = 0, 1$, such that $\mathcal{L}_{D^0} \cap \mathcal{L}_{D^1} = \emptyset$. Let $T(D^0 \cup D^1) = \langle S, V, R\rangle$. Then the following hold:*

(i) $S = S^0 \times S^1$;

(ii) *If $R^0 \neq \emptyset$ and $R^1 \neq \emptyset$ then, for $\langle s^0_0, A^0, s^0_1\rangle \in R^0$ and $\langle s^1_0, A^1, s^1_1\rangle \in R^1$, $\langle s^0_0 \cup s^1_0, A^0 \cup A^1, s^0_1 \cup s^1_1\rangle \in R$;*

(iii) *for $\langle s, A, s'\rangle \in R$, $\langle s \cap \mathcal{L}_{D^0}, A \cap \mathcal{L}_{D^0}, s' \cap \mathcal{L}_{D^0}\rangle \in R^0$ and $\langle s \cap \mathcal{L}_{D^1}, A \cap \mathcal{L}_{D^1}, s' \cap \mathcal{L}_{D^1}\rangle \in R^1$.*

Proof. *(i)* is trivial. We prove *(ii)* and *(iii)* as follows.
(ii) Suppose that $R^0 \neq \emptyset$ and $R^1 \neq \emptyset$. Take any $\langle s_0^0, A^0, s_1^0\rangle \in R^0$ and $\langle s_0^1, A^1, s_1^1\rangle \in R^1$. We show that $\langle s_0^0 \cup s_0^1, A^0 \cup A^1, s_1^0 \cup s_1^1\rangle \in R$. Suppose this is not the case. Then one of the following two cases holds:

(1) For some dynamic law d of the form (5) in $D^0 \cup D^1$, $s_0^0 \cup s_0^1 \cup A^0 \cup A^1$ satisfies H, and $s_1^0 \cup s_1^1$ does not satisfy $G \wedge L$. W.l.o.g., suppose that d is in D^0. Then, since $\mathcal{L}_{D^0} \cap \mathcal{L}_{D^1} = \emptyset$, $s_0^0 \cup A^0$ satisfies H and s_1^0 does not satisfy $G \wedge L$. This implies that $\langle s_0^0, A^0, s_1^0\rangle \notin R^0$, which is a contradiction.

(2) $s_2^0 \cup s_2^1$ is another state (different from $s_1^0 \cup s_1^1$) that satisfies the heads of all static laws (4) in $D^0 \cup D^1$ for which $s_0^0 \cup s_0^1$ satisfies G, and of every dynamic law (5) in $D^0 \cup D^1$, such that satisfaction of H by $s_0^0 \cup s_0^1 \cup A^0 \cup A^1$ implies that $s_1^0 \cup s_1^1$ satisfies G. Then, (since each causal law is in D^0 or D^1 but not in both, due to $\mathcal{L}_{D^0} \cap \mathcal{L}_{D^1} = \emptyset$) it follows that, s_2^0 satisfies the heads of all static laws (4) in D^0 for which s_0^0 satisfies G, and of every dynamic law (5) in D^0, such that satisfaction of H by $s_0^0 \cup A^0$ implies that s_1^0 satisfies G. This implies that $\langle s_0^0, A^0, s_1^0\rangle \notin R_1$. (Symmetrically, the claim holds for D^1.) This is again a contradiction.
(iii) Take any $\langle s, A, s'\rangle \in R$. W.l.o.g., suppose that $\langle s \cap \mathcal{L}_{D^0}, A \cap \mathcal{L}_{D^0}, s' \cap \mathcal{L}_{D^0}\rangle \notin R^0$. Then one of the following two cases holds:

(1) For some dynamic law d of the form (5) in D^0, $s \cap \mathcal{L}_{D^0} \cup A \cap \mathcal{L}_{D^0}$ satisfies H, and $s' \cap \mathcal{L}_{D^0}$ does not satisfy $G \wedge L$. Since $\mathcal{L}_{D^0} \cap \mathcal{L}_{D^1} = \emptyset$, $s \cup A$ satisfies H and s' does not satisfy $G \wedge L$. This implies $\langle s, A, s'\rangle \notin R$, a contradiction.

(2) s_2^0 is another state that satisfies the heads of all static laws in D^0 for which $s \cap \mathcal{L}_{D^0}$ satisfies G, and of every dynamic law (5) in D_1 such that satisfaction of H by $s \cap \mathcal{L}_{D^0} \cup A \cap \mathcal{L}_{D^0}$ implies that $s' \cap \mathcal{L}_{D^0}$ satisfies G. Consider $s'' = s_2^0 \cup s' \cap \mathcal{L}_{D^1}$. Due to *(i)* above, $s'' \in S$. Moreover, since $\mathcal{L}_{D^0} \cap \mathcal{L}_{D^1} = \emptyset$, the following holds: s'' satisfies the heads of all static laws (4) in $D^0 \cup D^1$ for which s satisfies G, and of every dynamic law (5) in $D^0 \cup D^1$, such that satisfaction of H by $s \cup A$ implies that s' satisfies G. This implies that $\langle s, A, s'\rangle \notin R$, which is a contradiction. □

Intuitively, this lemma describes how the transition diagram of an action description can be composed, if the action description consists of two syntactically disjoint parts. It can thus be exploited to decompose a given action description into disjoint parts as in our next result. For such a decomposition to be faithful in the sense that solutions to the respective ADU subproblems can be composed to yield a solution to the original ADU problem, care has to be taken with respect to two aspects: First, an empty set of transitions shall not compromise the approach, and thus has to be avoided, in the presence of dynamic queries (cf. Lemma 1 *(ii)*). This can be guaranteed by the NOP property, which will in fact be sufficient for composing near-solutions. Second, for composing solutions the composed preference relation needs to comply with the preferences of the subproblems. Stated from the viewpoint of decomposition, the preference relation must be factorizable.

Towards a formal treatment of these ideas, we need further terminology. We call $(\mathcal{L}^0, \mathcal{L}^1)$, where $\mathcal{L}^i = \langle \mathbf{F}^i, \mathbf{A}^i\rangle$, $i = 0, 1$, a *partitioning* of a signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A}\rangle$, if $(\mathbf{F^0}, \mathbf{F^1})$ and $(\mathbf{A^0}, \mathbf{A^1})$ are partitioning of $\mathbf{F}$ and $\mathbf{A}$, respectively. We first define decompositions of action descriptions and conditions.

Definition 5 (AD/Condition Decomposition) *Suppose $(\mathcal{L}^0, \mathcal{L}^1)$ is a partitioning of a signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A}\rangle$, and let X be either an action description or a set of conditions over $\mathcal{L}$. Then a partitioning (X^0, X^1) of X is called a* decomposition of X with respect to $(\mathcal{L}^0, \mathcal{L}^1)$, *if $\mathcal{L}_{X^i} \subseteq \mathcal{L}^i$, for $i = 0, 1$. Furthermore, X is* decomposable with respect to $(\mathcal{L}^0, \mathcal{L}^1)$, *if such a decomposition exists.*

Based on this, we next define the notion of a near-decomposition of an ADU problem, which splits the action description and the conditions in separate parts while disregarding preference.

Definition 6 (Near-Decomposition) *Let $(D, I, C, \sqsubset_C)$ be an ADU problem with signature $\mathcal{L}$, and let (D^0, D^1), (I^0, I^1), and (C^0, C^1) be decompositions of D, I, and C, respectively, with respect to a partitioning $(\mathcal{L}^0, \mathcal{L}^1)$ of $\mathcal{L}$. Then, $((D^0, I^0, C^0), (D^1, I^1, C^1))$ is a* near-decomposition *of $(D, I, C, \sqsubset_C)$* with respect to $(\mathcal{L}^0, \mathcal{L}^1)$.

The following theorem now formally shows that the near-solutions of an ADU problem can be obtained from those of a near-decomposition, provided that some ramifying conditions hold. We say that a query c occurs *positively* (resp. *negatively*) in a set C of queries, if c occurs in the scope of an even (resp. odd) number of negations in a query in C.

Theorem 1 (Disjointness) *Given an ADU problem $(D, I, C, \sqsubset_C)$ with signature $\mathcal{L}$, let $((D^0, I^0, C^0),$ $(D^1, I^1, C^1))$ be a near-decomposition with respect to a partitioning $(\mathcal{L}^0, \mathcal{L}^1)$ of $\mathcal{L}$, and let $\sqsubseteq_{C_i}$ be an arbitrary preference ordering for action descriptions over $\mathcal{L}^i$, $i = 0, 1$. Then the following holds:*

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

(ii) Let X be a near-solution to $(D, I, C, \sqsubset_C)$, and let (X^0, X^1) be any partitioning of X with respect to $(\mathcal{L}^0, \mathcal{L}^1)$ such that $X^i \subseteq D^i$ and X^i has NOP if some dynamic query occurs positively in C^{1-i}, for $i = 0, 1$. Then, X^i is a near-solution to $(D^i, I^i, C^i, \sqsubset_{C^i})$, for $i = 0, 1$.

Proof. Let $T(X^0 \cup X^1) = \langle S, V, R\rangle$ and let $T(X^i) = \langle S^i, V^i, R^i\rangle$. We first show for any static query c, that $X^0 \cup X^1, s \models c$ if $c \in C^i$, $X^i, s^i \models c$, and $s \cap \mathcal{L}^i = s^i$. Since for each fluent literal L in c, $s^i \models L$ implies $s \models L$, and since $c \in \mathcal{L}_{C^i} \subseteq \mathcal{L}^i$ (i.e., c contains only fluent literals from $\mathcal{L}^i$), the claim follows. Conversely, for any static query c, it holds that $X^i, s^i \models c$ if $c \in C^i$, $X^0 \cup X^1, s \models c$, and $s^i = s \cap \mathcal{L}^i$. Again due to the fact that every fluent literal L in c is from $\mathcal{L}^i$, we conclude that $s \models L$ implies $s^i \models L$, which proves the claim. Therefore, we conclude for any static query $c \in \mathcal{L}_{C^i} \subseteq \mathcal{L}^i$ that there exists a state $s \in S$ such that $X^0 \cup X^1, s \models c$ iff there exists a state $s^i \in S^i$ such that $X^i, s^i \models c$. Moreover by the structure of S (cf. Lemma 1 *(i)*), $X^0 \cup X^1, s \models c$ for all $s \in S$ iff $X^i, s^i \models c$ for all $s^i \in S^i$. Hence, if C just contains static queries, then $X^0 \cup X^1$ entails C iff X^0 entails C^0 and X^1 entails C^1.

We next consider dynamic queries c that are either of the form **necessarily** Q **after** $A_1; \ldots; A_n$ or $\neg$**necessarily** Q **after** $A_1; \ldots; A_n$ and show the following: (1) $X^0 \cup X^1, s \models c$ if $c \in C^i$, $X^i, s^i \models c$, $s \cap \mathcal{L}^i = s^i$, and X^{1-i} has NOP if c is negative, or Q contains a negative dynamic query; (2) $X^i, s^i \models c$ if $c \in C^i$, $X^0 \cup X^1, s \models c$, $s^i = s \cap \mathcal{L}^i$,and X^{1-i} has NOP if c is positive, or Q contains a positive dynamic query. We proceed by induction on the nesting depth k of the query.

Base Case ($k = 0$): (1) Let c be positive and towards a contradiction consider a state $s \in S$, such that $s \cap \mathcal{L}^i = s^i$ and there exists a history $h = s, A_1, s_1, \ldots, s_{n-1}, A_n, s_n$, such that $s_n \not\models Q$. By Lemma 1 *(iii)*, every transition of the history $h^i = s^i, A_1, s_1 \cap \mathcal{L}^i, \ldots, s_{n-1} \cap \mathcal{L}^i, A_n, s_n \cap \mathcal{L}^i$ is in R^i. Furthermore, $s_n \not\models Q$ implies $s_n \cap \mathcal{L}^i \not\models Q$ because $c \in X^i$ and Q contains only static queries. Contradiction. If c is negative, then there exists a history $h^i = s^i, A_1, s^i_1, \ldots, s^i_{n-1}, A_n, s^i_n$ such that $s^i_n \not\models Q$. Since X^{1-i} has NOP, there exists a sequence of $n+1$ states, such that $h^{1-i} = s^{1-i}, \emptyset, s^{1-i}_1, \ldots, s^{1-i}_{n-1}, \emptyset, s^{1-i}_n$ is a history of X^{1-i}. By Lemma 1 *(ii)*, $h = s^i \cup s^{1-i}, A_1, \ldots, A_n, s^i_n \cup s^{1-i}_n$ is a history of $X^0 \cup X^1$. Furthermore, $s^i_n \not\models Q$ implies $s^i_n \cup s^{1-i}_n \not\models Q$ because $c \in X^i$ and Q contains only static queries. Contradiction. This proves (1) for $k = 0$.

(2) Let c be positive and towards a contradiction consider a state $s^i \in S^i$, such that $s^i = s \cap \mathcal{L}^i$ and there exists a history $h = s^i, A_1, s^i_1, \ldots, s^i_{n-1}, A_n, s^i_n$, such that $s^i_n \not\models Q$. Since X^{1-i} has NOP, there exists a sequence of $n+1$ states, such that $h^{1-i} = s^{1-i}, \emptyset, s^{1-i}_1, \ldots, s^{1-i}_{n-1}, \emptyset, s^{1-i}_n$ is a history of X^{1-i}. By Lemma 1 *(ii)*, $h = s^i \cup s^{1-i}, A_1, \ldots, A_n, s^i_n \cup s^{1-i}_n$ is a history of $X^0 \cup X^1$. Furthermore, $s^i_n \not\models Q$ implies

$s_n^i \cup s_n^{1-i} \not\models Q$ because $c \in X^i$ and Q contains only static queries. Contradiction. If c is negative, then there exists a history $h = s, A_1, s_1, \ldots, s_{n-1}, A_n, s_n$, such that $s_n \not\models Q$. By Lemma 1 *(iii)*, every transition of the history $h^i = s^i, A_1, s_1 \cap \mathcal{L}^i, \ldots, s_{n-1} \cap \mathcal{L}^i, A_n, s_n \cap \mathcal{L}^i$ is in R^i. Furthermore, $s_n \not\models Q$ implies $s_n \cap \mathcal{L}^i \not\models Q$ because $c \in X^i$ and Q contains only static queries. Contradiction. This proves (2) for $k = 0$.

Induction Step: Let (1) and (2) be true for dynamic queries of nesting depth at most $k - 1$ and consider a dynamic query c of nesting depth k. Then, Q contains only static queries and dynamic queries of nesting depth at most $k - 1$. Thus, (1) and (2) also hold for c, as follows easily by the arguments of the base case, replacing justifications by the fact that Q contains only static queries with a respective justification that Q contains only static queries and dynamic queries of nesting depth at most $k - 1$.

So far, we have shown that (1) and (2) hold for any basic query. By the structure of S (cf. Lemma 1 *(i)*), we conclude for any existential or universal query c that $X^0 \cup X^1 \models c$ if $c \in C^i$, $X^i \models c$, and X^{1-i} has NOP if c contains a negative dynamic query, as well as that $X^i \models c$ if $c \in C^i$, $X^0 \cup X^1 \models c$, and X^{1-i} has NOP if c contains a positive dynamic query. Therefore, $X^i \models C^i$ and X^{1-i} has NOP if C^i contains a negative dynamic query, for $i \in \{0, 1\}$, implies $X^0 \cup X^1 \models C$. Conversely, $X^0 \cup X^1 \models C$ and X^{1-i} has NOP if C^i contains a positive dynamic query implies $X^i \models C^i$, for $i \in \{0, 1\}$.

We now proceed with the proof of the theorem. Case *(i)*: Let X^i be a near-solution to $(D^i, I^i, C^i, \sqsubset_{C^i})$, for $i = 0, 1$. Suppose that, for $i = 0, 1$, X^i has NOP if some dynamic query occurs negatively in C^{1-i}. We show that $X^0 \cup X^1$ is a near-solution to $(D, I, C, \sqsubset_C)$. By Lemma 1 *(i)*, $X^0 \cup X^1$ is consistent, since X^0 and X^1 are consistent. Furthermore, $D_u^0 \cup D_u^1 \cup I^0 \cup I^1 \subseteq X^0 \cup X^1 \subseteq D \cup I$ follows from $D_u^0 \cup I^0 \subseteq X^0 \subseteq D^0 \cup I^0$ and $D_u^1 \cup I^1 \subseteq X^1 \subseteq D^1 \cup I^1$, respectively. Eventually, $X^0 \models C^0$ and $X^1 \models C^1$ implies $X^0 \cup X^1 \models C$. This proves that $X^0 \cup X^1$ is a near-solution to $(D, I, C, \sqsubset_C)$.

Case *(ii)*: Let X be a near-solution to $(D, I, C, \sqsubset_C)$, 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 has NOP if some dynamic query occurs positively in C^{1-i}. We prove that for $i = 0, 1$, X^i is a near-solution to $(D^i, I^i, C^i, \sqsubset_{C^i})$. Since X is consistent, also X^0 and X^1 are consistent. To see this, observe that w.l.o.g., if X^0 is inconsistent, then the static laws in X^0 are unsatisfiable, which implies X is unsatisfiable as well, a contradiction. Moreover, $D_u \cup I \subseteq X \subseteq D \cup I$ implies $D_u^0 \cup I^0 \subseteq X^0 \subseteq D^0 \cup I^0$ and $D_u^1 \cup I^1 \subseteq X^1 \subseteq D^1 \cup I^1$. Finally, $X^0 \cup X^1 \models C$ implies $X^0 \models C^0$ and $X^1 \models C^1$. Thus, X^0 and X^1 are near solutions to $(D^0, I^0, C^0, \sqsubset_{C^0})$ and $(D^1, I^1, C^1, \sqsubset_{C^1})$, respectively. □

Informally, the NOP property in Theorem 1 is needed to ensure that the transition diagrams of near-solutions to the sub-problems can be "combined". As already mentioned above, this is only necessary in the presence of dynamic queries.

For a full decomposition of an ADU problem, we need beyond a near decomposition also a factorization of the preference relation, which is formally defined as follows.

