Updating Action Domain Descriptions ⋆

Thomas Eiter [a] Esra Erdem [b] Michael Fink [a] Ján Senko [a]

[a] *Institute of Information Systems, Vienna University of Technology, Vienna, Austria*

[b] *Faculty of Engineering and Natural Sciences, Sabancı University, Istanbul, Turkey*

Abstract

Incorporating new information into a knowledge base is an important problem which has been widely investigated. In this paper, we study this problem in a formal framework for reasoning about actions and change. In this framework, action domains are described in an action language whose semantics is based on the notion of causality. Unlike the formalisms considered in the related work, this language allows straightforward representation of non-deterministic effects and indirect effects of (possibly concurrent) actions, as well as state constraints; therefore, the updates can be more general than elementary statements. The expressivity of this formalism allows us to study the update of an action domain description with a more general approach compared to related work. First of all, we consider the update of an action description with respect to further criteria, for instance, by ensuring that the updated description entails some observations, assertions, or general domain properties that constitute further constraints that are not expressible in an action description in general. Moreover, our framework allows us to discriminate amongst alternative updates of action domain descriptions and to single out a most preferable one, based on a given preference relation possibly dependent on the specified criteria. We study semantic and computational aspects of the update problem, and establish basic properties of updates as well as a decomposition theorem that gives rise to a divide and conquer approach to updating action descriptions 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 additional constraints. 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 domain descriptions in an action language, in the presence of additional constraints.

Key words: Knowledge representation, reasoning about actions and change, theory change, action languages, preference-based semantics

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. [67,12,20,61] 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. [67,39,57]. 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 [64] for background and references, and Section 8.1 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. [1]

(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.

⋆ 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.

Email addresses: eiter@kr.tuwien.ac.at (Thomas Eiter), esraerdem@sabanciuniv.edu (Esra Erdem), michael@kr.tuwien.ac.at (Michael Fink), jan@kr.tuwien.ac.at (Ján Senko).

[1] Note that the statement is wrong; its defectiveness is observed and resolved upon update.

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.

We study the incorporation problem in the context of action languages [30]. In these formalisms, actions and change are described by "causal laws." For instance, in the action language $\mathcal{C}$ [32], 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

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

which expresses that if the fluent $PowerON$ is caused to be true, then the fluent $TvON$ is caused to be true as well.

Action description languages are quite expressive to easily handle nondeterminism, concurrency, ramifications, qualifications, etc. 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 (Section 2) shows an example. There are reasoning systems, like CCALC [2] and DLV$^{\mathcal{K}}$, [3] that accept domain descriptions in an action language, like $\mathcal{C}$ or $\mathcal{K}$ respectively, and support various kinds of reasoning tasks over these descriptions, including planning, prediction and postdiction in CCALC and computing different kinds of plans in DLV$^{\mathcal{K}}$.

As far as action languages are concerned, the update problem was studied to a remarkably little extent. For the basic action language $\mathcal{A}$ (see [30]), which is far less expressive than $\mathcal{C}$, the update problem has been considered, e.g., in [44,47]. 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. We address the update problem from a more general perspective in the following ways:

- We consider a richer language (i.e., a fragment of $\mathcal{C}$) to study the update problem, and updates are represented in terms of a set of arbitrary causal laws.
- We view the update problem from a more general perspective. Sometimes, ensuring consistency is not sufficient: we might want to ensure also that the updated action description entails some scenarios, conditions, or general properties of the

[2] `http://www.cs.utexas.edu/users/tag/cc/`
[3] `http://www.dbai.tuwien.ac.at/proj/dlv/K/`

domain that cannot be expressed by causal laws. In our update framework, such further knowledge could be taken into account.

For example, for the effective use of the TV system in the above scenario, the following constraint might be imposed:

(C) Pushing the power button on the remote control is always possible. [4]

If K_{RC} is simply added to K_{TV}, then (C) is not satisfied by $K_{RC} \cup K_{TV}$: 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); note that (C) is not expressible by causal laws in the action language $\mathcal{C}$.

To represent constraints like (C), we use formulas for "queries" in action languages like in [30]; here, the formula

$$\textbf{ALWAYS executable } \{PushPB_{RC}\}. \tag{3}$$

has to evaluate to true, where $\{PushPB_{RC}\}$ stands for the concrete action of pushing the power button on the remote control. Similarly, consider the following scenario that we might want the updated action description to entail:

(S) Sometimes, when the power is on, pushing the power button on TV turns the power off, and after that if we push the power button on the TV then the power is on again.

This scenario cannot be expressed by means of causal laws either; however, it can be expressed by a formula

$$\textbf{SOMETIMES evolves } PowerON; \{PushPB_{TV}\}; \neg PowerON; \{PushPB_{TV}\}; PowerON.$$

- Sometimes, an action description can be updated in several ways. Our framework allows us to discriminate amongst alternatives and to single out a most preferable candidate as the result, based on a given preference relation possibly dependent on the additional constraint formulas.

In this paper, we consider a generic framework for incorporating new causal laws into an existing action description, that takes into account the constraint formulas to be satisfied in the end. We thus take here the stance that the causal laws, which have been designed by the user or the knowledge engineer, are to be modified, while constraint formulas are not subject to change (they might capture indisputable properties of the domain); a violation of constraints might be tolerated, though, if indicated by the user. 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 constraint formulas, to determine a

[4] Note the conceptual difference between (C) and (TV2): (C) expresses an executability condition, whereas (TV2) captures a causal relationship.

(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) and scenarios like (S) by "constraints" using formulas from an *action query language*, similar to the one in [30]. 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 constraints, C_o, (which must hold under all circumstances) and preference constraints, 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 by D'; as, in general, different candidates D' are possible, we use a (strict) *preference relation* $\sqsubset_C$ over action descriptions [5] 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 constraints. 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., the presence or absence of paths in the transition diagram).

(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 constraints 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 constraints 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 constraints. 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) We discuss methods for computing solutions and "pre-solutions" which approximate them, by disregarding solution preference. As for solutions, we focus

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

on set-inclusion and one particular weight-based comparison, as preference relations $\sqsubset_C$, which use an oracle for pre-solutions. For pre-solutions, we present a method that reduces the problem into reasoning over an action description that is constructed from the problem input; here, evaluating constraint formulas can be exploited to test given candidates.

(5) Finally, we show the applicability of our algorithm based on the computation of pre-solutions, and the usefulness of the theoretical results on the decomposability of the update problem, in the context of the Zoo World, which is an action domain proposed by Erik Sandewall in his Logic Modelling Workshop.[6] The Zoo World consists of several cages and the exterior, gates between them, and animals of several species, including humans, and its actions include moving within and between cages, opening and closing gates, and mounting and riding animals; a description of this domain in the action language $\mathcal{C}+$ was given in [1].

Our results go significantly beyond previous results in the literature (see Section 8.1), 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 constraints that can be expressed as formulas to be entailed by the updated description. 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, we 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 constraint formulas 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. Example applications in the Zoo World are considered in Section 7. After a discussion of related work and further aspects of the problem in Section 8, we conclude with a summary and issues for further research.

2 Preliminaries

We describe action domains and the updates in an action description language, a fragment of $\mathcal{C}$ [32], by "causal laws." Therefore, in the following, first we describe

[6] `http://www.ida.liu.se/ext/etai/lmw/`

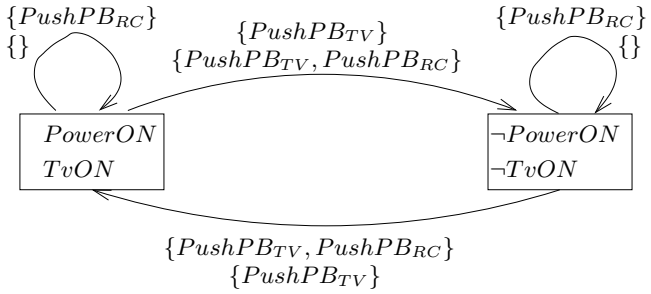

Fig. 1. A transition diagram

the syntax and the semantics of the action description language, which is defined by means of "transition systems."

While updating an action domain description, we sometimes would like ensure that the updated description entails some conditions or scenarios. Most of the time such scenarios or conditions are not expressible in the action language. We describe them as constraints using formulas from an action query language, like the one in [30]. Therefore, we also describe the action query language we use, and define satisfaction of a constraint by an action domain description.

Finally, we give sample constraints that are useful in action updates but cannot be represented in the action description language. We also discuss and emphasize the necessity of a query language in addition to the description language.

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. A *literal* is an expression of the form P or $\neg P$, where P is a fluent name. An *action* is a truth-valued function on $\mathbf{A}$, denoted by the set of action names that are mapped to t. Thus, action names represent basic (atomic) actions, while a (compound) action is identified with the basic actions taking place at the same time, providing an intuitive representation of both, atomic and concurrent, actions.

Definition 1 ([30]) *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* $V : \mathbf{F} \times S \to \{f, t\}$*, 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; and that* A *is* deterministic *in* s *if there is at most one such* s'*.*

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

caused $PowerON$ **after** $PushPB_{TV} \wedge \neg PowerON$

caused $\neg PowerON$ **after** $PushPB_{TV} \wedge PowerON$

caused $TvON$ **if** $PowerON$

caused $\neg TvON$ **if** $\neg PowerON$

inertial $PowerON, \neg PowerON, TvON, \neg TvON$

Fig. 2. An action description for K_{TV}

caused $TvON$ **after** $PushB_{RC} \wedge PowerON \wedge \neg TvON$

caused $\neg TvON$ **after** $PushB_{RC} \wedge TvON$

Fig. 3. Causal laws for K_{RC}

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 $\mathcal{C}$ [32] 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 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

$$\textbf{inertial } L_1, \ldots, L_k \tag{6}$$

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 unique 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: the static causal laws make sure that s' is a state, and handles the ramifications and the qualifications of A; whereas the dynamic causal laws handle the preconditions and the direct effects of A.

Action language $\mathcal{C}$ is based on the "principle of universal causation", according to which every fact that obtains is caused. In the definition above, the condition that s' is the *only* interpretation satisfying the heads of the applicable causal laws ensures this. For instance, the transition diagram described by the action description for K_{TV} in Figure 2 is presented in Figure 1. Consider the transition $\langle \{\neg PowerON, \neg TvON\}, \{PushPB_{TV}\}, \{PowerON, TvON\} \rangle$. The causal laws that are applicable to this transition are

caused $PowerON$ **after** $PushPB_{TV} \wedge \neg PowerON$

caused $TvON$ **if** $PowerON$.

Here $\{PowerON, TvON\}$ is the only interpretation that satisfies the heads of these causal laws, i.e., $\{PowerON, TvON\}$.