Definition 7 (Preference Factorization) *Let $\sqsubset_C$ be a preference relation for action descriptions over signature $\mathcal{L}$, and let $(\mathcal{L}^0, \mathcal{L}^1)$ be a partitioning of $\mathcal{L}$. A pair $(\sqsubset_{C^0}, \sqsubset_{C^1})$ of preference relations $\sqsubset_{C^i}$ for action descriptions over $\mathcal{L}^i$, $i = 0, 1$, is a* factorization of $\sqsubset_C$ with respect to $(\mathcal{L}^0, \mathcal{L}^1)$*, if for any action descriptions D, D' over $\mathcal{L}$ that are decomposable with respect to $(\mathcal{L}^0, \mathcal{L}^1)$, it holds that $D \sqsubset_C D'$ implies that either $D^0 \sqsubset_{C^0} D'^0 \wedge D'^1 \not\sqsubset_{C^1} D^1$ or $D'^0 \not\sqsubset_{C^0} D^0 \wedge D^1 \sqsubset_{C^1} D'^1$.*

Note that preference by strict subset inclusion ($\sqsubseteq_C = \subset$) is always factorizable (e.g., taking $\subset$ as the preference relations of the factorization). We also remark that if the set of queries C is decomposable with respect to $(\mathcal{L}^0, \mathcal{L}^1)$, then the query weight preference $<_{\mathit{weight}_q}$ is factorizable, provided that weights are nonnegative (for instance, taking the same weights for the factorization).

A full decomposition of an ADU problem is then as follows.

Definition 8 (ADU Decomposition) *A* decomposition *of an ADU problem $(D, I, C, \sqsubset_C)$ with respect to a partitioning $(\mathcal{L}^0, \mathcal{L}^1)$ of its signature $\mathcal{L}$ is a pair $((D^0, I^0, C^0, \sqsubset_{C^0}), (D^1, I^1, C^1, \sqsubset_{C^1}))$ such that $((D^0, I^0, C^0), (D^1, I^1, C^1))$ is a near-decomposition of $(D, I, C, \sqsubset_C)$ and $(\sqsubset_{C^0}, \sqsubset_{C^1})$ is a factorization of $\sqsubset_C$.*

The following result, which is the main result of this section regarding solutions of an ADU problem, is then easily obtained from Theorem 1.

Theorem 2 *Let $((D^0, I^0, C^0, \sqsubset_{C^0}), (D^1, I^1, C^1, \sqsubset_{C^1}))$ be a decomposition of an ADU problem $(D, I, C, \sqsubset_C)$ with respect to a partitioning $(\mathcal{L}^0, \mathcal{L}^1)$ of its signature $\mathcal{L}$. Suppose that either (i) no dynamic query occurs in C, or (ii) no dynamic query occurs in C^1. If X^i is a solution to $(D^i, I^i, C^i, \sqsubset_{C^i})$ for $i = 0, 1$, where in case (ii) X^1 has NOP, then $X^0 \cup X^1$ is a solution to $(D, I, C, \sqsubset_C)$. Furthermore, in case (i) every solution to $(D, I, C, \sqsubset_C)$ can be represented in this form.*

Item *(i)* states that we can fully decompose an ADU into two components, and that all solutions can be obtained by a simple combination of the solutions of the individual components. However, this works in general only in absence of dynamic queries (combining the transition graphs of the components is then unproblematic). Item *(ii)* accounts for possible dynamic queries in one component, which are unproblematic as long as solutions of the other have NOP. However, not all solutions can be composed from solutions of the components in general.

Example 4 Consider the ADU problem $(D \cup D', I, C, \subset)$, with D, I, and C as in Example 1, and D' as in Example 3. Since $X^0 = D \cup I \setminus \{(2)\}$ is a solution to $(D, I, C, \subset)$ (cf. Example 1), $X^1 = D'$ is (the unique) solution to $(D', \emptyset, \emptyset, \subset)$ (cf. Example 3), and D' has NOP (which is easily verified), by Theorem 2 *(ii)* $X^0 \cup X^1 = (D \cup D' \cup I) \setminus \{(2)\}$ is a solution to $(D \cup D', I, C, \subset)$. □

Example 5 Consider the ADU problem $(D \cup D', I, C, <_{weight_q})$, with D, I, C, and $weight_q$ as in Example 2, and D' as in Example 3. Again $X^0 \cup X^1 = (D \cup D' \cup I) \setminus \{(2)\}$ is a solution to $(D \cup D', I, C, <_{weight_q})$, as $X^0 = D \cup I \setminus \{(2)\}$ is a solution to $(D, I, C, <_{weight_q})$ (cf. Example 2), and as $X^1 = D'$ is (the unique) solution to $(D', \emptyset, \emptyset, <_{weight_q})$. By Theorem 1, $D_u \cup D' \cup I$ is a different near-solution to this ADU problem since $D_u \cup I$ is a near-solution to $(D, I, C, <_{weight_q})$. Moreover, setting the weight of query (12) to 0 (which amounts to assigning the preferred queries a 'don't care' status), it would be another solution. □

Theorem 1 provides a basis for decomposing an ADU into smaller ADUs that can be solved in a divide-and-conquer manner, and Theorem 2 shows some possible exploitation. These results can be integrated into algorithms for computing solutions, which we consider in Section 6 below. Finally, note that for our exemplary preference relations $\subset$ and $<_{weight_q}$ with non-negative weights, the benign properties of monotonicity and non-minimization with respect to $\mathcal{L}$, carry over to their standard factorizations (given by restricting the relation to the relevant domain) and can be recursively exploited.

5 Complexity Analysis

In this section, we investigate the computational complexity of relevant tasks for solving an ADU problem, including to decide whether a solution exists and whether given actions description is a solutions. The

$D \models C_o$ & $D \sqsubset_C D'$	**solution existence**	**solution checking**
in PSPACE	PSPACE	PSPACE
in Δ_i^P **(**$i > 1$**)**	Σ_i^P	Π_i^P
in P	NP	D^P

Table 1: Complexity of deciding solution existence and solution checking, depending on the complexity of the relevant subproblems (completeness results; hardness holds for fixed preference relation $\sqsubset_C$).

complexity of these tasks strongly depends on the complexity of deciding whether a given action description satisfies a set of (obligatory) queries (i.e., $D \models C_o$), and whether an action description is preferred over another action description under the given preference relation (i.e., $D \sqsubset_C D'$).

We first consider the worst-case complexity of the above mentioned subproblems as a parameter and derive upper bounds (in terms of membership results) for deciding whether an ADU problem has a solution, and for checking whether an action description is a solution to an ADU problem in a generic setting. We then 'instantiate' this generic setting by considering different classes (restricted sets) of queries which yield different complexities for deciding $D \models C_o$, and by studying concrete preference relations for which the complexity of deciding $D \sqsubset_C D'$ differs. In particular, we provide completeness results for the syntactic preference $\subset$ (for which deciding $D \sqsubset_C D'$ is polynomial) and for the semantic preference $<_{\mathit{weight}_q}$ (for which deciding $D \sqsubset_C D'$ ranges up to PSPACE) for the various classes of queries considered.

5.1 Generic Upper Bounds

Our main result on generic upper bounds, which however also gives the general picture of more precise complexity characterizations, is summarized in Table 1. Recall that PSPACE is the class of decision problems that can be decided by a (deterministic) Turing machine using space at most polynomial in the length of the input. PSPACE entails the so-called *polynomial hierarchy*, a sequence of classes defined as $\Delta_0^P = \Sigma_0^P = \Pi_0^P = \mathrm{P}$, and for $i \geq 0$, by $\Delta_{i+1}^P = \mathrm{P}^{\Sigma_i^P}$, $\Sigma_{i+1}^P = \mathrm{NP}^{\Sigma_i^P}$, and $\Pi_{i+1}^P = \mathrm{coNP}^{\Sigma_i^P}$. Finally, D^P is the class of decision problems whose *yes* instances are characterized by the "conjunction" of an NP problem and an independent coNP problem. The prototypical such problem is SAT-UNSAT, whose *yes* instances are pairs (F, G) of propositional formulas such that F is satisfiable and G is unsatisfiable; this problem is also complete for D^P. For a background in complexity theory, we refer to [49].

Informally, the results show that modulo the cost of deciding action queries and preference, the complexity of solution existence and checking increases at most by one level in the polynomial hierarchy, which is due to the exponential search space for a solution respectively a better solution candidate, which might be nondeterministically guessed. Since the search space can be traversed in polynomial space, there is no increase in complexity in the most general case.

We next formally establish Table 1. Given an ADU problem $(D, I, C, \sqsubset_C)$, let `Ccheck` denote the class of problems of deciding $D' \models C_o$ for any $D_u \cup I \subseteq D' \subseteq D \cup I$. Similarly, let `Pcheck` denote the class of problems of deciding whether $D_1 \sqsubset_C D_2$ holds, for action descriptions $D_u \cup I \subseteq D_i \subseteq D \cup I$ and $i \in \{1, 2\}$.

Theorem 3 *Deciding whether a given ADU problem* $(D, I, C, \sqsubset_C)$ *has a solution (or a near-solution) is (i) in* PSPACE *if* `Ccheck` *is in* PSPACE*, (ii) in* Σ_i^P *if* `Ccheck` *is in* Δ_i^P *and* $i > 1$*, (iii) in* NP *if* `Ccheck` *is*

in P.
Given an ADU problem $(D, I, C, \sqsubset_C)$ together with an action description D', deciding whether D' is a solution for it is (a) in PSPACE *if* `Ccheck` *and* `Pcheck` *are in* PSPACE*, (b) in Π_i^P if* `Ccheck` *and* `Pcheck` *are in Δ_i^P and $i > 1$, (c) in* D^P *if* `Ccheck` *and* `Pcheck` *are in* P.

Proof. Let $D = D_u \cup D_m$. In order to decide whether $(D, I, C, \sqsubset_C)$ has a solution, we can guess a near-solution D' such that $D_u \cup I \subseteq D' \subseteq D_u \cup I$, along with a state s for D' (to witness consistency), and check $D' \models C_o$ in polynomial space *(i)*, otherwise in polynomial time *(iii)*, respectively with the help of a Σ_{i-1}^P-oracle. This proves *(i)*, *(ii)*, and *(iii)*.

As for deciding whether a given D' is a solution, let us consider the complementary problem. We can nondeterministically guess D'' together with a state s'' and proceed as follows. We check in polynomial time whether $D_u \cup I \not\subseteq D'$, or $D' \not\subseteq D \cup I$. We also check whether D' is inconsistent *(a)* in polynomial space, respectively *(b)* with a single call to an NP-oracle. Deciding whether $D' \not\models C_o$ can be done in polynomial space in Case *(a)*, and in polynomial time with a Σ_{i-1}^P-oracle in Case *(b)*. Furthermore, we check in polynomial time whether $D_u \cup I \subset D'' \subseteq D \cup I$ and if D'' is consistent (whether s'' is state of D''). Two further checks decide whether $D'' \models C_o$ and $D' \sqsubset_C D''$ *(a)* in PSPACE, and *(b)* in polynomial time with the help of a Σ_{i-1}^P-oracle. Thus, the complementary problem is *(a)* in PSPACE, respectively *(b)* in Σ_i^P, proving *(a)* and *(b)*.

For *(c)* we nondeterministically guess a state s' of D' which we use to check consistency in polynomial time. Also we decide $D_u \cup I \subseteq D' \subseteq D_u \cup I$ in polynomial time. An independent coNP-check excludes more preferred near-solutions, i.e., the complementary problem of guessing D'' together with a state s'' and checking $D_u \cup I \subset D'' \subseteq D \cup I$, consistency (whether s'' is state of D''), $D'' \models C_o$, and $D' \sqsubset_C D''$ in polynomial time. This proves D^P-membership for *(c)*. □

Before we turn our attention to 'instantiating' this general result for ADU problems with different classes (restricted sets) of queries and concrete preference relations, which will yield precise complexity characterizations in terms of completeness results, we remark that to ease exposition, in the remainder of this section proofs are sketched, resembling the main arguments and constructions, while full proofs are given in Appendix A.

5.2 Query Entailment

As outlined in the beginning of this section, one of the two important subtasks in solving ADU problems is checking whether a set of queries is entailed by an action description. This subtask has a major influence on the complexity of finding solutions of an ADU problem. Therefore, besides considering arbitrary queries, we also investigate restricted classes of queries. In particular, when the maximal nesting depth of dynamic queries is fixed by an integer k, and when no dynamic queries occur at all.

Theorem 4 *Given an action description D and a set C of queries, deciding $D \models C$ is (i)* PSPACE*-complete in general, (ii) Θ_{k+3}^P-complete if k is the maximal nesting depth of dynamic queries in C, and (iii)* $\mathrm{P}_{\|}^{\mathrm{NP}}$*-complete if C does not involve dynamic queries.*

Here $\mathrm{P}_{\|}^{\mathrm{NP}}$ means polynomial-time with a single parallel evaluation of calls to an NP oracle. Similarly for $i > 1$, Θ_i^P is the class of problems that can be decided in polynomial time with parallel calls to a Σ_{i-1}^P oracle (alternatively, this class is often characterized by allowing $O(\log n)$ many oracle calls) [59].
Proof. Concerning *(i)* the result has been shown in [16]. Membership in Case *(iii)* follows from the fact that checking the truth of a negated universal query of the form $\neg$**ALWAYS** Q, where Q is a conjunction

of clauses over static queries of the form **holds** F or $\neg$**holds** F, is in NP. Hence, the complementary task, i.e., checking the truth of a positive universal query, **ALWAYS** Q, is in coNP. Thus, $D \models c$ is decided in polynomial time with a single parallel evaluation of n NP-oracle calls, given that n is the number of universal queries in c. Similarly, one proves in Case *(ii)* by induction on the nesting depth k, that $D \models c$ is decided in polynomial time with parallel Σ_{k+2}^{P}-oracle calls.

As for hardness, the problem in *(iii)* is reduced to the following $\mathrm{P}_{\|}^{\mathrm{NP}}$-hard decision version of *Maximum CNF Satisfiability* [37]: Given a Boolean formula F in *conjunctive normal form (CNF)* and an integer k, decide whether the maximum number of clauses in F that can be simultaneously satisfied by an interpretation is $0 \bmod k$. Consider a 3-CNF formula of the form $\bigwedge_{i=1}^{n} L_{i,1} \vee L_{i,2} \vee L_{i,3}$, where $L_{i,j}$, $1 \leq i \leq n$, $1 \leq j \leq 3$, is a literal over atoms $X = \{X_1, \ldots, X_m\}$, and the following action description D_1:

$$\left.\begin{array}{l}\textbf{caused } C_i \textbf{ if } L_{i,1}, \quad \textbf{caused } C_i \textbf{ if } L_{i,2}, \quad \textbf{caused } C_i \textbf{ if } L_{i,3},\\ \textbf{caused } \neg C_i \textbf{ if } \neg L_{i,1} \wedge \neg L_{i,2} \wedge \neg L_{i,3},\end{array}\right\} 1 \leq i \leq n$$

$$\begin{array}{ll}\textbf{caused } F_{1,1} \textbf{ if } C_1, & \textbf{caused } \neg F_{1,1} \textbf{ if } \neg C_1,\\ \textbf{caused } F_{1,0} \textbf{ if } \neg C_1, & \textbf{caused } \neg F_{1,0} \textbf{ if } C_1,\end{array}$$

$$\left.\begin{array}{l}\textbf{caused } F_{i,j} \textbf{ if } C_i \wedge F_{i-1,j-1},\\ \textbf{caused } \neg F_{i,j} \textbf{ if } \neg C_i \wedge F_{i-1,j-1},\end{array}\right\} 2 \leq i \leq n,\ 1 \leq j \leq i$$

$$\left.\begin{array}{l}\textbf{caused } F_{i,j} \textbf{ if } \neg C_i \wedge F_{i-1,j},\\ \textbf{caused } \neg F_{i,j} \textbf{ if } \quad C_i \wedge F_{i-1,j} \quad ,\end{array}\right\} 2 \leq i \leq n,\ 0 \leq j < i$$

Then $D_1 \models c_k$ iff the maximum number of clauses in F that can be simultaneously satisfied by an interpretation is $0 \bmod k$, where c_k is the following query:

$$\begin{array}{rcl} c_k & = & \textbf{ALWAYS holds } F_{n,0} \vee \\ & & \textbf{SOMETIMES holds } F_{n,k} \wedge \textbf{ALWAYS } (\neg\textbf{holds } F_{n,k+1} \wedge \ldots \wedge \neg\textbf{holds } F_{n,n}) \vee \\ & & \ldots \\ & & \textbf{SOMETIMES holds } F_{n,lk} \wedge \textbf{ALWAYS } (\neg\textbf{holds } F_{n,lk+1} \wedge \ldots \wedge \neg\textbf{holds } F_{n,n}), \end{array}$$

For hardness in Case *(ii)*, consider m *Quantified Boolean Formulas (QBFs)*

$$\Phi_l = Q_1 X_1^l\, Q_2\, X_2^l \cdots Q_n X_n^l\, E^l, \quad 1 \leq\ l \leq m$$

where $Q_i = \exists$ if $i \equiv 1 \bmod 2$ and $Q_i = \forall$ otherwise, X_i^k and X_j^l, $1 \leq i, j \leq n$ and $1 \leq k, l \leq m$, are pairwise disjunct sets of propositional variables if $i \neq j$ or $k \neq l$. and E^l is Boolean formula over atoms in $X^l = X_1^l \cup \cdots \cup X_n^l$, such that if Φ_l is false then $\Phi_{l+1}, \ldots, \Phi_m$ are false, too. Deciding whether the maximum index o, $1 \leq o \leq m$, such that Φ_o is true, is odd is Θ_{n+1}^{P}-hard [59]. The problem of deciding $D \models c$ for a query c with nesting depth k of dynamic queries is reduced to this problem, as follows.