Now consider the triple $\langle \{\neg PowerON, \neg TvON\}, \{\}, \{PowerON, TvON\} \rangle$. There is only one of the two causal laws above applicable to this triple, viz. the second. There are two interpretations that satisfy the head of this causal law: $\{PowerON, TvON\}$ and $\{\neg PowerON, TvON\}$. In other words, no causal law provides a causal explanation for $PowerON$. Therefore, this triple is not a transition.

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 Why Action Languages?

In this work, we consider action languages to formalize action domains. There are several reasons for this decision.

First of all, action description languages, like $\mathcal{C}$, are quite expressive to easily handle nondeterminism, concurrency, ramifications, defaults, qualifications, state constraints, etc. For instance, we can express that tossing a coin may lead to Heads or Tails by the causal laws

$$\textbf{caused } Heads \textbf{ if } Heads \textbf{ after } Toss$$

$$\textbf{caused } \neg Heads \textbf{ if } \neg Heads \textbf{ after } Toss.$$

Concurrency is allowed unless no qualification constraint is violated or unless explicitly stated otherwise by a causal law like

$$\textbf{caused } False \textbf{ after } MoveRight \wedge MoveLeft.$$

The commonsense law of inertia is immediately expressed by causal laws of the form (6). A direct effect of turning on the power is that the power is on; a ramification of turning on the power is that TV is on. We can express such a ramification by the causal law

$$\textbf{caused } TvON \textbf{ if } PowerON.$$

We can describe that a spring-loaded door is by default closed by the causal laws:

$$\textbf{caused } Open \textbf{ if } Open.$$

Second, there are reasoning systems, like CCALC and DLV$^{\mathcal{K}}$, that accept domain descriptions in an action language, like $\mathcal{C}$ or $\mathcal{K}$ respectively, and allow various kinds of reasoning tasks over these descriptions.

Third, there is a large amount of theoretical and application-oriented work on action languages, including our earlier work on planning and monitoring. On the other hand, as discussed briefly in the introduction, the update problem was studied to a remarkably little extent in the context of action languages. This paper not only extends our earlier work to take updates into account but also fulfills the need for a general approach to updates in action languages.

2.4 Expressive Constraints

Once we describe an action domain, we may want check whether this domain description entails some observations of the world, assertions about the effects of the execution of actions, or even some scenarios. Similarly as in [18], we express such

conditions as constraints using formulas from an action query language, like the one in [30]. After that, we can check whether a given action description satisfies a given constraint using reasoning systems, e.g. CCALC (cf. the examples in Appendix C).

Now constraint formulas are formally defined as follows. [7]

An *open constraint* is either *(a)* a *static constraint* of the form

$$\textbf{holds } F, \tag{7}$$

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

$$\textbf{necessarily } Q \textbf{ after } A_1; \ldots; A_n, \tag{8}$$

where Q is an open constraint and each A_i is an action; [8] or *(c)* any propositional combination of open constraints. An *existential constraint* is an expression of the form

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

where Q is an open constraint; a *universal constraint* is of the form

$$\textbf{ALWAYS } Q, \tag{10}$$

where Q is an open constraint. A *constraint* q is a propositional combination of existential constraints and universal constraints.

For an open constraint q, its *maximal nesting depth of dynamic constraints* k is defined inductively as follows. If q is a static constraint, then $k = 0$; if q is a dynamic constraint of the form (8), then $k = k_Q + 1$, where k_Q is the maximal nesting depth of dynamic constraints in Q; if q is a Boolean combination of open constraints, then k is a maximal element from the set of maximal nesting depths of dynamic constraints of its subformulas. This definition is easily extended to (general) constraints. For an existential (universal) constraint of the form (9) (resp. of the form (10)), its maximal nesting depth of dynamic constraints is k_Q, i.e., the maximal nesting depth of dynamic constraints in Q. For a propositional combination of existential and universal constraints, its maximal nesting depth of dynamic constraints is a maximal element from the set of maximal nesting depths of dynamic constraints of its subformulas.

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,

[7] In action query languages, "constraints" as here etc. are called "queries;" the former term is more appealing here as satisfaction is required.

[8] This amounts to $[Q]A_1; \ldots; A_n$ in dynamic logic [37]; however, we stick here to the commonly used syntax of action queries.

and a set R of transitions. A *history* of T of length n is a sequence

$$s_0, A_1, s_1, \ldots, s_{n-1}, A_n, s_n \tag{11}$$

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* an open constraint Q' of form (7) (resp. (8)) relative to T (denoted $T, s \models Q'$), if the interpretation $P \mapsto V(P, s)$ satisfies F (resp. if, for every history $h = s_0, A_1, s_1, \ldots, s_{n-1}, A_n, s_n$ of T which is of length n and such that $s = s_0$, open constraint Q is satisfied at state s_n). For other forms of open constraints 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 an open constraint 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 *satisfies* a constraint q (denoted $D \models q$) if one of the following holds:

- q is an existential constraint (9) and $D, s \models Q$ for some state $s \in S$;
- q is a universal constraint (10) 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 open constraint Q,

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

For a set C of constraints, we say that D *satisfies* C (denoted $D \models C$) if D satisfies every constraint in C. Consider, e.g., the action description presented in Figure 2. It does not satisfy any set of constraints containing

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

because this constraint is not satisfied at the state $\{TvON, PowerON\}$; but, it satisfies the constraints:

$$\textbf{ALWAYS}\ \ \textbf{holds}\ PowerON \equiv TvON, \tag{12}$$

$$\begin{aligned}\textbf{ALWAYS}\quad &\textbf{holds}\ PowerON \wedge TvON \supset \\ &\neg\textbf{necessarily}\ (\textbf{holds}\ TvON)\ \textbf{after}\ \{PushPB_{TV}\}.\end{aligned} \tag{13}$$

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

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

where Q is an open constraint and each A_i is an action, stands for the dynamic constraint

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

an expression of the form

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

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

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

and

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

where each A_i is an action, stands for

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

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

2.4.1 Examples

To get a better intuition about the capability of constraints, we give some examples of properties that can be expressed by them.

- *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 constraint

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

  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 (11) 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

constraint:

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

For instance, the constraint

$$\begin{aligned}\textbf{SOMETIMES evolves } & PowerON; \{PushPB_{TV}\}; \\ & \neg PowerON; \{PushPB_{TV}\}; PowerON.\end{aligned}$$

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

$$\begin{aligned}&\{PowerON, TvON\}, \{PushPB_{TV}\}, \\ &\{\neg PowerON, \neg TvON\}, \{PushPB_{TV}\}, \{PowerON, TvON\}.\end{aligned}$$

- *(Non-)executability of an action:* Like in [16], 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

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

- *Mandatory and possible effects of actions:* Like in [16], 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 constraints, F describes the effects and G the context.

2.4.2 Constraints vs. Causal Laws

In all action languages [30], queries have been expressed in a language different from the action description languages. As we consider constraints as queries that evaluate to true, it may look suggestive to merge causal laws and constraints into a single set of formulas that constitute an action description. However, constraints and causal laws are conceptually different: causal laws are axioms that describe action domains in a *generative manner* (in particular, in action language $\mathcal{C}$ via *causation*), whereas constraints express conditions (which may also refer to time steps) that we would like to ensure about an action domain; they thus serve for *eliminating* unwanted models. In other words, constraints restrict the possibilities for an action description, but they are non-constructive in the sense that they do not causally

generate transitions. The latter however is at the heart of $\mathcal{C}$ and many other action languages: each edge $\langle s, A, s' \rangle$ in the transition diagram is causally explained (or "generated"), meaning that the follow up state s' is uniquely described by the literals in heads of the causal laws that "fire", i.e., the causal laws that are applicable to $\langle s, A, s' \rangle$.

For instance, if the action description consists of the only causal law

$$\textbf{caused } G \textbf{ after } A_1 \wedge F,$$

where every fluent is inertial, then the transition diagram has the edge $\langle \{F, \neg G\}, \{A_1\}, \{F, G\} \rangle$, i.e., we can get from the state $s = \{F, \neg G\}$ to the state $s' = \{F, G\}$ with the occurrence of the action $\{A_1\}$. However, the transition diagram does not have the edge $\langle \emptyset, \{A_1\}, \{F, G\} \rangle$. If instead of the causal law above, we consider the constraint

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

which is similar to the causal law, then both $\langle \{F, \neg G\}, \{A_1\}, \{F, G\} \rangle$ and $\langle \emptyset, \{A_1\}, \{F, G\} \rangle$ would be included in the transition diagram (since F is false wrt. $s = \emptyset$, the implication is true).

Although in some cases a constraint may be expressed by an equivalent causal law (or multiple such laws), this is not always the case. Moreover, the meaning of a set of causal laws is described by a set of nodes and a set of edges that form a transition system, where each of these edges expresses a causal relationship (which generates the edge), whereas roughly speaking, the meaning of a constraint is described by a set of paths in the transition diagram without such a causal relation. In other words, constraints might describe conditions characterizing subgraphs of a transition diagram. Consequently, some constraints cannot be expressed as causal laws, for instance "existential constraints" like the constraint

$$\textbf{SOMETIMES possibly } F \textbf{ after } A,$$

the constraint (21), and similar constraints in Section 2.4.1. They can not be expressed via causal laws, as the latter are inherently universal statements. Also "universal" constraints like

$$\textbf{ALWAYS } (\textbf{possibly } F \textbf{ after } A) \wedge (\textbf{possibly } \neg F \textbf{ after } A)$$

(which implicitly enforce existence of causal transitions) are difficult to express via causal laws. Another aspect is that constraints allow us to talk about sequences of actions, while causal laws do not.

Due to the syntactic and the semantic differences between causal laws and queries, the reasoning systems (like CCALC) based on action languages also have different syntax for query formulas (cf. Appendix C); hence a difference in practice as well.

For instance, in reasoning systems, queries can be used to describe reasoning tasks (like temporal projection or planning) about a given action domain.

Although there are some formalisms (like situation calculus as in [60], dynamic logic as in [37], or answer set programming as in [48]) that can be used to describe both axioms and constraints, a distinction between formalisms to express axioms and constraints is not unusual in other areas either. Consider, for instance, the description of a circuit in propositional logic and the conditions we want to check about this circuit that are expressed in a temporal logic [25]. Also, consider ontologies described in ontology description languages (like RDF), and constraints described in ontology query languages (like SPARQL).

The differences between causal laws and query formulas also affect the computational efficiency of reasoning systems. For instance, given a domain description and a query, CCALC checks whether the query is entailed by the domain description as follows:

(1) it transforms the causal laws into a propositional theory Γ_D,
(2) it transforms the negated query into a propositional theory Γ_P,
(3) it checks whether $\Gamma_D \cup \Gamma_P$ is satisfiable;
(4) if $\Gamma_D \cup \Gamma_P$ is unsatisfiable, it returns Yes;
(5) otherwise, it returns No and presents a counter example extracted from a satisfying interpretation for $\Gamma_D \cup \Gamma_P$.