Let $n = k + 2$, $1 \leq l \leq m$, and let the action description D_2 consist of the following statements:

$$\left.\begin{array}{l}\textbf{caused } F_i^l \textbf{ if } F_i^l \textbf{ after } A_{i-1},\\ \textbf{caused } \neg F_i^l \textbf{ if } \neg F_i^l \textbf{ after } A_{i-1},\end{array}\right\} 2 \leq i \leq n, F_i^l \in X_i^l$$

$$\left.\begin{array}{l}\textbf{caused } F_j^l \textbf{ after } A_{i-1} \wedge F_j^l,\\ \textbf{caused } \neg F_j^l \textbf{ after } A_{i-1} \wedge \neg F_j^l,\end{array}\right\} 2 \leq i \leq n, 1 \leq j \leq n, i \neq j, F_j^l \in X_j^l$$

Consider the query:

$$c_o = \begin{cases} \bigvee_{l=0}^{(m-3)/2}(\textbf{SOMETIMES } f^{2l+1} \wedge \textbf{ALWAYS } \neg f^{2l+2}) \vee f^m & \text{if } m \text{ is odd,} \\ \bigvee_{l=0}^{(m-2)/2}(\textbf{SOMETIMES } f^{2l+1} \wedge \textbf{ALWAYS } \neg f^{2l+2}) & \text{otherwise,} \end{cases}$$

where

$$f^l = p_1 \textbf{ necessarily } p_1 (\ldots (p_{n-1} \textbf{ necessarily } p_{n-1} \textbf{ holds } E^l \textbf{ after } \{A_{n-1}\}) \ldots) \textbf{ after } \{A_1\},$$

where $p_i = \neg$ if i is even and p_i is void otherwise, for $1 \leq i \leq n-1$. Then, the maximum index o such that Φ_o is true, is odd iff $D_2 \models c_o$. $\square$

5.3 Solution Existence

Equipped with these precise complexity characterizations of `Ccheck` for ADU problems of some classes of queries, we aim to exactly characterize the complexity of the solution finding tasks for these classes of queries and particular preference relations. Notice that checking whether a solution exists is independent of the concrete preference relation and its computation. This leads to the following result.

Theorem 5 *Deciding whether a given ADU problem $(D, I, C, \sqsubset_C)$ has a solution (or a near-solution) is (i)* PSPACE*-complete in general, (ii) Σ_{k+3}^P-complete, if k is the maximal nesting depth of dynamic queries in C_o, (iii) Σ_2^P-complete, if C_o does not involve dynamic queries, and (iv)* NP*-complete if $C_o = \emptyset$.*

Proof. Membership follows from Theorems 3 and 4, and Hardness in Case *(i)* follows from Theorem 4. For hardness in Case *(ii)*, let $n = k+2$ and let $\Phi = \exists Y\, Q_1 X_1 \cdots Q_n X_n\, E$ be a QBF, where $Q_i = \exists$ if $i \equiv 0 \bmod 2$ and $Q_i = \forall$ otherwise. Consider

$$D_u = D_2 \cup \{\textbf{caused } Y_i \textbf{ after } A_{i-1} \wedge Y_i, \textbf{caused } \neg Y_i \textbf{ after } A_{i-1} \wedge \neg Y_i \mid 2 \leq i \leq n\},$$

where D_2 is the action description from the proof of Theorem 4 with $l = 1$, $D_m = \{\textbf{caused } Y_i, \textbf{caused } \neg Y_i \mid Y_i \in Y\}$, $I = \emptyset$, $C = C_o \cup C_p$ with $C_p = \emptyset$ and

$$C_o = \{\textbf{ALWAYS } p_1 \textbf{ necessarily } p_1(\ldots (p_{n-1} \textbf{ necessarily } p_{n-1} \textbf{holds } E \textbf{ after } \{A_{n-1}\}) \ldots) \textbf{ after } \{A_1\}\},$$

where $p_i = \neg$ if i is odd, void otherwise, for $1 \leq i \leq n-1$. Then, there exists a solution to the action description update problem $(D_u \cup D_m, I, C, \sqsubset_C)$ iff Φ is true.

For *(iii)* let $\Phi = \exists Y\, \forall X\, E$ and consider the action description update problem $(D_u \cup D_m, I, C, \sqsubset_C)$, where $D_u = \emptyset$, $D_m = \{\textbf{caused } Y_i, \textbf{caused } \neg Y_i \mid Y_i \in Y\}$, $I = \emptyset$, and $C = C_o = \{\textbf{ALWAYS holds } E\}$. Again, the action description update problem $(D_u \cup D_m, I, C, \sqsubset_C)$ has a solution iff Φ is true.

Finally, for *(iv)*, let E be a Boolean formula over atoms Y and let us define $D_u = \{\textbf{caused } Y_1 \textbf{ if } \neg E, \textbf{caused } \neg Y_1 \textbf{ if } \neg E\}$, $D_m = \{\textbf{caused } Y_i, \textbf{caused } \neg Y_i \mid Y_i \in Y\}$, $I = \emptyset$, and $C = \emptyset$. Then, $(D_u \cup D_m, I, C, \sqsubset_C)$ has a solution iff E is satisfiable. $\square$

This result can be instantiated with any preference relation and yields completeness results for deciding the existence of a solution. When instantiated with our syntactic preference $\subset$, a remarkable consequence is the following. Deciding whether $D \cup I$ is a solution to an ADU problem $(D, I, C, \subset)$ has the same complexity as deciding $D \models C_o$ in general. Deciding the existence of an arbitrary solution is slightly harder than deciding $D \models C_o$ for restricted settings of queries in C_o. Intuitively, the additional computational effort accounts for the search of a solution candidate.

5.4 Solution Checking

We finally turn our attention to the recognition of solutions, where we provide respective results for the syntactic preference $\subset$ and the semantic preference $<_{\mathit{weight}_q}$. Again the problem turns out to be PSPACE-complete in general. Similarly as before, for $\subset$ in restricted query settings testing arbitrary solution candidates has higher complexity than testing $D \cup I$, which intuitively accounts for the additional maximality criterion to be checked for a solution.

Theorem 6 *Given an ADU problem $(D, I, C, \subset)$ and an action description D', deciding whether D' is a solution for it is (i)* PSPACE*-complete for general queries in C_o, (ii) Π^P_{k+3}-complete if k is the maximal nesting depth of dynamic queries in C_o, (iii) Π^P_2-complete if C_o does not involve dynamic queries, and (iv)* D^P*-complete if $C_o = \emptyset$.*

Proof. Membership follows from Theorem 3, observing that for any given action descriptions D' and D'', deciding $D' \subset D''$ can be done in polynomial time, i.e., that `Pcheck` is in P for $\subset$.

Hardness in Case *(i)* follows from Theorem 4. For *(ii)* let $n = k+2$ and let $\Phi = \forall Y\, Q_1 X_1 \cdots Q_n X_n\, E$ be a QBF, where $Q_i = \exists$ if $i \equiv 1 \bmod 2$ and $Q_i = \forall$ otherwise. Consider

$$D_u = D_2 \cup \{\textbf{caused } Y_i \textbf{ after } A_{i-1} \wedge Y_i, \textbf{caused } \neg Y_i \textbf{ after } A_{i-1} \wedge \neg Y_i \mid 2 \leq i \leq n\},$$

where D_2 is the action description from the proof of Theorem 4 with $l = 1$, $D_m = \{\textbf{caused } Y_i, \textbf{caused } \neg Y_i \mid Y_i \in Y\}$, $I = \emptyset$, and $C = C_o = \{\textbf{ALWAYS } f \vee g\}$, where

$$\begin{aligned} f &= p_1 \textbf{ necessarily } p_1 (\ldots (p_{n-1} \textbf{ necessarily } \bar{p}_{n-1} \textbf{ holds } E \textbf{ after } \{A_{n-1}\}) \ldots) \textbf{ after } \{A_1\}, \\ g &= \textstyle\bigwedge_{Y_i \in Y} \textbf{SOMETIMES holds } Y_i \wedge \textbf{SOMETIMES holds } \neg Y_i, \end{aligned}$$

where $p_i = \neg$ if i is odd, void otherwise, for $1 \leq i \leq n-1$, and $\bar{p}_n - 1 = \neg$ if n is odd and void otherwise. Then, D_u is a solution to the action description update problem $(D_u \cup D_m, I, C, \subset)$ iff Φ is true.

For *(iii)* let $\Phi = \forall Y\, \exists X\, E$ and consider the action description update problem $(D_u \cup D_m, I, C, \subset)$, where $D_u = \emptyset$, $D_m = \{\textbf{caused } Y_i, \textbf{caused } \neg Y_i \mid Y_i \in Y\}$, $I = \emptyset$, and $C = C_o = \{\textbf{ALWAYS } \neg\textbf{holds } E \vee g\}$, with g as before. The ADU problem $(D_u \cup D_m, I, C, \subset)$ has $D_u = \emptyset$ as a solution iff Φ is true.

Finally *(iv)*, let E_1 and E_2 be Boolean formulas over atoms Y_1 and Y_2, respectively. Consider $D_u = \{\textbf{caused } \neg F, \textbf{caused } F \textbf{ if } \neg E_1\}$, $D_m = \{\textbf{caused } F \textbf{ if } \neg E_2\}$, $I = \emptyset$, and $C = \emptyset$. Then, $(D_u \cup D_m, I, C, \subset)$ has solution D_u iff E_1 is satisfiable and E_2 is unsatisfiable. $\square$

We next consider solution checking for the semantic preference $<_{\mathit{weight}_q}$. Note that while `Pcheck` is polynomial for $\subset$, this is no longer the case for $<_{\mathit{weight}_q}$. However, intuitively whenever the complexity of `Pcheck` does not outweigh the complexity of `Ccheck`, i.e., when we do not allow for more complex queries in C_p than in C_o, then we stay within the same upper bounds as for $\subset$. Providing also matching lower bounds yields the following result, which differs from the previous one only if $C = \emptyset$. The intuitive reason is that for the syntactic preference also in this case a maximality check is needed to recognize a solution, while the semantic preference is indifferent for $C = \emptyset$, which means that basically a consistency check is sufficient and that every near-solution also is a solution.

Theorem 7 *Given an ADU problem $(D, I, C, <_{\mathit{weight}_q})$ and an action description D', deciding whether D' is a solution for it is (i)* PSPACE*-complete for general queries in C, (ii) Π^P_{k+3}-complete if k is the maximal nesting depth of dynamic queries in C, (iii) Π^P_2-complete if C does not involve dynamic queries, and (iv)* NP*-complete if $C = \emptyset$.*

Proof. Membership for *(i), (ii),* and *(iii)* follows easily from Theorems 3 and 4. For *(iv),* i.e. $C = \emptyset$, `Pcheck` is trivial for $<_{weight_q}$, hence we can decide whether D' is a solution essentially by checking consistency.

Hardness in Case *(i)* follows from Theorem 4. For *(ii)* let $n = k+2$ and consider Φ, D_u, D_m, I, and C_o from the proof of Theorem 6 *(ii).* Additionally, let $C_p = \{$**ALWAYS holds** Y_i, **ALWAYS holds** $\neg Y_i \mid Y_i \in Y\}$ and consider a weight of 1 for each $c \in C_p$. Then, D_u is a solution to $(D_u \cup D_m, I, C_o \cup C_p, <_{weight_q})$ iff it is a solution to $(D_u \cup D_m, I, C_o, \subset)$.

For *(iii)* consider Φ, D, I, and C_o from the proof of Theorem 6 *(ii).* Again, let $C_p = \{$**ALWAYS holds** Y_i, **ALWAYS holds** $\neg Y_i \mid Y_i \in Y\}$ with weight 1 for each $c \in C_p$. Then, for the same reason as above, D_u is a solution to $(D_u \cup D_m, I, C_o \cup C_p, <_{weight_q})$ iff it is a solution to $(D_u \cup D_m, I, C_o, \subset)$.

Finally *(iv),* let E be a Boolean formula over atoms Y and consider the ADU problem given by $D_u = \{$**caused** Y_1 **if** $\neg E$, **caused** $\neg Y_1$ **if** $\neg E\}$, $D_m = \emptyset$, $I = \emptyset$, and $C = \emptyset$. Then, D_u is a solution to $(D_u \cup D_m, I, C, <_{weight_q})$ iff E is satisfiable. □

Hence, even recognizing solutions is quite hard. However, recognizing near-solutions is easier for restricted sets of queries (Θ^P_{k+3}-complete if the maximal nesting depth of dynamic queries in C is k, $\mathrm{P}^{\mathrm{NP}}_{\parallel}$-complete if C has no dynamic queries, and NP-complete if $C = \emptyset$). This follows easily from Theorem 4.

6 Computing Solutions

Equipped with a clear picture of the computational cost in terms of complexity for the relevant (sub-)tasks of solving an ADU problem, we now turn to the issue of computing solutions using dedicated, deterministic algorithms.

6.1 General Algorithms

With an oracle for near-solutions, in case of the syntactic preference $\subset$, we can incrementally compute a solution to an ADU problem $(D, I, C, \subset)$ where $D = D_u \cup D_m$, in polynomial time using the algorithm in Figure 5. By virtue of Theorems 5 and 6, this algorithm is worst case optimal, even when the nesting depth k of dynamic queries is restricted, since computing a solution needs the power of a Σ^P_{k+3} oracle. If the existence test for a near-solution of $(D_u \cup D_m, I, C, \subset)$ in Step 1 or Step 2 in fact returns some near-solution D^n, then we can replace the respective assignment to D' by the assignments $D' := D^n$ and $D_m := D_m \setminus D^n$.

We remark that for semantic preferences, like $<_{weight_q}$, such a deterministic polynomial time procedure for computing solutions, using an oracle for computing near solutions, does not work in general. However, in certain cases an oracle for near-solutions can be used effectively in a similar way. For instance, whenever the queries in C_p can be strictly ordered according to their (non-negative) weights, such that no subset of queries that are before a query c in the ordering can sum up to a higher weight than c. Then, in a procedure similar to SOLUTION, one can iterate through the set of queries C_p once, using the oracle to determine whether near-solutions exist to the slightly modified problem where certain queries from C_p are added to C_o in order to determine the set of queries from C_p satisfied by an optimal solution. Once this set is known, any near-solution of the problem where these queries are added to C_o, is a solution to the original problem.

For the general case of $<_{weight_q}$ with nonnegative weights, for instance, a branch and bound algorithm can be devised from Algorithm SOLUTION that uses an oracle for near-solutions to compute an initial solution candidate and, throughout the computation, better candidates as usual in the style of an anytime algorithm.

Algorithm SOLUTION$_\subset$
Input: an ADU problem $(D, I, C, \subset)$
Output: some solution of $(D, I, C, \subset)$, if one exists.

```
Step 1  if $(D_u \cup D_m, I, C, \subset)$ has a near-solution
            then $D' := D_u$  else halt; // no solution exists

Step 2  while $D_m \neq \emptyset$ do
            choose some $\ell \in D_m$;
            $D_u := D' \cup \{\ell\}$; $D_m := D_m \setminus \{\ell\}$;
            if $(D_u \cup D_m, I, C, \subset)$ has a near-solution then $D' := D' \cup \{\ell\}$;
          endwhile;

Step 3  output $D'$.
```
□

Figure 5: Algorithm to compute some solution preferred by set-inclusion

For other preferences $\sqsubset_C$, algorithms will have to be developed that similarly exploit the structure of $\sqsubset_C$ to prune the search space effectively. If $\sqsubset_C$ is monotone with respect to the underlying signature, we may adapt Algorithm SOLUTION similarly as for $<_{weight_q}$ to a branch and bound algorithm that aims at enumerating near-solutions (for which e.g. techniques as in [12] are useful) and cuts branches in the search tree if no better near-solutions compared to the currently most preferred ones, $D_1, \ldots, D_m$, can be found in them; more precisely, any branch for a (partial) near-solution D can be cut such that $D \cup \{\ell_1, \ldots, \ell_m\} \sqsubset_C D_i$ for some D_i. Note that every solution preferred under $\sqsubset_C$ is also preferred under set-inclusion, and we can adapt in the same way the variant of Algorithm SOLUTION that exploits near-solutions returned by the oracle. This scheme may be further refined, as usual, by exploiting properties like solution dominance (for each possible solution D' such that $D \subseteq D' \subseteq \{\ell_1, \ldots, \ell_m\}$, one of the solutions D_i is preferred); further investigation remains for future work.

6.2 Near-Solutions

Near-solutions to a given ADU problem may be nondeterministically computed as in the membership part of Theorem 5, or may be obtained from a QBF encoding using a QBF solver. We present here a different computation method, which builds on update descriptions and "update fluent sets." Roughly, rather than to consider varying update descriptions, in this method the problem is compiled into a single action description, called the *update description*, in which special update fluents govern the inclusion and exclusion of causal laws. Determining an update amounts then to determine an appropriate update fluent set, which is semantically defined and may be computed by query answering and state set generation algorithms.

Definition 9 *Let $D = D_u \cup D_m$ be an action description with signature $\langle \mathbf{F}, \mathbf{A} \rangle$. The* update description *$U(D)$ 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 (4) in D_m with a fluent $H_i \in \mathbf{H}$:*

$$\textbf{caused } L \textbf{ if } G \wedge H_i, \tag{21}$$

```
Algorithm NEAR-SOLUTION$(D, I, C, \sqsubset_C)$
Input: an ADU problem $(D, I, C, \sqsubset_C)$
Output: some near-solution of $(D, I, C, \sqsubset_C)$, if one exists.

  Step 1  if $D \cup I$ is consistent and $D \cup I \models C_o$ then output $D \cup I$ and halt;
  Step 2  construct the update description $U$ of $D \cup I = D_u \cup I \cup D_m$;
  Step 3  if some update fluent set $\mathbf{M}$ for $U$ relative to $C_o$ exists
              then take an arbitrary such $\mathbf{M}$ else halt; // no near-solution exists
  Step 4  identify the set $W$ of causal laws in $D_m$ labeled by the elements of $\mathbf{M}$;
  Step 5  output $D_u \cup W \cup I$.
```

Figure 6: Algorithm to compute some near-solution

and each dynamic law (5) in D_m with a fluent $H_i \in \mathbf{H}$:

$$\textbf{caused } L \textbf{ if } G \textbf{ after } H \wedge H_i, \tag{22}$$

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:*

$$\textbf{inertial } H_i, \neg H_i. \tag{23}$$

We next define update fluent sets. To this end, we define, given an action description $D_u \cup D_m$ and a set of conditions C on the same signature, a partitioning $S^U_C, S^U_{\neg C}$ of the state set S^U of the update description $U = U(D)$ of $D_u \cup D_m$ having the set $\mathbf{H}$ of update fluents, as follows. For any two states $s, s' \in S^U$ let $s =_{\mathbf{H}} s'$ iff $s \cap \mathbf{H} = s' \cap \mathbf{H}$, and let $S^U_{\mathbf{H},s} = \{s' \in S^U \mid s' =_{\mathbf{H}} s\}$. Given a query c and state $s \in S^U$, we say that c *holds at* s wrt. $S^U_{\mathbf{H},s}$, if in case *(i)* c is existential (8), $E, s' \models Q$ holds at some $s' \in S^U_{\mathbf{H},s}$; *(ii)* c is universal (9), $E, s' \models Q$ holds at all $s' \in S^U_{\mathbf{H},s}$; *(iii)* c is a Boolean combination of existential and universal queries c_i, the combination evaluates to true if each c_i has the value at s wrt. $S^U_{\mathbf{H},s}$. Then, $S^U_C = \{s \in S^U \mid c$ holds at s wrt. $S^U_{\mathbf{H},s}$, for all $c \in C\}$. Furthermore, in the rest of this section, we identify states with the sets of fluents which are true at that state.

Definition 10 *An* update (fluent) set *for U relative to C is a set $\mathbf{M} \subseteq \mathbf{H}$ such that (i) $s \cap \mathbf{H} = \mathbf{M}$ for some $s \in S^U$, and (ii) $S^U_{\mathbf{H},s} \subseteq S^U_C$.*

With the notions above, we can compute a near-solution to an ADU problem $(D, I, C, \sqsubset_C)$, where $D = D_u \cup D_m$, with the algorithm NEAR-SOLUTION shown in Figure 6. The key to its correctness is the following proposition.

Proposition 2 *Let $(D, I, C, \sqsubset_C)$ be an ADU problem, with $D = D_u \cup D_m$. Let U be the update description of $D \cup I = D_u \cup I \cup D_m$, and let W denote a subset of D_m containing laws labeled by the elements $\mathbf{M} \subseteq \mathbf{H}$ in U. Then $D' = D_u \cup I \cup W$ is a near-solution to $(D, I, C, \sqsubset_C)$ iff M is an update set for U relative to C_o.*

The proof of this correspondence result, which is technically involving, is given in Appendix B. It follows the intuition that by considering an update set for $D \cup I$ relative to C_o and 'adding' the corresponding labeled laws (which by construction are from D_m) to $D_u \cup I$, one ends up with an action description D' that satisfies C_o. The essential argument is by showing that for any state s of D', $s \cup \mathbf{M}$ is a state of U, and due to Condition *(ii)* of Definition 10 it is a state in $S^U_{C_o}$, which in turn implies that $s \in S^{D'}_{C_o}$, i.e., that $D' \models C_o$. Moreover, Condition *(i)* of Definition 10 guarantees that D' is consistent. Vice versa, to every near-solution corresponds an update set M, given by the labels of the modifiable laws included in the near-solution.

From Proposition 2, the correctness of algorithm NEAR-SOLUTION is then easily established.

Theorem 8 *Let $(D, I, C, \sqsubset_C)$ be an ADU problem, with $D = D_u \cup D_m$. Then Algorithm* NEAR-SOLUTION *outputs some near-solution of $(D, I, C, \sqsubset_C)$ if and only if some near-solution of $(D, I, C, \sqsubset_C)$ exists.*

We observe that for $\subset$ as the preference ordering $\sqsubset_C$, the algorithm can be easily adapted to find solutions instead of near solutions: to this end, in Step 3 we take a maximal one. We also note that Step 1 is not necessary as far as mere computation of any near-solution is concerned. However, in the view of ADU problem solving it may be worthwhile to particularly return $D \cup I$ first, if it is a near-solution, since it constitutes the case where I can be incorporated without modification to D. This is in particular relevant for preference relations $\sqsubset_C$ that are non-minimizing, as then in fact a solution is output.

Example 6 Consider an ADU problem $(D, I, C, \sqsubset_C)$ given by D, I, and C as presented in Example 1. Note that $D \cup I \not\models C$ (as explained in Example 1). We obtain the following update description U of $D_u \cup I \cup D_m$, which contains $D_u \cup I$ and the laws:

$$\begin{array}{l}
\textbf{caused } \mathit{TvON} \textbf{ if } \mathit{PowerON}, H_1,\\
\textbf{caused } \neg\,\mathit{TvON} \textbf{ if } \neg \mathit{PowerON}, H_2,\\
\textbf{inertial } H_i, \neg H_i \qquad (1 \leq i \leq 2).
\end{array}$$

According to the transition diagram described by U, we have that action PushPB_{RC} is not executable, i.e., query (3) fails, at any state $s \supseteq \{\mathit{PowerON}, \mathit{TvON}, H_1\}$. Moreover, at any state $s \supseteq \{\mathit{PowerON}, \mathit{TvON}\}$ such that $H_2 \notin s$, query (12) fails due to missing causation for $\neg\,\mathit{TvON}$. Query (17), however, is satisfied at every state of U. We thus obtain

$$S^U_{\neg C} = \{s \in S^U \mid s \text{ satisfies } H_1 \vee \neg H_2\},$$

and, for instance, $\{\mathit{PowerON}, \mathit{TvON}, H_2\} \in S^U_C$. Therefore, $\{H_2\}$ is an update set for U relative to C, and obviously it is the only one. Hence, if we add the law labeled by H_2 to $D_u \cup I$, or equivalently remove the law (2) that is labeled by H_1 from $D \cup I$, we obtain a near-solution to the problem (cf. also Example 1). □

Example 7 Consider a slight variant of the previous Example 6, where also the dynamic laws in D (except for the inertia laws) are modifiable, and with the following causal laws added to D_m:

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

The transition diagram described by $D \cup I$ is the same as in Figure 4, and thus for the same reasons as mentioned in Example 1, $D \cup I \not\models C$. The update description U of $D_u \cup I \cup D_m$ consists of $D_u \cup I$, the labeled laws as presented in Example 6, and the following causal laws:

$$\begin{array}{l}
\textbf{caused } \mathit{PowerON} \textbf{ after } \mathit{PushPB}_{TV} \wedge \neg \mathit{PowerON}, H_3, \\
\textbf{caused } \neg \mathit{PowerON} \textbf{ after } \mathit{PushPB}_{TV} \wedge \mathit{PowerON}, H_4, \\
\textbf{caused } \mathit{TvON} \textbf{ after } \mathit{PushPB}_{TV} \wedge \neg \mathit{PowerON}, H_5, \\
\textbf{caused } \neg \mathit{TvON} \textbf{ after } \mathit{PushPB}_{TV} \wedge \mathit{PowerON}, H_6, \\
\textbf{inertial } H_i, \neg H_i \qquad (3 \leq i \leq 6).
\end{array}$$

According to the transition diagram described by U, query (3) still fails, since the action PushPB_{RC} is not executable whenever $s \supseteq \{\mathit{PowerON}, \mathit{TvON}, H_1\}$. Let us consider the remaining states s of U, i.e., only those such that $H_1 \notin s$. We first observe that a failure of query (12) is witnessed by any such state $s \supseteq \{\mathit{PowerON}, \mathit{TvON}\}$ such that $H_6 \notin s$ and either $H_2 \notin s$ or $H_4 \notin s$ (or both), since there is no causation for $\neg \mathit{TvON}$ when executing PushPB_{TV}. Finally, query (17) does not hold at any such state s where the power and the TV are off, i.e., $s \cap \{\mathit{PowerON}, \mathit{TvON}\} = \emptyset$, if $\{H_2, H_5\} \subseteq s$ and $H_3 \notin s$. More formally,

$$S^U_{\neg C} = \{s \in S^U \mid s \text{ satisfies } H_1 \vee (\neg H_6 \wedge (\neg H_2 \vee \neg H_4)) \vee (\neg H_3 \wedge H_2 \wedge H_5)\}.$$

Two update sets for U relative to C are $\{H_3, H_4, H_5, H_6\}$ and $\{H_2, H_3, H_4, H_6\}$. (That they actually constitute update sets is witnessed, e.g., by $\{H_3, H_4, H_5, H_6\} \in S^U_C$ and $\{H_2, H_3, H_4, H_6\} \in S^U_C$, respectively.) We may choose either one and, by adding the corresponding causal laws to $D_u \cup I$, we get a near-solution to the problem. Note however, that in case of $\sqsubset_C = \subset$, for instance, none of the near-solutions is a solution, as removing (2) is sufficient. This is reflected by the (maximal) update set $\{H_2, H_3, H_4, H_5, H_6\}$. $\Box$

Algorithm NEAR-SOLUTION can be run in polynomial space, and is thus within the worst case optimal bounds. Indeed, the update description U for D and C can be easily computed in polynomial time, and after the consistency and entailment check in Step 1, the bulk of the work is with Step 3, i.e., to compute an update set $\mathbf{M}$. Here, we can resort to different methods. If the full state set S^U of U would be explicitly given, then Step 3 is clearly feasible in polynomial time. Otherwise, we can use an algorithm that enumerates S^U, and for each state s generated take $s \cap \mathbf{H}$ as candidate update set $\mathbf{M}$ for which condition *(ii)* $S^U_{\mathbf{H},s} \subseteq S^U_C$ is tested using query answering; a brief outline is as follows. Let $F_s = \bigwedge_{H_i \in \mathbf{M}} H_i \wedge \bigwedge_{H_i \in \mathbf{H} \setminus \mathbf{M}} \neg H_i$; intuitively, F_s holds at a state s' iff s' belongs to $S^U_{\mathbf{H},s}$. Then, for each existential query c of form (8), define $c_s =$ **SOMETIMES holds** $F_s \wedge Q$, and for each universal query c of form (9), let $c_s =$ **ALWAYS holds** $F_s \supset Q$. For a Boolean combination c of existential and universal queries, we define c_s as the query obtained by rewriting each occurrence of an existential or universal query as described above. Then $S^U_{\mathbf{H},s} \subseteq S^U_C$ is equivalent to entailment $U \models c_s$ for each query c in C.

Thus, one can build algorithms to compute near-solutions of an ADU on top of basic reasoning services for action descriptions that generate sets of states and perform query answering (as supported e.g. in AD-Query [19], under some limitations), which are applied to the update description $U(D)$. Compared to a simple search over the near-solution candidates D' such that $D_u \cup I \subseteq D' \subseteq D \cup I$ and testing whether $D' \models C_o$, this approach has some attractive advantages. One is that we may compile the transition diagram of $U(D)$ into an efficient representation (e.g., into binary decision diagrams that are customary in efficient processing of transition-based formalisms), and perform state generation and query answering over this single representation, rather than to consider reasoning over varying transition diagrams, which may have considerable management cost (setting up data structures anew, etc.) at least without further precaution and effort.

Furthermore, the update description is a useful basis for iterated Markovian (history-less) updates under lazy evaluation, and more generally for realizing non-Markovian semantics of sequences of updates $I_1, \dots, I_k$, in analogy to update programs in the context of logic programming updates [4, 18]. In the Markovian case, the result of updating an action description D is obtained by incorporating the I_i, $i = 1, \dots, k$ one after the other into D. The update description $U(D)$ may be generalized to capture such iterative updates rather easily, by using time stamped copies of action descriptions that are suitably linked, and modifying the preference ordering $\sqsubset_C$ appropriately into a prioritized version. In the non-Markovian case, linkage and preference ordering can be tailored to realize particular update semantics. Investigating this is left for further work.

7 Discussion

7.1 Related Work

Updating and revising knowledge bases has been studied extensively in the context of both databases and AI, with different approaches, and in various representation frameworks, e.g. [61, 36, 54, 28, 29, 18, 46, 33, 34, 35, 56, 41, 44, 57, 64, 62, 32, 38, 16]. The relation of this problem to reasoning about actions has been identified earlier [60, 55, 51], since the effects of executing an action in a given situation can be modeled as the change of a theory representing the current state by a formula representing the action effects. However, compared to reasoning in action languages, such an approach leaves the action under consideration and its effects rather implicit. Therefore, we restrict our attention to those works that either treat the notion of an action explicitly in the language, or that are otherwise more closely related to our work.

Sakama and Inoue's work [56] is similar to our work in that it also studies update problems in a non-monotonic 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 I, inconsistency removal from D, and view update of $D = D_u \cup D_m$ by some new information I. In the context of reasoning about actions and change, these kinds of updates are expressible as ADU problems $(D, I, \emptyset, \subset)$, $(D, \emptyset, \emptyset, \subset)$, and $(D_u \cup D_m \cup I), \emptyset, \emptyset, \subset)$. Sakama and Inoue show in [56] that checking for solution existence is NP-hard for each problem; this complies with Theorem 5 *(iii)*. An important difference to [56] is that in an ADU problem $(D, I, C, \subset)$, the 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 [41] and Liberatore [44] 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, I, 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

$$E \textbf{ occurs at } t_i, \tag{24}$$

$$P \textbf{ holds at } t_i, \tag{25}$$

where E is an action name, P is a fluent name, and t_i is a time stamp; a state s satisfies a query (24) resp. (25) if, for some history (10) 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 in [63]. 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 [63] can be achieved in our framework, by describing possible scenarios or formulas as queries.

Some of the related work mentioned above, like [5, 45, 2, 32], study action description updates in connection with the problem of elaboration tolerance. The goal is to answer the following question: how can an action description be updated to tolerate new elaborations on the action domain? [32] studies the update problem in the context of dynamic logic [30]. Here action domains are represented in a simplified version of dynamic logic. An action domain description consists of static laws (e.g., $Up \rightarrow Light$, which expresses that "if the switch is up then the light is on"), effect laws for actions (e.g., $\neg Up \rightarrow [Toggle]\, Up$, which expresses that "whenever the switch is down, after toggling it, the room is lit up"), and executability laws for actions (e.g., $\neg Broken \rightarrow \langle Toggle\rangle\top$, which expresses that "toggle can not be executed if the switch is broken"). To handle the frame problem and the ramification problem, a consequence relation is built (in a meta-language) over the action description. Note that the action description language $\mathcal{C}$ does not require such a meta-language to be able to handle these problems. In this formal framework for reasoning about actions and change, the authors consider revising beliefs about states of the world (as in, e.g., [33, 57]), as well as revising beliefs about the action laws. They update action descriptions with respect to some elaborations (described also by causal laws), by modifying the causal laws in the action description by first "contraction" and then "expansion". In the end, the antecedents of some causal laws in the action description are strengthened with respect to the new elaborations. Consider the example above; during a blackout, the agent toggles the switch when it is down, and the room is still dark. Such an elaboration is described by a causal law, like $Blackout \rightarrow [Toggle]Light$. The action description is modified by this elaboration, by first contracting the effect laws (e.g., $\neg Up \rightarrow [Toggle]\, Up$) and then expanding the theory with the weakened laws (e.g., $\neg Up \wedge \neg Blackout \rightarrow [Toggle]\, Up$). The idea behind modifying a theory with an elaboration of the form $\phi \rightarrow [a]\psi$ in this way, is to ensure two conditions when ψ does not hold: first a still has the effect ψ; and second a has no effect except those literals that are consequences of $\neg\psi$. The semantics of such syntactic operations are given in terms of changes (e.g., addition/removal of edges) in the transition diagram. Note that [32] modifies causal laws to tolerate elaborations, whereas we add new causal laws (which may be obtained from some observations, or which may describe some elaborations) to the original description and furthermore we drop a minimal set of causal laws from the original theory so that given queries (which may describe some desired/preferred conditions on the domain) are entailed by the updated description.

Another related work that studies action description updates, for elaboration tolerance, is [2]. The authors introduce an action description language, called Evolp Action Programs (EAPs), built upon the update language Evolp [3]. This language can be used to represent action domains, as well as their updates due to some elaborations. An action domain description consists of static rules (e.g., $Light \leftarrow Up$), dynamic rules (e.g., $effect(Light \leftarrow Up) \leftarrow Toggle, \neg Up$ which expresses that, if at some step n the switch is down, then $Light \leftarrow Up$ becomes true at step $n+1$), inertial declarations (e.g., $inertial(Light)$), and initialize declarations (e.g., $initialize(Light)$ which stands for $Light \leftarrow prev(Light)$ where $prev(F)$ is a new atom introduced for describing the value of fluent F in the previous state) introduced for representing inertia. Note that in the action language $\mathcal{C}$, there is no need to introduce new atoms to be able to handle the frame problem. An elaboration is encoded as a separate action description D, and then "asserted" to the main description, using the $assert$ construct of Evolp. Adding to the main description $assert(D)$ is different from adding D. For instance, consider adding $assert(Light \leftarrow Up) \leftarrow Toggle$ to an action description. Then, when the switch is toggled, the rule $Light \leftarrow Up$ remains inertially true until its truth is possibly deleted afterwards. The semantics of an EAP (an thus the $assert$ construct) is given by means of stable models [25]. Note that [2] is similar to our work in that updates that consist of static/dynamic rules are described in the same language as the action description. On the other hand, the language of [2] allows us to talk about, as a part of the updates, changes over rules (using the assert construct).

The works by Lifschitz [45] and by Balduccini and Gelfond [5] are similar to [2] in that they also modify action descriptions with respect to new elaborations, by means of adding causal laws, in the sense of additive elaboration tolerance [48, 50]. Lifschitz describes in [45] an action domain in language $\mathcal{C}$ such that every causal law is defeasible (by means of an abnormality predicate). To formulate some other variations of the domain, the agent can just add new causal laws, some of which "disable" some existing causal laws. In [5], the authors extend an action description, encoded as a logic program, with "consistency restoring" rules, so that when the action description and given observations are incompatible, these rules can be "applied" to get some consistent answer set. This, however, is more geared towards handling exceptions. In [45] and [5], the causal laws of the original domain description are not modified.

Concerning results on the computational complexity, Eiter and Gottlob [21] study a number of syntax-based as well as model-based knowledge base revision operators and provide precise complexity characterizations for the problem of checking whether a given formula is derivable from a revised (updated) knowledge base by reducing the problem to the evaluation of counterfactuals. Herzig [31] improved these complexity bounds for restricted settings under Winslett's Possible Models Approach. Liberatore [43] considers further approaches for belief update from the literature, derived corresponding complexity results, and extended them to the problem of iterated update. Cayrol *et al.* [10] study pre-ordered belief bases. A priority relation on belief bases induces a preference relation on the set of subsets of a belief base, which can be used to select preferred subsets in order to define (refined) change operators. They provide complexity results for inclusion-based preference, maximum cardinality, and lexicographic preference by investigating entailment of a formula by a set of consistent subsets of a belief base under various entailment principles including credulous, skeptical, and so-called argumentative entailment. Baral and Zhang [6] considers the complexity of model checking for knowledge update. As for traditional belief update, the relation to reasoning about actions consists in regarding the effects of an action as an update to the current state. However, motivated by sensing actions that do not change the world, Baral and Zhang distinguish knowledge updates as belief updates where changes not only correspond to alterations of the real world but may also be affect an agent's knowledge about the world. They give a model theoretic account of knowledge updates based on modal logics, show that the complexity of model checking is on the second layer of the polynomial hierarchy, and identify tractable subclasses.