The transformations in the first two steps are different: the one in 1) is based on literal completion, whereas the one in 2) is based on a simpler procedure (see [31] for a detailed description). Such a difference allows one to check the entailment of other queries without executing the first step again. If we had described a constraint by means of causal laws, then in general we would have to transform the union of the causal laws and the constraint into a propositional theory; for large domain descriptions, like the Zoo World, such a bulk transformation would lead to inefficient computations.

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 pre-solution. The basic problem is a theory change problem, i.e., a problem of incorporating new information into an existing stock of knowledge (cf. Sections 4.2 and 8.2 for more detailed discussions of relations to well-known work in this area). Since we study the incorporation problem in the context of an action language, we consider single causal laws as the atomic entities that are subject to change (for a discussion how to refine this further, see Section 8.3). In addition to causal laws for incorporation, the new information

may contain constraints that characterize intended properties of the change (reasons for the distinction between causal laws and constraints have been discussed in the previous section). Concerning solutions of the problem, we aim at keeping the size of the search space practically reasonable, as well as at building on natural analogies with change operators developed in other areas of database or AI research (cf. [67,61,57] and references therein).

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 partial action description), a set $C = C_o \cup C_p$ of constraints, 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 constraints in C_o are "hard (obligatory) constraints" that have to be satisfied in an acceptable action description, while the constraints 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 2 (Action Description Update) *Given an action description $D = D_u \cup D_m$, a set I of causal laws (a partial action description), a set $C = C_o \cup C_p$ of constraints, 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 $D' \sqsubset_C D''$. [9]

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* pre-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 *(iii)* imposes semantical constraints C_o 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).

[9] Note that soft constraints C_p are used implicitly in this definition, since the preference relation $\sqsubset_C$ is one in which $C = C_o \cup C_p$ is explicitly known as parameter.

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 certain paths present in the transition diagram of the description) by a function $weight$, we can take $\sqsubset_C$ to be an operator $<_{weight}$ comparing the weights of the action descriptions; then an action description D is less preferable than an action description D' if $D <_{weight} D'$.

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) [66], which should be governed by different sets of postulates in axiomatic approaches like the AGM theory [2] and the KM theory [39]. Our notion of ADU has more of a revision flavor, but we do not govern it with AGM or KM postulates, as the formalism does not satisfy the prerequisites; see Section 8.2 for more discussion. However, the constraints 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.

Eventually we remark that, in descriptive domains, like physical domains, one might carry out tests and collect respective results (observations) in order to find out erroneous causal laws. In this case, the update problem would be rather viewed as a diagnosis problem. Note however, that such an approach hinges on the possibility to make observations for learning causal relationships. In contrast, our approach is intended to also allow for normative (artificial) worlds modeled as action descriptions (e.g., agent systems, games, protocols), where the world is designed, rather than perceived. In such domains, and likewise for physical worlds that are not observable (where one is impeded to make observations for whatever reason), it is not feasible to treat the update problem as a diagnosis problem.

3.1 Examples

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

Example 1 Let D be the action description for K_{TV} in Figure 2, i.e., $D = D_u \cup D_m$

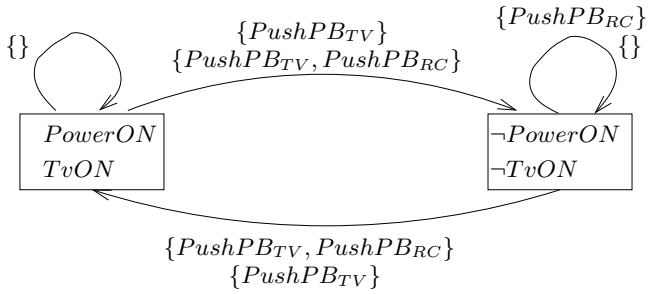

Fig. 4. Transition diagram described by $D \cup I$ of Ex. 1.

with

$$\begin{aligned} D_u = \{\ & \textbf{caused}\ PowerON\ \textbf{after}\ PushPB_{TV} \wedge \neg PowerON, \\ & \textbf{caused}\ \neg PowerON\ \textbf{after}\ PushPB_{TV} \wedge PowerON, \\ & \textbf{inertial}\ PowerON, \neg PowerON, TvON, \neg TvON \quad \} \end{aligned}$$

and $D_m = \{\textbf{caused}\ TvON\ \textbf{if}\ PowerON, \textbf{caused}\ \neg TvON\ \textbf{if}\ \neg PowerON\}$, and let I be the set of causal laws for K_{RC} in Figure 3:

$$\begin{aligned} & \textbf{caused}\ TvON\ \textbf{after}\ PushPB_{RC} \wedge PowerOn \wedge \neg TvON, \\ & \textbf{caused}\ \neg TvON\ \textbf{after}\ PushPB_{RC} \wedge TvON. \end{aligned}$$

Furthermore, let $C = C_o$ contain besides constraints (3) and (13):

$$\begin{aligned} & \textbf{ALWAYS}\ \ \textbf{executable}\ \{PushPB_{RC}\}, \\ & \textbf{ALWAYS}\ \ \textbf{holds}\ PowerON \wedge TvON \supset \\ & \qquad\qquad \neg\textbf{necessarily}\ (\textbf{holds}\ TvON)\ \textbf{after}\ \{PushPB_{TV}\}, \end{aligned}$$

also the constraint

$$\textbf{ALWAYS executable}\ \{PushPB_{TV}\}, \tag{14}$$

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), i.e., **caused** $TvON$ **if** $PowerON$, from $D \cup I$. □