More closely related to our work are investigations concerning the complexity of reasoning about actions in an action language. For the action language $\mathcal{A}$, Liberatore [42] establishes, for instance, NP-completeness of consistency checking and coNP-completeness for entailment, which essentially amounts to checking whether

$$D \models \textbf{ALWAYS necessarily } (\textbf{holds } F) \textbf{ after } A_1; \ldots; A_n,$$

for a given action description D, a fluent F, and a sequence of actions $A_1; \ldots; A_n$ in our setting. Lang *et al.* [39] refer to this as "progression problem;" they investigated its computational complexity for simple causal action theories which constitute a special case of causal theories in different languages, in particular capturing the fragment of action language $\mathcal{C}$ that we considered. Besides the progression problem, the complexity of other reasoning tasks, including executability and determinism, is addressed in this framework which is further extended to so-called generalized action theories. We remark that, like for progression, several of these results can be obtained as special cases of deciding $D \models c$ for particular queries c in our setting. Moreover, to the best of our knowledge, the complexity of deciding action queries has not been addressed so far (apart from the PSPACE result for the general case for the query language we considered, which has been proven in [16]), let alone the problem of updating action descriptions in the presence of meta-knowledge expressed by action queries.

7.2 Nature of Change

As already briefly mentioned in Section 3, our notion of action update has more of a belief revision than a belief update flavor. This view is supported by a deeper analysis of change in connection with reasoning about actions and change [38, 53]. Lang [38] describes a scope for revision and for update, and he notices that, as pointed out by [23, 24], the scope can not be simply decided by whether the theory is about static vs. dynamic worlds. Then, as also pointed out by [9, 13], Lang relates revision and update by means of backward-forward reasoning, in particular, by means of action progression. According to [38], belief revision is to correct some initial beliefs about the past/present/future state of the world by some observations about the past/present state of the world. On the other hand, belief update by some formula α corresponds to progressing the theory by a specific feedback-free action that will make α true with respect to a given update operator; here α does not describe observations. In this framework, Lang says that our approach is closer to a revision process than to an update; however, since our approach changes the transition diagram of an action description, it is meaningful to consider it as an update process as well.

The AGM and KM postulates [1, 36] are based on an underlying logic that is monotonic in nature. However, the action language $\mathcal{C}$ we consider is nonmonotonic. For instance, if D consists of the single law

$$\textbf{caused } P \textbf{ if } P$$

where P is the single fluent and there are no actions, then the transition diagram described by D, $T(D)$, has a single state $s = \{P\}$. Thus the causal law **caused** P is satisfied by $T(D)$ (equivalently, $D \models$ **ALWAYS holds** P), and can be seen a semantic consequence of D. However, if we add

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

to D, then $T(D)$ has another state $s' = \{\neg P\}$ and $D \not\models$ **ALWAYS holds** P; thus **caused** P is no longer a semantic consequence. The AGM framework, and similarly the KM framework, is not suitable for non-monotonic settings, as discussed, e.g., for non-monotonic logic programming in [18] and for defeasible logic in [8]. Thus governing our action description updates with the AGM or KM postulates is not meaningful; an AGM- respectively KM-style theory for non-monotonic logics with significant attention is, to our knowledge, still missing. Instead, we considered some basic properties in Section 4 that are analog to properties for non-monotonic logic programming updates [18]. We note that [33], for instance, considers the incorporation of belief change into the fluent calculus, geared by an axiomatic treatment of belief revision and update satisfying the AGM and KM postulates, respectively. However, the underlying logic is monotonic and only static knowledge is subject to change, and preference is based on a ranking of states.

7.3 Repair of Action Descriptions

We can sometimes improve solutions (and near-solutions) to an ADU problem $(D, I, C, \sqsubset_C)$ 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 I is a dynamic law of the form:

$$\textbf{caused } L' \textbf{ after } A' \wedge G',$$

where L' is a literal, G' is a propositional combination of fluents, and A' is an action. We can obtain an action description D^s from D, which describes the same transition diagram as D, by replacing each dynamic law (5) in D_m with:

$$\textbf{caused } L \textbf{ if } F \textbf{ after } H \wedge G',$$
$$\textbf{caused } L \textbf{ if } F \textbf{ after } H \wedge \neg G'.$$

We then have that for each near-solution D' to $(D, I, C, \sqsubset_C)$ there exists some near-solution $D^{s\prime}$ to $(D^s, I, C, \sqsubset_C)$ which contains D' as a subset (in particular, for subset preference $\subset$, each solution to $(D, I, C, \subset)$ gives rise to some solution of $(D^s, I, C, \subset)$); with an (ad-hoc) adaptation of the solution preference $\sqsubset_C$ to $\sqsubset^s_C$, the solutions of $(D, I, C, \sqsubset_C)$ can then be recovered from the ones of $(D^s, I, C, \sqsubset^s_C)$. Therefore, such a replacement method can be useful to prevent "complete removal" of some laws from the given action description. Furthermore, solutions of $(D^s, I, C, \sqsubset^s_C)$ which do not correspond to solutions of the original problem $(D, I, C, \sqsubset_C)$ can be viewed as approximations of solutions for the latter. They might be of particular interest if the original problem has no solution.

Similar methods are also useful for repairing an action description, e.g., if some dynamic laws (5) in the action description have missing formulas in H. In this case, we need to replace such causal laws by some modified statement(s) from a candidate space. Our current framework can be generalized in this direction by changing the candidate solution space for a solution D' from $D_u \subseteq D' \subseteq D_u \cup I$ to a set of action descriptions $cand(D, I)$ such that $D_u \cup I \subseteq D'$ holds for each $D' \in cand(D, I)$; if a modifiable causal law ℓ_i in D gives rise to alternative candidate replacements $cand(\ell_i, I)$, then $cand(D, I) = \{\bigcup_{i=1}^n D_i \mid D_i \in cand(\ell_i, I)\}$ should hold, where $D = \{\ell_1, \ldots, \ell_n\}$.

We note that as for repairing action descriptions, [15] took a slightly different, semantics-oriented view for resolving conflicts between an action description and a set of conditions, in the context of action language $\mathcal{C}$. Conflicts are characterized by means of states and transitions in the transition diagram described by the given action description that violate some given conditions. The goal is to resolve each conflict by modifying the action description, but not necessarily by deleting some causal laws. However, the repair of a single conflict might be achieved by numerous alternative changes to the action description, such that the candidate solution space is very large; furthermore, the repairs of individual conflicts interfere with each other, and might introduce other conflicts. This led the authors of [15] to propose support for the user in terms of query services on an action description and conditions, which provide explanations for certain disorders, rather than an automated repair; a respective tool and methodology for its usage to correct editorial errors in the knowledge representation process (e.g., by typos or omitted formula parts) are described in [19, 20]. An interesting issue for further work is to analyze under which conditions such repairs can be obtained as solutions of an ADU problem in a generalized framework as outlined above.

8 Conclusion

In this paper, we have considered the problem of updating an action description with some new information in the framework of action languages, where meta-level knowledge about the domain in terms of observations and other constraints is respected. To this end, we have introduced a formal notion of action description update which, given an action description D, the new information I (as a set of statements) and some desired conditions C (expressed in an action query language), singles out a solution to the update problem, based on a preference relation $\sqsubset_C$ over action descriptions.

We then studied semantical and computational properties of action updates in this framework, where we presented among other results decomposition results and complexity characterizations of basic decision problems associated with computing solutions, viz. deciding solution existence and solution recognition. We considered in the complexity analysis generic settings as well as particular instances, paying attention to different classes of conditions and preference relations. Furthermore, we presented some algorithms for computing solutions and near-solutions (which approximate solutions), and we discussed our work in the context of the literature.

Several issues remain for further work. Our computational results provide a basis for the realization of concrete implementations to incorporate updates into action descriptions in the action language $\mathcal{C}$, based on top of existing reasoning system like the causal calculator [47] or AD-Query [19], which is an important need for deploying such systems to applications. However, for practical concerns, efficient domain-tailored algorithms will need to be developed.

In connection with this, meaningful fragments of low (polynomial) complexity are of interest; related to this is the study of language fragments that correspond to simpler (less expressive) action languages, such as $\mathcal{A}$ or $\mathcal{B}$ (see [26]). However, several of the intractability results that we established here involved rather simple action descriptions, which suggests that polynomial complexity will have to be achieved by pragmatic constraints rather than logical or structural conditions. On the other hand, also richer, more expressive action languages, such as the language $\mathcal{C}$ with disjunctive causal laws may be studied, the action language $\mathcal{C}+$ [40], or the action language $\mathcal{K}$ [17] (into which the language considered here maps naturally) may be studied.

Further issues are to consider richer forms of conditions (e.g., by generalized action query languages), and to extend the current computational study to further notions of preference relations. For example, to syntax-based preference using cardinality, lexicographic ordering, or formula ranking, possibly with priority levels on top [7, 10], or to semantic-based preference that uses other weight assignments like those in [16] (which are computable in polynomial space) or preference based on state- and transition-rankings, inspired by approaches e.g. in conditional reasoning (see [22]).

Another issue are multiple updates. The update descriptions that we presented here provide a useful basis for a realization of Markovian (history-less) updates $I_1, I_2, \ldots, I_k$ of an action description under lazy evaluation, and may be used, similar as update programs in the context of logic program updates [4, 18], also to realize non-Markovian semantics of a sequence of updates to an action description. However, this remains to be explored in further investigation.

Finally, in regard with connection with AGM and KM theory, postulates and properties that are tailored to theories of action in a non-monotonic setting would be interesting.

Appendix

A Proofs for Section 5

Theorem 4 *Given an action description D and a set C of queries, deciding $D \models C$ is (i)* PSPACE-*complete in general, (ii) Θ^P_{k+3}-complete if k is the maximal nesting depth of dynamic queries in C, and (iii)* $\mathrm{P}^{\mathrm{NP}}_{\|}$*-complete if C does not involve dynamic queries.*

Proof. Concerning *(i)* the result has been shown in [16]. We proceed with the proof of *(ii)* and *(iii)*.
Membership: W.l.o.g. C contains a single query c. Let us consider *(iii)* first. Then, c is a conjunction of clauses over universal queries of the form **ALWAYS** Q or $\neg$**ALWAYS** Q, where Q is a conjunction of clauses over static queries of the form **holds** F or $\neg$**holds** F. Checking truth of a negated universal (sub-)query of this form is in NP. To wit, we nondeterministically guess a possible state s of D and verify in polynomial time that s is a state of D (satisfies all static laws of D) and that s does not satisfy Q (there is a clause in Q such that none of its static queries is satisfied at s). Hence, the complementary task, i.e., checking the truth of a positive universal query, **ALWAYS** Q, is in coNP. Thus, we can decide $D \models c$ in polynomial time with

a single parallel evaluation of n NP-oracle calls, given that n is the number of universal queries in c. This proves $\mathrm{P}_{\|}^{\mathrm{NP}}$-membership.

For *(ii)*, the query c is a conjunction of clauses over universal queries of the form **ALWAYS** Q or $\neg$**ALWAYS** Q, where Q is a conjunction of clauses over static queries as above and over dynamic queries **necessarily** Q_{k-1} **after** $A_1; \ldots ; A_n$ or $\neg$**necessarily** Q_{k-1} **after** $A_1; \ldots ; A_n$, where Q_{k-1} is a basic query of nesting depth $k-1$. Let $c_1 - c_4$ denote queries of the form $c_1 =$ **ALWAYS** Q, $c_2 = \neg$**ALWAYS** Q, $c_3 =$ **ALWAYS** $\neg Q$, and $c_4 = \neg$**ALWAYS** $\neg Q$, respectively. We show by induction that deciding whether $D \models c$ is in Θ_{k+3}^P.

Base case ($k = 0$): For static Q, by *(iii)* deciding $D \models c_i$, is in $\mathrm{P}_{\|}^{\mathrm{NP}}$, for $1 \leq i \leq 4$. Hence, let $Q =$ **necessarily** Q_{k-1} **after** $A_1; \ldots ; A_n$ be a dynamic query. Deciding $D \models c_1$ is in NP since the complementary problem $D \models c_2$ is in coNP. The latter problem is decided by nondeterministically guessing a history $h = s_0, A_1, s_1, \ldots, s_{n-1}, A_n, s_n$ of length n and checking in polynomial time that h is a history of D, i.e., that s_i $(0 \leq i \leq n)$ is a state of D and that $\langle s_i, A_{i+1}, s_{i+1}\rangle$ $(0 \leq i < n)$ is in R. Furthermore, $D, s_n \models \neg Q_{k-1}$ can be checked in polynomial time since Q_{k-1} is a propositional combination of static queries, witnessing $D \not\models c_1$. Deciding $D \models c_3$ is in Π_2^P and the complementary problem $D \models c_4$ is in Σ_2^P. To wit, in order to disprove $D \models c_3$, guess a state s and—as outlined above—use the NP-oracle to verify that for all histories h of length n emanating from s $(s_0 = s)$ it holds that $D, s_n \models Q_{k-1}$. This establishes $D, s \not\models \neg Q$ and hence, $D \not\models c_3$. Putting all together, in order to decide $D \models c$, an oracle for Σ_2^P problems is sufficient to decide the truth of any universal query in c. Thus, $D \models c$ can be checked in polynomial time with a polynomial number of parallel Σ_2^P-oracle calls and therefore is in Θ_3^P.

Induction step: Let the nesting depth of dynamic queries be $k > 0$, and assume that deciding $D \models Q_{k-1}$ is in Θ_{k+2}^P for any subquery of nesting depth $k-1$. Then, as easily seen by the arguments for the base case above, $D \models I$ can be decided by means of a Σ_{k+2}^P-oracle for any universal query $Q \in c$. Thus, again by parallel evaluation, $D \models c$ is in Θ_{k+3}^P.

Hardness: In order to prove *(iii)* we reduce the problem to the following $\mathrm{P}_{\|}^{\mathrm{NP}}$-hard decision version of *Maximum CNF Satisfiability*: Given a Boolean formula F in *conjunctive normal form (CNF)* and an integer k, decide whether the maximum number of clauses in F that can be simultaneously satisfied by an interpretation is $0 \bmod k$.

W.l.o.g., let F be a 3-CNF formula of the form $\bigwedge_{i=1}^{n} L_{i,1} \vee L_{i,2} \vee L_{i,3}$, where $L_{i,j}$, $1 \leq i \leq n$, $1 \leq j \leq 3$, is a literal over atoms $X = \{X_1, \ldots, X_m\}$. For $X_i \in X$, by $\neg L$ we denote $\neg X_i$ if $L = X_i$ and X_i if $L = \neg X_i$. Consider the action description D_1 consisting of:

$$\left.\begin{array}{l}\textbf{caused } C_i \textbf{ if } L_{i,1}, \quad \textbf{caused } C_i \textbf{ if } L_{i,2}, \quad \textbf{caused } C_i \textbf{ if } L_{i,3},\\ \textbf{caused } \neg C_i \textbf{ if } \neg L_{i,1} \wedge \neg L_{i,2} \wedge \neg L_{i,3},\end{array}\right\} 1 \leq i \leq n$$

$$\begin{array}{ll}\textbf{caused } F_{1,1} \textbf{ if } C_1, & \textbf{caused } \neg F_{1,1} \textbf{ if } \neg C_1,\\ \textbf{caused } F_{1,0} \textbf{ if } \neg C_1, & \textbf{caused } \neg F_{1,0} \textbf{ if } C_1,\end{array}$$

$$\left.\begin{array}{l}\textbf{caused } F_{i,j} \textbf{ if } C_i \wedge F_{i-1,j-1},\\ \textbf{caused } \neg F_{i,j} \textbf{ if } \neg C_i \wedge F_{i-1,j-1},\end{array}\right\} 2 \leq i \leq n,\ 1 \leq j \leq i$$

$$\left.\begin{array}{l}\textbf{caused } F_{i,j} \textbf{ if } \neg C_i \wedge F_{i-1,j},\\ \textbf{caused } \neg F_{i,j} \textbf{ if } \quad C_i \wedge F_{i-1,j} \quad ,\end{array}\right\} 2 \leq i \leq n,\ 0 \leq j < i$$

Observe that D_1 contains only static laws. A state, s, consistent with D_1 corresponds to an arbitrary total interpretation on X together with a total interpretation on fluents C_i, $1 \leq i \leq n$, such that C_i is true at s iff the interpretation on X satisfies clause C_i. The latter is enforced by the first $4n$ laws in D_1. The remaining laws cause a total interpretation on fluents $F_{i,j}$, $1 \leq j \leq i \leq n$, such that $F_{i,j}$ is true at s iff the interpretation on X satisfies j clauses among $\{C_1, \ldots, C_i\}$.

Now consider the following query:

$$\begin{aligned} c_k \;=\; & \textbf{ALWAYS holds } F_{n,0} \vee \\ & \textbf{SOMETIMES holds } F_{n,k} \wedge \textbf{ALWAYS } (\neg\textbf{holds } F_{n,k+1} \wedge \ldots \wedge \neg\textbf{holds } F_{n,n}) \vee \\ & \ldots \\ & \textbf{SOMETIMES holds } F_{n,lk} \wedge \textbf{ALWAYS } (\neg\textbf{holds } F_{n,lk+1} \wedge \ldots \wedge \neg\textbf{holds } F_{n,n}), \end{aligned}$$

where $l = \lfloor n/k \rfloor$.

We show that the maximum number of clauses in F that can be simultaneously satisfied by an interpretation is $0 \bmod k$ iff $D_1 \models c_k$.

Only-If: Suppose that the maximum number o of clauses in F that can be simultaneously satisfied by an interpretation is $0 \bmod k$. Consider $o = 0$ first. Then, no clause of F is satisfiable. By construction, $F_{i,0}$ holds for $1 \leq i \leq n$ at every state s of D_1. In particular, $F_{n,0}$ holds at every state, and therefore **ALWAYS holds** $F_{n,0}$ is entailed by D_1, i.e., $D_1 \models c_k$. Now let $o > 0$. W.l.o.g. $o = ak$ for some $1 \leq a \leq l$. Then, by construction $F_{n,j}$ is false for $o < j \leq n$ at every state s of D_1. Therefore, $D_1 \models$ **ALWAYS** $(\neg$**holds** $F_{n,ak+1} \wedge \ldots \wedge \neg$**holds** $F_{n,n})$. Also by construction, $F_{n,o}$ is true at a state corresponding to an assignment that maximizes the simultaneously satisfied clauses. This implies $D_1 \models$ **SOMETIMES holds** $F_{n,ak}$. Observing that, together, these two queries constitute a conjunct of c_k, we conclude that $D_1 \models c_k$.