For an instance of a semantic definition of $\sqsubset_C$, consider the following setting based on weights that are assigned to constraints on C (i.e., *weighted constraints* in [18]). We define the weight of an action description D relative to a set C of constraints, and a weight function $f : C \to \mathbb{R}$ mapping each constraint in C to a real number

by

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

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

Example 2 Reconsider our previous example where C_p consists of the constraint (13) with weight 1:

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

Suppose that the preference relation $\sqsubset_C$ is defined in terms of a weight function on constraints (i.e., $\sqsubset_C = <_{weight_q}$). Then, the action descriptions $D' = (D \cup I) \setminus \{\textbf{caused } TvON \textbf{ if } PowerON\}$ and $D'' = D_u \cup I$ satisfy C_o and thus are pre-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 descriptions by means of weighted constraints and other semantic preferences, we refer the reader to [18].

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 pre-solutions, and establish a disjoint factorization result that allows decomposing an ADU into smaller parts.

4.1 Basic Update Properties

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

Definition 3 (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 constraints, let us denote by C_D the set $\{c \in C \mid D \models c\}$.

Definition 4 *Given a set of constraints 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 constraints 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 non-

negative (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 constraints 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)* for every D', $D_u \cup I \subseteq D' \subseteq D \cup I$ 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 $\mathbf{F}$. 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. $\square$

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 [5,20].

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 $(D, D, C, \sqsubset_C)$.*

Note that Void Update and Idempotence can easily be extended to cases where $I \subseteq D$: given that D is consistent and $D \models C$, it holds that D is a solution if $\sqsubset_C$ is non-minimizing; for strongly non-minimizing $\sqsubset_C$, it is the unique solution.

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 [5,20].

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

inertial $LightON, \neg LightON$,
caused $LightON$ **after** $SwitchLight \wedge \neg LightON$,
caused $\neg LightON$ **after** $SwitchLight \wedge LightON$.

Since D is consistent and $\subset$ is strongly non-minimizing, we can state for any set C of constraints, 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 $<_{weight_q}$ with nonnegative weights for any constraint $c \in C$ 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. $\square$

In the literature, two kinds of changes have been identified with the incorporation of new information, viz. revision (which adds more precise knowledge about the domain) and update (which is a change of the world per se) [66]. Despite the nature of change, a distinction is made whether beliefs are represented by a theory, i.e., by a logically closed set of sentences, or through a theory base (knowledge base), i.e., a finite representation of a theory [33]. Ideally, operators for the different kinds of belief change are characterized by different sets of axioms or postulates like the AGM theory [2] for belief revision and the KM theory [39] for belief base update.

Fitting our approach in this context, we first observe that a common basic assumption of the different belief change postulates is that beliefs are sentences from a given logical language which is closed under the standard Boolean connectives; this is not the case for action languages. In order to evaluate our approach in the style of AGM, or KM respectively, it is thus necessary to interpret and adapt respective postulates. We note, however, that additional assumptions of the AGM theory regarding the underlying inference relation, that it satisfies super-classicality, modus ponens, the deduction theorem etc. (cf. [57]), are inapplicable.

Since an action description constitutes a finite representation of a theory about an action domain, our update approach has to be classified as operating on belief bases. Let us briefly recall the KM postulates for belief base update: [10]

(U1) $KB \diamond \phi$ implies ϕ.
(U2) If KB implies ϕ, then $KB \diamond \phi \equiv KB$.
(U3) If both KB and ϕ are satisfiable, then $KB \star \phi$ is satisfiable.
(U4) If $KB_1 \equiv KB_2$ and $\phi_1 \equiv \phi_2$, then $KB_1 \diamond \phi_1 \equiv KB_2 \diamond \phi_2$.
(U5) $(KB \diamond \phi_1) \wedge \phi_2$ implies $KB \diamond (\phi_1 \wedge \phi_2)$.
(U6) If $KB \diamond \phi_1$ implies ϕ_2 and $KB \diamond \phi_2$ implies ϕ_1, then $KB \diamond \phi_1 \equiv KB \diamond \phi_2$.
(U7) If KB is complete, then $(KB \diamond \phi_1) \wedge (KB \diamond \phi_2)$ implies $KB \diamond (\phi_1 \vee \phi_2)$.
(U8) $(KB_1 \vee KB_2) \diamond \phi \equiv (KB \diamond \phi_1) \vee (KB \diamond \phi_2)$.

Besides these postulates for update, Katsuno and Mendelzon have reformulated the AGM postulates for the case of belief base revision in propositional logic:

(R1) $KB \star \phi$ implies ϕ.
(R2) If $KB \wedge \phi$ is satisfiable, then $KB \star \phi \equiv KB \wedge \phi$.
(R3) If ϕ is satisfiable, then $KB \star \phi$ is satisfiable.
(R4) If $KB_1 \equiv KB_2$ and $\phi_1 \equiv \phi_2$, then $KB_\star \phi_1 \equiv KB_2 \star \phi_2$.
(R5) $(KB \star \phi_1) \wedge \phi_2$ implies $KB \star (\phi_1 \wedge \phi_2)$.

[10] Hansson's [33] postulates for contraction would, in style of Levi Identity give rise to revision via contraction and expansion; however, this requires the use of negation, which we lack here.

(R6) If $(KB \star \phi_1) \wedge \phi_2$ is consistent, then $KB \star (\phi_1 \wedge \phi_2)$ implies $(KB \star \phi_1) \wedge \phi_2$.