If: Suppose $D_1 \models c_k$, and assume $D_1 \models$ **ALWAYS holds** $F_{n,0}$ first. Then, by construction no clause in F is satisfiable, Hence the maximum number o of clauses in F that can be simultaneously satisfied by an interpretation is 0 and thus $o \equiv 0 \bmod k$. Now let any other conjunct of c_k be entailed by D_1, i.e., for some $1 \leq a \leq l$ it holds that $D_1 \models$ **SOMETIMES holds** $F_{n,ak}$ and $D_1 \models$ **ALWAYS** $(\neg$**holds** $F_{n,ak+1} \wedge \ldots \wedge \neg$**holds** $F_{n,n})$. Then, there is a state s at which $F_{n,ak}$ is true. By construction, this means that ak clauses of F can simultaneously be satisfied. Moreover, $F_{n,j}$ is false at every state s of D_1 if $j > ak$. Again by construction, this implies that ak is the maximum number of clauses in F that can be simultaneously satisfied. Since $ak \equiv 0 \bmod k$ this proves the claim.

For hardness in Case *(ii)*, consider m quantified Boolean formulas of form

$$\Phi_l = Q_1 X_1^l \, Q_2 \, X_2^l \cdots Q_n X_n^l \, E^l, \qquad 1 \leq l \leq m,$$

where $Q_i = \exists$ if $i \equiv 1 \bmod 2$ and $Q_i = \forall$ otherwise, X_i^k and X_j^l, $1 \leq i, j \leq n$ and $1 \leq k, l \leq m$, are pairwise disjunct sets of propositional variables if $i \neq j$ or $k \neq l$. and E^l is Boolean formula over atoms in $X^l = X_1^l \cup \cdots \cup X_n^l$, such that if Φ_l is false then $\Phi_{l+1}, \ldots, \Phi_m$ are false, too. Deciding whether the maximum index o, $1 \leq o \leq m$, such that Φ_o is true, is odd is Θ_{n+1}^P-hard.

We reduce the problem of deciding $D \models c$ for a query c with nesting depth k of dynamic queries to this problem, as follows.

Let $n = k + 2$, $1 \leq l \leq m$, and let D_2 be the action description consisting of the statements:

$$\left.\begin{array}{l} \textbf{caused } F_i^l \textbf{ if } F_i^l \textbf{ after } A_{i-1}, \\ \textbf{caused } \neg F_i^l \textbf{ if } \neg F_i^l \textbf{ after } A_{i-1}, \end{array}\right\} 2 \leq i \leq n, F_i^l \in X_i^l$$

$$\left.\begin{array}{l} \textbf{caused } F_j^l \textbf{ after } A_{i-1} \wedge F_j^l, \\ \textbf{caused } \neg F_j^l \textbf{ after } A_{i-1} \wedge \neg F_j^l, \end{array}\right\} 2 \leq i \leq n, 1 \leq j \leq n, i \neq j, F_j^l \in X_j^l$$

Observe that a state s of D_2 corresponds to an arbitrary consistent total interpretation over $X^1 \cup \cdots \cup X^m$. Note also that $\langle s, \{A_i\}, s'\rangle$ $(1 \leq i \leq n-1)$ is a transition in the transition diagram described by D_2 iff all fluents are interpreted identically except those over $X^1_{i+1} \cup \cdots \cup X^m_{i+1}$.

Consider the query:

$$c_o = \begin{cases} \bigvee_{l=0}^{(m-3)/2}(\textbf{SOMETIMES } f^{2l+1} \wedge \textbf{ALWAYS } \neg f^{2l+2}) \vee f^m & \text{if } m \text{ is odd,} \\ \bigvee_{l=0}^{(m-2)/2}(\textbf{SOMETIMES } f^{2l+1} \wedge \textbf{ALWAYS } \neg f^{2l+2}) & \text{otherwise,} \end{cases}$$

where

$$f^l = p_1 \textbf{ necessarily}, p_1 (\ldots (p_{n-1} \textbf{ necessarily}\, p_{n-1} \textbf{ holds } E^l \textbf{ after } \{A_{n-1}\}) \ldots) \textbf{ after } \{A_1\},$$

where $p_i = \neg$ if i is even, void otherwise, for $1 \leq i \leq n-1$.

We first prove that Φ_l is true iff there exists a state s of D_2, such that $D_2, s \models f^l$.

For the only-if direction suppose Φ_l is true. We show by a recursive argument that if a state s_0 coincides with a satisfying truth assignment for Φ_l on X^l_1 then $D_2, s_0 \models f^l$. Assume that s_{n-2} is a state of D_2 that coincides with a satisfying truth assignment for Φ_l on $X^l_1 \cup \cdots \cup X^l_{n-1}$. We show that $D_2, s_{n-2} \models p_{n-1} N\ p_{n-1}$ **holds** E^l **after** $\{A_{n-1}\}$. If $n-1$ is odd then $Q_n = \forall$. Thus, any assignment on X^l_n will turn the assignment on $X^l_1 \cup \cdots \cup X^l_{n-1}$ given by s_{n-2} into a satisfying assignment for E^l. Thus, every transition by $\{A_{n-1}\}$ from s_{n-2} will lead to a state s_{n-1} that satisfies E^l. This proves $D_2, s_{n-2} \models$ **necessarily holds** E^l **after** A_{n-1} if $n-1$ is odd. So let $n-1$ be even. Then $Q_n = \exists$. In this case, there exists an assignment on X^l_n that, together with the assignment on $X^l_1 \cup \cdots \cup X^l_{n-1}$ given by s_{n-2}, is a satisfying assignment for E^l. Thus, there is a transition by $\{A_{n-1}\}$ from s_{n-2} to a state s_{n-1} that satisfies E^l. Therefore, $D_2, s_{n-2} \models \neg$**necessarily** $\neg$**holds** E^l **after** A_{n-1} if $n-1$ is even. In any case, $D_2, s_{n-2} \models p_{n-1} N$ **holds** $p_{n-1} E^l$ **after** $\{A_{n-1}\}$. Applying this argument recursively proves the claim that if a state s_0 coincides with a satisfying truth assignment for Φ_l on X^l_1, then $D_2, s_0 \models f^l$, and thus, that there exists a state of D_2 such that $D_2, s \models f^l$.

For the if-direction let s be a state of D_2, such that $D_2, s \models f^l$. We establish the truth of Φ_l recursively as follows. Let $h = s, A_1, s_1, \ldots, s_{n-3}A_{n-2}, s_{n-2}$ be a history of D_2. We show that s_{n-2} is a state of D_2 that coincides with a truth assignment on $X^l_1 \cup \cdots \cup X^l_{n-1}$, such that $Q_n\, E^l$ is true. If $n-1$ is odd, then $D_2, s_{n-2} \models$ **necessarily holds** E^l **after** A_{n-1}, since $D_2, s \models f^l$. Thus, any assignment on X^l_n will turn the assignment on $X^l_1 \cup \cdots \cup X^l_{n-1}$ given by s_{n-2} into a satisfying assignment for E^l. If $n-1$ is even, then $D_2, s_{n-2} \models \neg$**necessarily** $\neg$**holds** E^l **after** A_{n-1}, since $D_2, s \models f^l$. Therefore, there exists an assignment on X^l_n that will turn the assignment on $X^l_1 \cup \cdots \cup X^l_{n-1}$ given by s_{n-2} into a satisfying assignment for E^l. Hence, in any case $Q_n\, E^l$ is true. Applying this argument recursively proves the claim that $D_2, s \models f^l$ implies the truth of Φ_l.

We now show that the maximum index o such that Φ_o is true, is odd iff $D_2 \models c_o$.

Only-If: Let the maximum index o such that Φ_o is true be odd. Consider any state s of D_2 such that $D_2, s \models f^o$. If $o = m$ this proves $D_2 \models c_o$. So let $o < m$. Then additionally $D_2, s \not\models f^{o+1}$, for every state s' of D_2. Hence, $D_2 \models$ **SOMETIMES** f^o and $D_2 \models$ **ALWAYS** $\neg f^{o+1}$, i.e., for $l = (o-1)/2$ $D_2 \models$ **SOMETIMES** $f^{2l+1} \wedge$ **ALWAYS** $\neg f^{2l+2}$. This proves $D_2 \models c_o$.

If: Assume $D_2 \models c_o$. If m is odd and $D_2 \models f^m$, Then m is the maximum index o such that Φ_o is true, and o is odd. This proves the claim. So consider the remaining cases, i.e., there is an index l $(0 \leq l \leq (m-3)/2$ if m is odd and $0 \leq l \leq (m-2)/2$, otherwise), such that $D_2 \models$ **SOMETIMES** $f^{2l+1} \wedge$ **ALWAYS** $\neg f^{2l+2}$. Then, there is a state s of D_2 such that f^{2l+1} is entailed, whereas f^{2l+2} is not entailed at any state s' of D_2. Let $o = 2l + 1$. We conclude that Φ_o is true and Φ_{o+1} is false. Thus, o is the maximum index such that Φ_o is true, and it is odd. This proves the claim and therefore Θ^P_{n+1}-hardness, i.e., Θ^P_{k+3}-hardness. □

Theorem 5 *Deciding whether a given ADU problem* $(D, I, C, \sqsubset_C)$ *has a solution (or a near-solution) is (i)* PSPACE*-complete in general, (ii)* Σ^P_{k+3}*-complete, if* k *is the maximal nesting depth of dynamic queries in* C_o*, (iii)* Σ^P_2*-complete, if* C_o *does not involve dynamic queries, and (iv)* NP*-complete if* $C_o = \emptyset$*.*

Proof.
Membership: Follows from Theorems 3 and 4.
Hardness: Hardness in Case *(i)* follows from Theorem 4. For *(ii)* let $n = k + 2$ and let

$$\Phi = \exists Y\, Q_1 X_1 \cdots Q_n X_n\, E$$

be a QBF, where $Q_i = \exists$ if $i \equiv 0 \bmod 2$ and $Q_i = \forall$ otherwise. Consider

$$D_u = D_2 \cup \{\textbf{caused}\; Y_i\; \textbf{after}\; A_{i-1} \wedge Y_i, \textbf{caused}\; \neg Y_i\; \textbf{after}\; A_{i-1} \wedge \neg Y_i \mid 2 \leq i \leq n\},$$

where D_2 is the action description from the proof of Theorem 4 with $l = 1$, $D_m = \{\textbf{caused}\; Y_i, \textbf{caused}\; \neg Y_i \mid Y_i \in Y\}$, $I = \emptyset$, $C = C_o \cup C_p$ with $C_p = \emptyset$ and

$$C_o = \{\textbf{ALWAYS}\;\; p_1\;\; \textbf{necessarily}\, p_1(\ldots(p_{n-1}\;\; \textbf{necessarily}\, p_{n-1}\textbf{holds}\, E\;\; \textbf{after}\; \{A_{n-1}\})\ldots)\; \textbf{after}\; \{A_1\}\},$$

where $p_i = \neg$ if i is odd, void otherwise, for $1 \leq i \leq n - 1$. We show that there exists a solution to the action description update problem $(D_u \cup D_m, I, C, \sqsubset_C)$ iff Φ is true.

For the only-if direction, let $D_u \subseteq D' \subseteq D_u \cup D_m$ be a solution. Then D' is consistent and states of D' coincide with some interpretation on Y and an arbitrary interpretation on $X_1, \ldots, X_n$. By the same arguments as in the hardness proof of Theorem 4 *(ii)*, the fact that $D' \models C_o$ witnesses the truth of Φ.

For the if-direction let Φ be true. Consider a satisfying truth assignment on Y, let D'_m be the set of static causal laws from D_m compliant with this assignment, and let $D' = D_u \cup D'_m$. Then, D' is consistent and $D_u \subseteq D' \subseteq D_u \cup D_m$. Moreover, by the same arguments as in the hardness proof of Theorem 4 *(ii)*, $D' \models C_o$. This proves that D' is a near-solution, and hence the existence of a solution.

For *(iii)* let $\Phi = \exists Y \forall X\, E$ and consider the action description update problem $(D_u \cup D_m, I, C, \sqsubset_C)$, where $D_u = \emptyset$, $D_m = \{\textbf{caused}\; Y_i, \textbf{caused}\; \neg Y_i \mid Y_i \in Y\}$, $I = \emptyset$, and $C = C_o = \{\textbf{ALWAYS holds}\; E\}$. We prove that the action description update problem $(D_u \cup D_m, I, C, \sqsubset_C)$ has a solution iff Φ is true.

For the only-if direction, let $D_u \subseteq D' \subseteq D_m$ be a solution. Then D' is consistent and states of D' coincide with some interpretation on Y and an arbitrary interpretation on X. Since $D' \models C_o$, E is true at every such state, witnessing that any truth assignment on X turns the joint assignment on both, Y and X, into a satisfying assignment for E. This proves the truth of Φ.

For the if-direction let Φ be true. Consider a satisfying truth assignment on Y, and let D' be the set of static causal laws from D_m compliant with this assignment. Then, D' is consistent and $D_u \subseteq D' \subseteq D_m$. Moreover, since Φ is true, any truth assignment on X turns the joint assignment on both, Y and X, into a satisfying assignment for E. Therefore, E holds at all states of D', witnessing $D' \models C_o$. This proves that D' is a near-solution, and hence the existence of a solution.

Finally. for *(iv)*, let E be a Boolean formula over atoms Y and let us define $D_u = \{\textbf{caused}\; Y_1\; \textbf{if}\; \neg E$, $\textbf{caused}\; \neg Y_1\; \textbf{if}\; \neg E\}$, $D_m = \{\textbf{caused}\; Y_i, \textbf{caused}\; \neg Y_i \mid Y_i \in Y\}$, $I = \emptyset$, and $C = \emptyset$. Then, $(D_u \cup D_m, I, C, \sqsubset_C)$ has a solution iff E is satisfiable.

For the only-if direction, let $D_u \subseteq D' \subseteq D_u \cup D_m$ be a solution. Then D' is consistent and states of D' coincide with some interpretation on Y. Since $D_u \subseteq D'$, E is true at every such state. This proves the satisfiability of E.

For the if-direction let E be satisfiable. Consider a satisfying truth assignment on Y, and let D'_m be the set of static causal laws from D_m compliant with this assignment. Then, $D' = D_u \cup D'_m$ is consistent and $D_u \subseteq D' \subseteq D_u \cup D_m$. Moreover $D' \models C_o$ trivially. This proves that D' is a near-solution, and hence the existence of a solution. $\Box$

Theorem 6 *Given an ADU problem $(D, I, C, \subset)$ and an action description D', deciding whether D' is a solution for it is (i)* PSPACE*-complete for general queries in C_o, (ii) Π^P_{k+3}-complete if k is the maximal nesting depth of dynamic queries in C_o, (iii) Π^P_2-complete if C_o does not involve dynamic queries, and (iv)* D^P*-complete if $C_o = \emptyset$.*

Proof.

Membership: Follows from Theorem 3, observing that for any given action descriptions D' and D'', deciding $D' \subset D''$ can be done in polynomial time, i.e., that `Pcheck` is in P for $\subset$.

Hardness: Hardness in Case *(i)* follows from Theorem 4. For *(ii)* let $n = k + 2$ and let

$$\Phi = \forall Y\, Q_1 X_1 \,\cdots\, Q_n X_n\, E$$

be a QBF, where $Q_i = \exists$ if $i \equiv 1 \bmod 2$ and $Q_i = \forall$ otherwise. Consider

$$D_u = D_2 \cup \{\textbf{caused}\ Y_i\ \textbf{after}\ A_{i-1} \wedge Y_i, \textbf{caused}\ \neg Y_i\ \textbf{after}\ A_{i-1} \wedge \neg Y_i \mid 2 \leq i \leq n\},$$

where D_2 is the action description from the proof of Theorem 4 with $l = 1$, $D_m = \{\textbf{caused}\ Y_i, \textbf{caused}\ \neg Y_i \mid Y_i \in Y\}$, $I = \emptyset$, and $C = C_o = \{\textbf{ALWAYS}\ \ f \vee g\}$, where

$$\begin{aligned} f &= p_1\ \textbf{necessarily}\, p_1\, (\ldots (p_{n-1}\ \textbf{necessarily}\, \bar{p}_{n-1}\ \textbf{holds}\ E\ \ \textbf{after}\ \{A_{n-1}\}) \ldots)\ \textbf{after}\ \{A_1\}, \\ g &= \bigwedge_{Y_i \in Y} \textbf{SOMETIMES}\ \ \textbf{holds}\ Y_i \wedge \textbf{SOMETIMES}\ \ \textbf{holds}\ \neg Y_i, \end{aligned}$$

where $p_i = \neg$ if i is odd, void otherwise, for $1 \leq i \leq n-1$, and $\bar{p}_n - 1 = \neg$ if n is odd and void otherwise. We show that D_u is a solution to the action description update problem $(D_u \cup D_m, I, C, \subset)$ iff Φ is true.

Obviously, D_u is consistent and $I \subseteq D_u$. Additionally, states of D_u consist of arbitrary truth assignments to Y and $X_1, \ldots, X_n$. Therefore, D_u satisfies g, and hence $D_u \models C_o$. This proves that D_u is a near-solution. We show that it is a maximum near-solution iff Φ is true.

For the only-if direction, towards a contradiction assume that Φ is false. Then $\neg\Phi$ is true. Observe that $\neg\Phi$ is a QBF of the form considered in the hardness proof of Theorem 5 *(ii)* with E negated. Applying the arguments of this proof, we obtain that there exists a set $D_u \subset D' \subseteq D_u \cup D_m$, such that D' is consistent and $D', s \models f$ for every state s of D' (Note that $\bar{p}_{n-1}$ accounts for the negation of E). Therefore $D' \models C_o$, and thus D' is a near-solution. This contradicts the maximality of D_u.

For the if-direction, towards a contradiction assume that D_u is not maximal. Then, all sates of a maximum solution coincide on at least one assignment to some $Y_i \in Y$, and therefore it does not satisfy g. Consequently, f is satisfied at all states of a maximum solution. Applying the arguments of the hardness proof of Theorem 5 *(ii)*, we conclude that $\neg\Phi$ is true, a contradiction.