4.2.1 Interpretation of Postulates

As for an interpretation of these postulates in our setting, we may take the subsumption relation between an action description and a set of causal laws for characterizing implication (and thus equivalence).

Lemma 1 (Equivalence) *Let D_1 and D_2 be action descriptions over a signature $\mathcal{L} = \langle \mathbf{F}, \mathbf{A} \rangle$. Suppose that for every causal law l over $\mathcal{L}$, it holds that D_1 subsumes l iff D_2 subsumes l. Then $T(D_1) = T(D_2)$.*

Proof. Let $T(D_1) = \langle S_1, V_1, R_1 \rangle$ and $T(D_2) = \langle S_2, V_2, R_2 \rangle$. Towards a contradiction first suppose that $S_1 \neq S_2$. W.l.o.g., assume that s is a state in S_1 such that $s \notin S_2$. Consider an arbitrary fluent F, and let $\bar{F}$ denote $\neg F$ if F is true in s, and F otherwise. Then, D_1 does not subsume $l =$ **caused** $\bar{F}$ **if** $\bigwedge_{s(G)=t} G \wedge \bigwedge_{s(G)=f} \neg G$, whereas D_2 trivially subsumes l, a contradiction. Hence, $S_1 = S_2$ holds, and therefore also $V_1 = V_2$. Next, suppose $R_1 \neq R_2$, and w.l.o.g. assume that $\langle s, A, s' \rangle \in R_1$ and $\langle s, A, s' \rangle \notin R_2$. Again consider an arbitrary fluent F, and let $\bar{F}$ denote $\neg F$ if F is true in s', and F otherwise. Then, D_1 does not subsume

$$l = \textbf{caused } \bar{F} \textbf{ if } \bigwedge_{s'(G)=t} G \wedge \bigwedge_{s'(G)=f} \neg G \textbf{ after } \bigwedge_{H_a \in A} H_a \wedge \bigwedge_{H_a \in \mathbf{A} \setminus A} \neg H_a \wedge \bigwedge_{s(H_s)=t} H_s \wedge \bigwedge_{s(H_s)=f} \neg H_s,$$

whereas D_2 trivially subsumes l, a contradiction. Therefore $R_1 = R_2$ holds as well. This proves the claim. □

Closing the language under conjunction is also no problem, since an action description can be regarded as the conjunction of its causal laws. However, the meaning of negation (and disjunction) of causal laws and action descriptions is ambiguous and undefined. Therefore, we refrain from an interpretation of postulates (**U7**) and (**U8**).

Another difficulty arises from the fact that the new information to be incorporated into our action description is characterized by syntactically and semantically different entities, namely causal laws and constraints. Naturally, KB implies ϕ might be understood component-wise as KB subsumes the causal laws given by ϕ and KB satisfies the constraints given by ϕ.

Given these considerations, we paraphrase the postulates as follows:

(R1), (U1) If D' is a solution to $(D, I, C, \sqsubset_C)$, then D' subsumes I and $D' \models C$.

(R2) If $D \cup I$ is consistent and $D \cup I \models C$, then $T(D') = T(D)$ for any solution D' of $(D, I, C, \sqsubset_C)$.

(U2) If D subsumes I and $D \models C$, then $T(D') = T(D)$ for any solution D' of $(D, I, C, \sqsubset_C)$.

(R3) If there exists an action description D' such that D' is consistent, D' subsumes I and $D' \models C$, then there exists a solution to $(D, I, C, \sqsubset_C)$.

(U3) If D is consistent, and there exists an action description D' such that D' is consistent, D' subsumes I, and $D' \models C$, then $(D, I, C, \sqsubset_C)$ has a solution.

(R4), (U4) If $T(D_1) = T(D_2)$, $T(I_1) = T(I_2)$, and $C_1 \equiv C_2$, then $T(D_1') = T(D_2')$ for any solutions D_1' and D_2' of $(D_1, I_1, C_1, \sqsubset_C)$ and $(D_2, I_2, C_2, \sqsubset_C)$, respectively.

(R5), (U5) If D' is a solution to $(D, I_1, C_1, \sqsubset_C)$ and $D' \cup I_2$ subsumes l, then D'' subsumes l for some solution D'' of $(D, I_1 \cup I_2, C_1 \cup C_2, \sqsubset_C)$.

(R6) If D' is a solution to $(D, I_1, C_1, \sqsubset_C)$ and $D' \cup I_2$ is consistent and satisfies C_2, then D'' subsumes l implies $D' \cup I_2$ subsumes l, for some solution D'' of $(D, I_1 \cup I_2, C_1 \cup C_2, \sqsubset_C)$.

(U6) If D_1' is a solution to $(D, I_1, C_1, \sqsubset_C)$ such that D_1' subsumes I_2 and $D_1' \models C_2$, and D_2' is a solution to $(D, I_2, C_2, \sqsubset_C)$ such that D_2' subsumes I_1 and $D_2' \models C_1$, then $T(D_1') = T(D_2')$.

Obviously, **R1** and **U1** hold by definition, whereas **R2** holds for strongly non-minimizing $\sqsubset_C$. For **U2**, we also know from Proposition 1 that it holds if D is consistent, in addition to a strongly non-minimizing $\sqsubset_C$. Both conditions are necessary.

Concerning **R3** and **U3**, they do not hold in general, unless $D_u \subseteq D' \subseteq D \cup I$. In case of the latter they hold by definition; to wit the former, let $D = D_u = \{\textbf{caused } F\}$, $\sqsubset_C = \subset$, $I = \{\textbf{caused } \neg F\}$, and $C = \emptyset$. Note that the property holds if $C = \emptyset$ and $D_u = \emptyset$.