For *(iii)* let $\Phi = \forall Y\, \exists X\, E$ and consider the action description update problem $(D_u \cup D_m, I, C, \subset)$, where $D_u = \emptyset$, $D_m = \{\textbf{caused}\ Y_i, \textbf{caused}\ \neg Y_i \mid Y_i \in Y\}$, $I = \emptyset$, and $C = C_o = \{\textbf{ALWAYS}\ \neg\textbf{holds}\ E \vee g\}$, with g as before. We prove that the action description update problem $(D_u \cup D_m, I, C, \subset)$ has $D_u = \emptyset$ as a solution iff Φ is true.

Obviously, D_u is consistent and $I \subseteq D_u$. Additionally, states of D_u consist of arbitrary truth assignments to Y and X. Therefore, D_u satisfies g, and hence $D_u \models C_o$. This proves that D_u is a near-solution. We show that it is a maximum near-solution iff Φ is true.

For the only-if direction, towards a contradiction assume that Φ is false. Then $\neg\Phi$ is true. Observe that $\neg\Phi$ is a QBF of the form considered in the hardness proof of Theorem 5 *(iii)* with E negated. Applying the arguments of this proof, we obtain that there exists a set $D_u \subset D' \subseteq D_m$, such that D' is consistent and $D' \models$ **ALWAYS** $\neg$**holds** E, i.e., $D' \models C_o$. Therefore, D' is a near-solution, which contradicts the maximality of D_u.

For the if-direction, towards a contradiction assume that D_u is not maximal. Then, all sates of a maximum solution coincide on at least one assignment to some $Y_i \in Y$, and therefore it does not satisfy g. Consequently, a maximum solution must satisfy **ALWAYS** $\neg$**holds** E . Applying the arguments of the hardness proof of Theorem 5 *(iii)*, we conclude that $\neg\Phi$ is true, a contradiction.

Finally *(iv)*, let E_1 and E_2 be Boolean formulas over atoms Y_1 and Y_2, respectively. Consider $D_u = \{\textbf{caused}\ \neg F, \textbf{caused}\ F\ \textbf{if}\ \neg E_1\}$, $D_m = \{\textbf{caused}\ F\ \textbf{if}\ \neg E_2\}$, $I = \emptyset$, and $C = \emptyset$. Then, $(D_u \cup D_m, I, C, \subset)$ has solution D_u iff E_1 is satisfiable and E_2 is unsatisfiable.

Obviously, $I \subseteq D_u$, and $D_u \models C_o$. Therefore, D_u is a solution iff it is consistent and maximal, i.e., no superset of D_u is consistent. We show that this two conditions hold iff E_1 is satisfiable and E_2 is unsatisfiable.

For the only-if direction, assume that D_u is consistent and maximal. Then E_1 is satisfiable witnessed by the truth assignment to Y_1 of any state of D_u. Furthermore, $D_u \cup D_m$ is inconsistent (otherwise it would be a solution, since it trivially satisfies C_o), which implies that E_2 is unsatisfiable.

For the if-direction, let E_1 be satisfiable and E_2 be unsatisfiable. Then any satisfying assignment to fluents in Y_1 together with assigning falsity to F and any truth assignment to fluents from Y_2 yields a state of D_u witnessing its consistency. Moreover, $D_u \cup D_m$ is inconsistent due to the unsatisfiability of E_2, which implies that D_u is maximal. This proves D^P-hardness. □

Theorem 7 *Given an ADU problem $(D, I, C, <_{weight_q})$ and an action description D', deciding whether D' is a solution for it is (i)* PSPACE*-complete for general queries in C, (ii) Π^P_{k+3}-complete if k is the maximal nesting depth of dynamic queries in C, (iii) Π^P_2-complete if C does not involve dynamic queries, and (iv)* NP*-complete if $C = \emptyset$.*

Proof.
Membership: For *(i)*, *(ii)*, and *(iii)* membership follows from Theorems 3 and 4. Note that in order to decide $D_1 <_{weight_q} D_2$ for any action descriptions D_1 and D_2, such that $D_u \cup I \subseteq D_i \subseteq D \cup I$ for $i \in \{1, 2\}$, and a set of weighted queries C_p, we decide $D_i \models c$, for every $c \in C_p$ (i.e., polynomially many), and sum up the corresponding weights in polynomial time. Thus, if $D_i \models c$ can be decided in polynomial space, respectively in polynomial time with the help of a Σ^P_{i-1}-oracle, then `Pcheck` is in PSPACE, respectively in Δ^P_i, for $<_{weight_q}$. For *(iv)*, i.e. $C = \emptyset$, `Pcheck` is trivial for $<_{weight_q}$. In this case we can decide whether D' is a solution by guessing a state s and checking that it is a state of D' in polynomial time (witnessing consistency) and additionally checking $D_u \cup I \subseteq D'$ and $D' \subseteq D \cup I$ in polynomial time. This proves NP-membership for *(iv)*.
Hardness: Hardness in Case *(i)* follows easily from Theorem 4. For *(ii)* let $n = k + 2$ and consider Φ, D_u, D_m, I, and C_o from the proof of Theorem 6 *(ii)*. Additionally, let

$$C_p = \{\textbf{ALWAYS holds}\ Y_i, \textbf{ALWAYS holds}\ \neg Y_i \mid Y_i \in Y\}$$

and consider a weight of 1 for each $c \in C_p$. Then, $D_u <_{weight_q} D'$ for every $D_u \subset D' \subseteq D_u \cup D_m$, since $weight_q(D_u) = 0$, whereas all states of D' coincide on at least one assignment to some $Y_i \in Y$, thus making at least one of the queries in C_p true, i.e., $weight_q(D') \geq 1$. Therefore, D_u is a solution to $(D_u \cup D_m, I, C_o \cup C_p, <_{weight_q})$ iff it is a solution to $(D_u \cup D_m, I, C_o, \subset)$, which proves Π^P_{k+3}-hardness (cf. Theorem 6 *(ii)*).

For *(iii)* consider Φ, D, I, and C_o from the proof of Theorem 6 *(ii)*. Again, let

$$C_p = \{\textbf{ALWAYS holds } Y_i, \textbf{ALWAYS holds } \neg Y_i \mid Y_i \in Y\}$$

with weight 1 for each $c \in C_p$. Then, for the same reason as above, $D_u <_{weight_q} D'$ for every $D_u \subset D' \subseteq D_u \cup D_m$. Therefore, D_u is a solution to $(D_u \cup D_m, I, C_o \cup C_p, <_{weight_q})$ iff it is a solution to $(D_u \cup D_m, I, C_o, \subset)$, proving Π^P_2-hardness.

Finally *(iv)*, let E be a Boolean formula over atoms Y and consider the ADU problem given by $D_u = \{\textbf{caused } Y_1 \textbf{ if } \neg E, \textbf{ caused } \neg Y_1 \textbf{ if } \neg E\}$, $D_m = \emptyset$, $I = \emptyset$, and $C = \emptyset$. Then, D_u is a solution to $(D_u \cup D_m, I, C, <_{weight_q})$ iff E is satisfiable.

For the only-if direction, let D_u be a solution. Then D_u is consistent, states of D_u coincide with some interpretation on Y, and E is true at every such state. This proves the satisfiability of E.

For the if-direction let E be satisfiable. A satisfying truth assignment on Y is a state of D_u, i.e., D_u is consistent. Moreover, $D_u \cup I \subseteq D_u \subseteq D \cup I$ and $D_u \models C_o$ trivially. And since $D_u \cup I = D_u = D \cup I$, we conclude that D_u is a solution. □

B Proofs for Section 6

Prior to the proof of Proposition 2, we establish the following lemma which pinpoints the relation between states and transitions of an update description U and any action description D' obtained by an (arbitrary) selection of modifiable laws.

Lemma 2 *Let $D = D_u \cup D_m$ be an action description, and let D'_m be a subset of D_m. Let $\langle S, V, R\rangle$ be the transition diagram described by $D' = D_u \cup D'_m$. Let $U = U(D)$ be the update description of D, with a set $\mathbf{H}$ of update fluents, and let $\langle S^U, V^U, R^U\rangle$ be the transition diagram described by U. Let $\mathbf{M}$ be the subset of $\mathbf{H}$ labeling the laws in D'_m. Then the following hold:*

(i) $s \setminus \mathbf{H} \in S$ *iff* $s \in S^U$ *and* $s \cap \mathbf{H} = \mathbf{M}$,

(ii) $\langle s, A, s'\rangle$ *in* R^U *iff* $s =_{\mathbf{H}} s'$, *and*

(iii) $\langle s \setminus \mathbf{H}, A, s' \setminus \mathbf{H}\rangle \in R$ *iff* $\langle s, A, s'\rangle \in R^U$ *and* $s \cap \mathbf{H} = \mathbf{M}$.

Proof.

(i) For the only-if direction consider any state $s \in S$. By the definition of a transition diagram described by an action description, for every static law (4) in D', s satisfies $G \supset L$.

Case 1. Take any static law (4) in U, that does not contain any $H_i \in \mathbf{H}$. By the definition of an update description, this static law is in D_u as well. Then, since s satisfies $G \supset L$, $s \cup \mathbf{M}$ satisfies $G \supset L$.

Case 2. Take any static law (21) in U such that $H_i \in \mathbf{M}$. By the definition of an update description, there is a corresponding static law (4) in D'_m. Then, since s satisfies $G \supset L$, $s \cup \mathbf{M}$ satisfies $G \wedge H_i \supset L$.

Case 3. Take any static law (21) in U such that $H_i \not\in \mathbf{M}$. Since $s \cup \mathbf{M}$ does not satisfy $G \wedge H_i$, $s \cup \mathbf{M}$ satisfies $G \wedge H_i \supset L$.

By the definition of an update description, U does not contain any other static laws. Therefore, from these three cases, it follows that $s \cup \mathbf{M}$ is a state in S^U.

For the if-direction consider any state s in S^U, such that $s \cap \mathbf{H} = \mathbf{M}$. By the definition of a transition diagram described by an action description, for every static law (4) in U, s satisfies $G \supset L$.

Case 1. Take any static law (4) in D_u. By the definition of an update description it is also in U, and it does not contain any element of $\mathbf{H}$. Therefore, $s \setminus \mathbf{H}$ satisfies $G \supset L$.

Case 2. Take any static law (4) in D'_m. By the definition of an update description, for every static law (4) in D'_m, there is a static law (21) in U. Since, for every corresponding static law (21) in U, s satisfies $G \wedge H_i \supset L$, and since by assumption H_i is in s, $s \setminus \mathbf{H}$ satisfies $G \supset L$.

From these two cases, it follows that, for every static law (4) in D', $s \setminus \mathbf{H}$ satisfies $G \supset L$. Thus, $s \setminus \mathbf{H}$ is in S.

(ii) Since no element of $\mathbf{H}$ appears in the head of any causal law in U except for the inertia laws (23), we conclude that $\langle s, A, s' \rangle$ in R^U iff $s =_{\mathbf{H}} s'$.

(iii) For the only-if direction consider any $\langle s, A, s' \rangle$ in R. By the definition of a transition diagram described by an action description, for every dynamic law (5) in D', s' satisfies L if the law is applicable to $\langle s, A, s' \rangle$ (i.e., $s \cup A$ satisfies H and s' satisfies G). Due to *(i)*, both $s \cup \mathbf{M}$ and $s' \cup \mathbf{M}$ are in S^U.

Case 1. Consider any dynamic law (5) in U, that does not contain any $H_i \in \mathbf{H}$. Suppose that it is applicable to $\langle s \cup \mathbf{M}, A, s' \cup \mathbf{M} \rangle$. Then, since no $H_i \in \mathbf{H}$ occurs in this law, it is applicable to $\langle s, A, s' \rangle$ as well. By the definition of an update description, this law is in D_u. Since $\langle s, A, s' \rangle$ is in R, s' satisfies L. Then $s' \cup \mathbf{M}$ satisfies L.

Case 2. Consider any dynamic law (22) in U, that is not of the form (23), where H_i labels a dynamic law (5) in D'_m, i.e., $H_i \in \mathbf{M}$. Suppose that it is applicable to $\langle s \cup \mathbf{M}, A, s' \cup \mathbf{M} \rangle$. That is, $s \cup \mathbf{M} \cup A$ satisfies $H \wedge H_i$ and $s' \cup \mathbf{M}$ satisfies G. Since H does not contain any $H_i \in \mathbf{H}$, $s \cup A$ satisfies H; since G does not contain any $H_i \in \mathbf{H}$, s' satisfies G. Then, the corresponding dynamic law (5) in D'_m is applicable to $\langle s, A, s' \rangle$. Since $\langle s, A, s' \rangle$ is in R, s' satisfies L. Then, $s' \cup \mathbf{M}$ satisfies L.

Case 3. Consider any dynamic law (23) in U. By (*ii*) we conclude that $\langle s, A, s' \rangle$ in R^U iff $s =_{\mathbf{H}} s'$. Hence, $s \cup \mathbf{M}$ satisfies H_i iff $s' \cup \mathbf{M}$ satisfies H_i. Therefore, this law is applicable to $\langle s \cup \mathbf{M}, A, s' \cup \mathbf{M} \rangle$ iff $L = H_i$ and H_i is in $\mathbf{M}$, or $L = \neg H_i$ and $H_i \notin \mathbf{M}$. Consequently, $\mathbf{M}$ is the only interpretation on $\mathbf{H}$ satisfying the heads of the applicable inertia laws.

By the definition of an update description, U does not contain any other dynamic laws applicable to $\langle s \cup \mathbf{M}, A, s' \cup \mathbf{M} \rangle$.

So far we have shown that, *(a)* for every $\langle s, A, s' \rangle$ in R, $s' \cup \mathbf{M}$ satisfies the heads of every dynamic law in U that is applicable to $\langle s \cup \mathbf{M}, A, s' \cup \mathbf{M} \rangle$. Moreover, we can observe that *(b)* for each dynamic law in D' applicable to $\langle s, A, s' \rangle$, there is a corresponding law in U applicable to $\langle s \cup \mathbf{M}, A, s' \cup \mathbf{M} \rangle$, and that *(c)* except for the inertia laws (23), U does not contain any other dynamic laws applicable to $\langle s \cup \mathbf{M}, A, s' \cup \mathbf{M} \rangle$.

Since we know that s' is the only interpretation satisfying the heads of all dynamic laws in D' applicable to $\langle s, A, s' \rangle$, it follows from *(a)–(c)* and Case 3 above, that $s' \cup \mathbf{M}$ is the only interpretation satisfying the heads of all dynamic laws in U applicable to $\langle s \cup \mathbf{M}, A, s' \cup \mathbf{M} \rangle$. Therefore, $\langle s \cup \mathbf{M}, A, s' \cup \mathbf{M} \rangle$ is in R^U.

For the if-direction consider any $\langle s, A, s'\rangle$ in R^U, such that $s \cap \mathbf{H} = s' \cap \mathbf{H} = \mathbf{M}$. Due to *(i)* above, $s \setminus \mathbf{H}$ and $s' \setminus \mathbf{H}$ are in S. By the definition of a transition diagram described by an action description, for every dynamic law (5) in U, s' satisfies L if the law is applicable to $\langle s, A, s'\rangle$ (i.e., $s \cup A$ satisfies H and s' satisfies G).

Consider any dynamic law (5) in D'. Suppose that it is applicable to $\langle s \setminus \mathbf{H}, A, s' \setminus \mathbf{H}\rangle$. That is, $(s \setminus \mathbf{H}) \cup A$ satisfies H and $s' \setminus \mathbf{H}$ satisfies G.

Case 1. This law is in D_u. Since G and H do not contain any element of $\mathbf{H}$, $s \cup A$ satisfies H and s' satisfies G, and thus the law (5) is applicable to $\langle s, A, s'\rangle$ as well. By the definition of an update description, this law is also in U. Since $\langle s, A, s'\rangle$ is in R^U, s' satisfies L. Since L does not contain any element of $\mathbf{H}$, $s' \setminus \mathbf{H}$ satisfies L.

Case 2. This law is in D'_m. Since s contains every element H_i of $\mathbf{H}$ labeling a dynamic law in D'_m, $s \cup A$ satisfies $H \wedge H_i$. By the definition of an update description, there is a corresponding law (22) in U, which is applicable to $\langle s, A, s'\rangle$. Since $\langle s, A, s'\rangle$ is in R^U, s' satisfies L. Since L does not contain any element of $\mathbf{H}$, $s' \setminus \mathbf{H}$ satisfies L.

So far we have shown that, *(a)* for every $\langle s, A, s'\rangle$ in R^U, $s' \setminus \mathbf{H}$ satisfies the heads of every dynamic law in D' that is applicable to $\langle s \setminus \mathbf{H}, A, s' \setminus \mathbf{H}\rangle$. Moreover, we can observe that *(b)* for each dynamic law in U applicable to $\langle s, A, s'\rangle$, except for the inertia laws (23), there is a corresponding law in D' applicable to $\langle s \setminus \mathbf{H}, A, s' \setminus \mathbf{H}\rangle$, and that *(c)* D' does not contain any other dynamic laws applicable to $\langle s \setminus \mathbf{H}, A, s' \setminus \mathbf{H}\rangle$.

Since we know that s' is the only interpretation satisfying the heads of all dynamic laws in U applicable to $\langle s, A, s'\rangle$, it follows from *(a)–(c)* that $s' \setminus \mathbf{H}$ is the only interpretation satisfying the heads of all dynamic laws in D' applicable to $\langle s \setminus \mathbf{H}, A, s' \setminus \mathbf{H}\rangle$. Therefore, $\langle s \setminus \mathbf{H}, A, s' \setminus \mathbf{H}\rangle$ is in R. □

Proposition 2 *Let $(D, I, C, \sqsubset_C)$ be an ADU problem, with $D = D_u \cup D_m$. Let U be the update description of $D \cup I = D_u \cup I \cup D_m$, and let W denote a subset of D_m containing laws labeled by the elements $\mathbf{M} \subseteq \mathbf{H}$ in U. Then $D' = D_u \cup I \cup W$ is a near-solution to $(D, I, C, \sqsubset_C)$ iff M is an update set for U relative to C_o.*

Proof. Let $(D, I, C, \sqsubset_C)$ be an ADU problem, with $D = D_u \cup D_m$. Let U be the update description of $D \cup I = D_u \cup I \cup D_m$, with a set $\mathbf{H}$ of update fluents, describing the transition diagram $T^U = \langle S^U, V^U, R^U\rangle$. Let W be a subset of D_m containing laws labeled by $\mathbf{M} \subseteq \mathbf{H}$ in U. Let $T = \langle S, V, R\rangle$ be the transition diagram described by $D' = D_u \cup I \cup W$. We show that D' is a near-solution to $(D, I, C, \sqsubset_C)$ iff $\mathbf{M}$ is an update set for U relative to C_o.

For the if-direction suppose that $\mathbf{M}$ is an update set for U relative to C_o. We show that D' is a near-solution to $(D, I, C, \sqsubset_C)$ the definition of a solution hold.

(i) Since $s \cap \mathbf{H} = \mathbf{M}$ for some state $s \in S^U$, and due to Lemma 2 *(i)*, S is not empty. Therefore, D' is consistent.

(ii) It follows from the definition of D' that $D_u \cup I \subseteq D' \subseteq D \cup I$.

(iii) For any state s in S, observe that by Lemma 2 *(i)*, $s \cup \mathbf{M}$ is in S^U.

We show for any static or dynamic query c and any state s in S, that $U, s \cup \mathbf{M} \models c$ implies $D', s \models c$. Towards a contradiction assume $U, s \cup \mathbf{M} \models c$ and $D', s \not\models c$, and consider a static query c first. Since no element of $\mathbf{H}$ appears in c, and the query is static, $s \cup \mathbf{M} \not\models c$ follows. However, this contradicts the assumption. So let c be a dynamic query and h a history (10) in T such that $s_0 = s$ and $D', s_n \not\models Q$. We continue by induction on the nesting depth k of c. If $k = 0$, then Q is a static query and, since no element of $\mathbf{H}$ appears in c, it follows that $s_n \cup \mathbf{M} \not\models Q$. Moreover, by Lemma 2 *(iii)*,

$$h^U = s_0 \cup \mathbf{M}, A_0, s_1 \cup \mathbf{M}, \ldots, s_{n-1} \cup \mathbf{M}, A_n, s_n \cup \mathbf{M}$$

is a history in T^U. Thus, we conclude $U, s \cup \mathbf{M} \not\models c$, a contradiction. So let us assume the claim holds for dynamic queries with maximum nesting depth $k - 1$, and consider a dynamic query of nesting depth k. Then, Q contains only static queries and dynamic queries of nesting depth at most $k - 1$. By hypothesis, $D', s_n \not\models Q$ implies $U, s_n \cup \mathbf{M} \not\models Q$. Furthermore, again by Lemma 2 *(iii)*, the history h^U corresponding to h is a history in T^U. Thus, we conclude $U, s \cup \mathbf{M} \not\models c$, a contradiction. This proves $U, s \cup \mathbf{M} \models c$ implies $D', s \models c$ for all s in S, and any static or dynamic query c, and thus also for any basic query c.

We continue considering existential and universal queries c. We show that if c holds at $s \cup \mathbf{M}$ wrt. $S^U_{\mathbf{H}, s \cup \mathbf{M}}$, then $D' \models c$. For an existentially quantified basic query Q, the claim follows from the fact that, by definition, if c holds at $s \cup \mathbf{M}$ wrt. $S^U_{\mathbf{H}, s \cup \mathbf{M}}$, some $s' \in S^U$ exists such that $U, s' \models Q$ and $s' =_{\mathbf{H}} s$. By Lemma 2 *(i)*, we conclude that $s' \setminus \mathbf{H}$ is a state of D'. Moreover, from $U, s' \models Q$ and the fact that Q is basic, it follows that $D', s' \setminus \mathbf{H} \models Q$, and hence $D' \models c$. So let c be a universally quantified basic query Q, and towards a contradiction, assume that $D' \not\models c$. Then, there exists a state s' of D' such that $D', s' \not\models Q$. Note that by Lemma 2 *(i)* $s' \cup \mathbf{M} \in S^U$. Moreover, since Q is basic we conclude that $U, s' \cup \mathbf{M} \not\models Q$ (otherwise $D', s' \models Q$ follows which is in contradiction with our assumption). However, $U, s' \cup \mathbf{M} \not\models Q$ contradicts that c holds at $s \cup \mathbf{M}$ wrt. $S^U_{\mathbf{H}, s \cup \mathbf{M}}$. Therefore, if c holds at $s \cup \mathbf{M}$ wrt. $S^U_{\mathbf{H}, s \cup \mathbf{M}}$, then $D' \models c$ for every existential and universal query c; the same follows for any Boolean combination of existential and universal queries. This proves that if c holds at $s \cup \mathbf{M}$ wrt. $S^U_{\mathbf{H}, s \cup \mathbf{M}}$, then $D' \models c$, for any query c.

Finally, we show that $D' \models C_o$. Consider an arbitrary $s \in S$ (which exists, since by *(i)* D' is consistent). Then, due to Condition *(ii)* for update fluent sets, $s \cup \mathbf{M} \in S^U_{C_o}$. This means by definition that c holds at s wrt. $S^U_{\mathbf{H}, s}$ for every $c \in C_o$. As we have shown above, this implies $D' \models c$ for all $c \in C_o$. This proves $D' \models C_o$.

For the only-if direction let D' be a near-solution to $(D, I, C, \sqsubset_C)$. We show that $\mathbf{M}$ is an update set for U relative to C_o, i.e., *(i)* $s \cap \mathbf{H} = \mathbf{M}$ for some $s \in S^U$, and *(ii)* $S^U_{\mathbf{H}, s} \subseteq S^U_{C_o}$.

(i) Since D' is consistent there exists a state $s \in S$. Furthermore, by Lemma 2 *(i)* we conclude that $s \cup \mathbf{M} \in S^U$, for any such state s.

(ii) We first show for any static or dynamic query c and any state s in S, that $D', s \models c$ implies $U, s \cup \mathbf{M} \models c$. Towards a contradiction assume $D', s \models c$ and $U, s \cup \mathbf{M} \not\models c$, and consider a static query c first. Since no element of $\mathbf{H}$ appears in c, and the query is static, $s \not\models c$ follows. However, this contradicts the assumption $D', s \models c$. So let c be a dynamic query and h^U a history (10) in T^U such that $s_0 = s \cup \mathbf{M}$ and $U, s_n \not\models Q$. We continue by induction on the nesting depth k of c. If $k = 0$, then Q is a static query and, since no element of $\mathbf{H}$ appears in c, it follows that $s_n \setminus \mathbf{H} \not\models Q$. Furthermore, by Lemma 2 *(ii)*, $s_i =_{\mathbf{H}} s_0$ for $1 \leq i \leq n$. Therefore, by Lemma 2 *(iii)*,

$$h = s_0 \setminus \mathbf{H}, A_0, s_1 \setminus \mathbf{H}, \ldots, s_{n-1} \setminus \mathbf{H}, A_n, s_n \setminus \mathbf{H}$$

is a history in T. Thus, we conclude $D', s \not\models c$, a contradiction. So let us assume the claim holds for dynamic queries with maximum nesting depth $k-1$, and consider a dynamic query of nesting depth k. Then, Q contains only static queries and dynamic queries of nesting depth at most $k-1$. By hypothesis, $U, s_n \not\models Q$ implies $D', s_n \setminus \mathbf{H} \not\models Q$. Furthermore, again by Lemma 2 *(ii)* and *(iii)*, the history h corresponding to h^U is a history in T. Thus, we conclude $D', s \not\models c$, a contradiction. This proves $D', s \models c$ implies $U, s \cup \mathbf{M} \models c$ for all s in S, and any static or dynamic query c, and thus also for any basic query c.

We continue considering existential and universal queries c. Let s be any state in S^U such that $s \cap \mathbf{H} = \mathbf{M}$. We show that $D' \models c$ implies that c holds at s wrt. $S^U_{\mathbf{H},s}$. For an existentially quantified basic query Q, the claim follows from the fact that then there exists a state $s' \in S$, such that $D', s' \models Q$. By Lemma 2 *(i)* $s' \cup \mathbf{M}$ is a state in S^U, and since Q is basic, it follows that $U, s' \cup \mathbf{M} \models Q$. Moreover $s' \cup \mathbf{M} =_{\mathbf{H}} s$, and hence, c holds at s wrt. $S^U_{\mathbf{H},s}$ by definition. So let c be a universally quantified basic query Q, and towards a contradiction, assume that c does not hold at s wrt. $S^U_{\mathbf{H},s}$. Then there exists $s' \in S^U_{\mathbf{H},s}$, such that $U, s' \not\models Q$. By Lemma 2 *(i)* $s' \setminus \mathbf{M}$ is a state of D', and since Q is basic, $D', s \not\models Q$ follows. However, this contradicts $D' \models c$. Therefore, if $D' \models c$, then c holds at s wrt. $S^U_{\mathbf{H},s}$ for every existential and universal query c; the same follows for any Boolean combination of existential and universal queries. This proves that $D' \models c$ implies that c holds at s wrt. $S^U_{\mathbf{H},s}$.

Therefore, given that D' is a near-solution and hence $D' \models C_o$, we conclude that $S^U_{\mathbf{H},s} \subseteq S^U_{C_o}$. $\square$

References

[1] C. Alchourrón, P. Gärdenfors, and D. Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[2] J. J. Alferes, F. Banti, and A. Brogi. From logic programs updates to action description updates. In *Proc. CLIMA V (revised selected and invited papers)*, volume 3487 of *LNCS*, pages 52–77. Springer, 2004.

[3] J. J. Alferes, A. Brogi, J. A. Leite, and L. M. Pereira. Evolving logic programs. In *Proc. JELIA-02*, pages 50–61, 2002.

[4] 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.

[5] M. Balduccini and M. Gelfond. Logic programs with consistency-restoring rules. In *International Symposium on Logical Formalization of Commonsense Reasoning, AAAI 2003 Spring Symposium Series*, pages 9–18, 2003.

[6] C. Baral and Y. Zhang. Knowledge updates: Semantics and complexity issues. *Artificial Intelligence*, 164(1-2):209–243, 2005.

[7] S. Benferhat, C. Cayrol, D. Dubois, J. Lang, and H. Prade. Inconsistency management and prioritized syntax-based entailment. In *Proc. IJCAI-93*, pages 640–647, 1993.

[8] D. Billington, G. Antoniou, G. Governatori, and M. J. Maher. Revising nonmonotonic theories: The case of defeasible logic. In *Proc. German National Conference on Artificial Intelligence (KI)-99*, pages 101–112, 1999.

[9] C. Boutilier. A unified model of qualitative belief change: A dynamical systems perspective. *Artificial Intelligence*, 98(1-2):281–316, 1998.

[10] C. Cayrol, M.-C. Lagasquie-Schiex, and T. Schiex. Nonmonotonic reasoning: From complexity to algorithms. *Annals of Mathematics and Artificial Intelligence*, 22(3-4):207–236, 1998.

[11] J. Chomicki, R. van der Meyden, and G. Saake. *Logics for Emerging Applications of Databases*. Springer-Verlag, 2003.

[12] R. Dechter and A. Itai. Finding all solutions if you can find one. Technical Report ICS-TR-92-61, University of California at Riverside, September 1992.

[13] A. del Val and Y. Shoham. A unified view of belief revision and update. *Journal of Logic and Computation*, 4(5):797–810, 1994.

[14] T. Eiter, E. Erdem, M. Fink, and J. Senko. Updating action domain descriptions. In *Proc. IJCAI-05*, pages 418–423, 2005.

[15] T. Eiter, E. Erdem, M. Fink, and J. Senko. Resolving conflicts in action descriptions. In *Proc. ECAI-06*, pages 424–433, 2006.

[16] T. Eiter, E. Erdem, M. Fink, and J. Senko. Comparing action descriptions based on semantic preferences. *Annals of Mathematics and Artificial Intelligence*, 50(3-4):273–304, 2007.

[17] T. Eiter, W. Faber, N. Leone, G. Pfeifer, and A. Polleres. A logic programming approach to knowledge-state planning: Semantics and complexity. *ACM Transactions on Computational Logic*, 5(2):206–263, 2004.

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

[19] T. Eiter, M. Fink, and J. Senko. A tool for answering queries on action descriptions. In *Proc. JELIA-06*, number 4160 in LNCS, pages 473–476. Springer, 2006.

[20] T. Eiter, M. Fink, and J. Senko. Error classification in action descriptions: A heuristic approach. In *Proc. AAAI-08*, pages 905–910. AAAI Press, 2008.

[21] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence*, 57(2-3):227–270, 1992.

[22] T. Eiter and T. Lukasiewicz. Default reasoning from conditional knowledge bases: Complexity and tractable cases. *Artificial Intelligence*, 124(2):169–241, 2000.

[23] N. Friedman and J. Y. Halpern. Belief revision: A critique. *Journal of Logic, Language and Information*, 8(4):401–420, 1999.

[24] N. Friedman and J. Y. Halpern. Modeling belief in dynamic systems, part II: Revision and update. *Journal of Artificial Intelligence Research*, 10:117–167, 1999.

[25] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. International Conference and Symposium on Logic Programming (ICLP/SLP)*, pages 1070–1080, 1988.

[26] M. Gelfond and V. Lifschitz. Action languages. *Electronic Transactions on Artificial Intelligence*, 3:195–210, 1998.

[27] E. Giunchiglia and V. Lifschitz. An action language based on causal explanation: Preliminary report. In *Proc. AAAI-98*, pages 623–630, 1998.

[28] M. Goldszmidt and J. Pearl. Rank-based systems: A simple approach to belief revision, belief update, and reasoning about evidence and actions. In *Proc. KR-92*, pages 661–672, 1992.

[29] S. O. Hansson. *A Textbook of Belief Dynamics: Theory Change and Database Updating (Applied Logic)*. Kluwer, 1999.

[30] D. Harel, D. Kozen, and J. Tiuryn. Dynamic logic. In *Handbook of Philosophical Logic*, pages 497–604. MIT Press, 1984.

[31] A. Herzig. The PMA revisited. In *Proc. KR-96*, pages 40–50, 1996.

[32] A. Herzig, L. Perrussel, and I. J. Varzinczak. Elaborating domain descriptions. In *Proc. ECAI-06*, pages 397–401, 2006.

[33] Y. Jin and M. Thielscher. Representing beliefs in the fluent calculus. In *Proc. ECAI-04*, pages 823–827, 2004.

[34] Y. Jin and M. Thielscher. Iterated belief revision, revised. *Artificial Intelligence*, 171(1):1–18, 2007.

[35] Y. Jin and M. Thielscher. Reinforcement belief revision. *Journal of Logic and Computation*, 18(5):783–813, 2008.

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

[37] M. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36:490–509, 1988.

[38] J. Lang. Belief update revisited. In *Proc. IJCAI-07*, pages 2517–2522, 2007.

[39] J. Lang, F. Lin, and P. Marquis. Causal theories of action: A computational core. In *Proc. IJCAI-03*, pages 1073–1078, 2003.

[40] J. Lee and V. Lifschitz. Describing additive fluents in action language C+. In *Proc. IJCAI-03*, pages 1079–1084, 2003.

[41] R. Li and L. M. Pereira. What is believed is what is explained (sometimes). In *Proc. AAAI-96*, pages 550–555, 1996.

[42] P. Liberatore. The complexity of the language $\mathcal{A}$. *Electronic Transactions on Artificial Intelligence*, 1:13–38, 1997.

[43] P. Liberatore. The complexity of belief update. *Artificial Intelligence*, 119(1-2):141–190, 2000.

[44] P. Liberatore. A framework for belief update. In *Proc. JELIA-00*, pages 361–375, 2000.

[45] V. Lifschitz. Missionaries and cannibals in the causal calculator. In *Proc. KR-00*, pages 85–96, 2000.

[46] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating description logic aboxes. In *Proc. KR-06*, pages 46–56, 2006.

[47] N. McCain and H. Turner. Satisfiability planning with causal theories. In *Proc. KR-98*, pages 212–223. Morgan Kaufmann, 1998.

[48] J. McCarthy. Elaboration tolerance. In *Proc. CommonSense*, 1998.

[49] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[50] A. Parmar. *Formalizing Elaboration Tolerance*. Dissertation, Department of Computer Science, Stanford University, August 2003.

[51] P. Peppas. *Belief Change and Reasoning about Action – An Axiomatic Approach to Modelling Dynamic Worlds and the Connection to the Logic of Theory Change*. Dissertation, Basser Department of Computer Science, University of Sydney, 1993.

[52] P. Peppas. Belief revision. In F. van Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, chapter 8, pages 317–360. Elsevier, 2008.

[53] P. Peppas, A. C. Nayak, M. Pagnucco, N. Y. Foo, R. B. H. Kwok, and M. Prokopenko. Revision vs. update: Taking a closer look. In *Proc. ECAI-96*, pages 95–99, 1996.

[54] L. M. Pereira, J. J. Alferes, and J. N. Aparicio. Contradiction removal within well-founded semantics. In *Proc. LPNMR-91*, pages 105–119, 1991.

[55] A. S. Rao and N. Y. Foo. Minimal change and maximal coherence: A basis for belief revision and reasoning about actions. In *Proc. IJCAI-89*, pages 966–971, 1989.

[56] C. Sakama and K. Inoue. An abductive framework for computing knowledge base updates. *Theory and Practice of Logic Programming*, 3(6):671–713, 2003.

[57] S. Shapiro, M. Pagnucco, Y. Lespérance, and H. J. Levesque. Iterated belief change in the situation calculus. In *Proc. KR-00*, pages 527–538, 2000.

[58] F. van Harmelen, V. Lifschitz, and B. Porter. *Handbook of Logic in Artificial Intelligence and Logic Programming*. Elsevier, 2008.

[59] K. Wagner. Bounded query classes. *SIAM Journal on Computing*, 19(5):833–846, 1990.

[60] M. Winslett. Reasoning about actions using a possible models approach. In *Proc. AAAI-88*, pages 89–93, 1988.

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

[62] D. Zhang. Properties of iterated multiple belief revision. In *Proc. LPNMR-04*, pages 314–325, 2004.

[63] D. Zhang, S. Chopra, and N. Foo. Consistency of action descriptions. In *Proc. PRICAI-02*, pages 70–79, 2002.

[64] Y. Zhang and N. Y. Foo. Updates with disjunctive information: From syntactical and semantical perspectives. *Computational Intelligence*, 16(1):29–52, 2000.