Proposition 2 (Solution Existence) *Let $(D, I, C, \sqsubset_C)$ be an ADU problem, such that D' is consistent, $D' \models C$, and D' subsumes I for some action description D' with signature $\mathcal{L}$. Then, there exists a solution to $(D, I, C, \sqsubset_C)$ if (i) $D_u \subseteq D' \subseteq D \cup I$ and $D' \cup I \models C$, or (ii) $C = \emptyset$ and $D_u = \emptyset$.*

Proof. Note that consistency of D' and that D' subsumes I implies that I is consistent. In Case (i) this implies that $D' \cup I$ is consistent. Furthermore, $D_u \subseteq D' \cup I \subseteq D \cup I$ and $D' \cup I \models C$ hold. Hence, $D' \cup I$ is a pre-solution, which proves the existence of a solution. For (ii), observe that $D_u = \emptyset \subseteq I \subseteq D \cup I$, and that $I \models C$ (since $C = \emptyset$). Hence, I is a pre-solution, which proves the existence of a solution. □

Irrelevance of Syntax (**R4**/**U4**) does not hold, even for $\sqsubset_C = \subset$ and $C = \emptyset$: Consider $D_1 = \{\textbf{caused } F, \textbf{caused } \neg F\}$, $D_2 = \{\textbf{caused } G, \textbf{caused } F \textbf{ if } G, \textbf{caused } \neg F \textbf{ if } G\}$, and $I_1 = I_2 = \emptyset$.

We remark, that the above counterexample is a canonical one, in the sense that $I_1 = I_2 = \emptyset$ and $C = \emptyset$, however with inconsistent D_1 and D_2. Note, that is is easily modified to a counterexample where D_1 and D_2 are consistent (and, for instance, I_1 and I_2 are nonempty).

Property **R5**, **U5** holds if just consistency is required ($C = \emptyset$), $D' \cup I_2$ is consistent, and $\sqsubset_C$ is strongly non-minimizing. In general it fails as witnessed by: $D = D_m = \{$**caused** F **after** $A\}$, $\sqsubset_C{=}\subset$, $I_1 = \{$**caused** $\neg F$ **after** $A \wedge \neg F\}$, $I_2 = \{$**caused** $\neg F$ **after** $A \wedge F\}$, and $C = \{$**SOMETIMES executable** $A\}$. In this case, $D'' = I_1 \cup I_2$ is the only solution of $(D, I_1 \cup I_2, C, \sqsubset_C)$ (since $D \cup I_1 \cup I_2$ does not satisfy C). However, D'' does not subsume **caused** F **after** A, which is the case for $D' = D \cup I_1$. The property also does not hold for strongly non-minimizing $\sqsubset_C$ in case of $C = \emptyset$ if $D' \cup I_2$ is inconsistent: Let $D = D_m = \{$**caused** $F\}$, $\sqsubset_C{=}\subset$, $I_1 = \{$**caused** $G\}$, $I_2 = \{$**caused** $\neg F\}$. Then, $D'' = I_1 \cup I_2$, which does not subsume **caused** F.

Similarly, **R6** holds if just consistency is required ($C = \emptyset$), and $\sqsubset_C$ is strongly non-minimizing. In general it fails: Let $D = D_m = \{$**caused** F **after** $A \wedge F\}$, $\sqsubset_C{=}\subset$, $I_1 = \emptyset$, $I_2 = \{$**caused** F **after** $A \wedge \neg F\}$, $C_1 = \{$**SOMETIMES** $\neg$**executable** $A\}$, and $C_2 = \emptyset$. Then, $D'' = I_2$, which subsumes $l =$ **caused** $\neg F$ **after** $A \wedge F$. However, $D' \cup I_2 = D \cup I_2$ does not subsume l, although it is consistent and trivially satisfies C_2.

Proposition 3 (Unique Consequence) *Let $(D, I_1, \emptyset, \sqsubset_C)$ and $(D, I_1 \cup I_2, \emptyset, \sqsubset_C)$ be ADU problems, such that $\sqsubset_C$ is strongly non-minimizing wrt. $\mathcal{L}$. If D' is a solution to $(D, I_1, \emptyset, \sqsubset_C)$ and $D' \cup I_2$ is consistent, then $D' \cup I_2$ is a solution to $(D, I_1 \cup I_2, \emptyset, \sqsubset_C)$.*

Proof. Obviously $D' \cup I_2$ is a pre-solution of $(D, I_1 \cup I_2, \emptyset, \sqsubset_C)$, since it is consistent and trivially satisfies $C = \emptyset$. Towards a contradiction assume that there is a consistent action description D'' such that $(D' \cup I_2) \sqsubset_C D''$ and $D_u \cup I_1 \cup I_2 \subseteq D'' \subseteq D \cup I_1 \cup I_2$. Then, since $\sqsubset_C$ is strongly non-minimizing wrt. $\mathcal{L}$, we conclude that $(D' \cup I_2) \subset D''$. Let $D_1 = D'' \setminus (I_2 \setminus D')$. Then, $D' \subset D_1$. Furthermore, D_1 is consistent (because satisfaction of static laws is monotone) and trivially satisfies C, i.e., D_1 is a pre-solution to $(D, I_1, \emptyset, \sqsubset_C)$. Because $\sqsubset_C$ is strongly non-minimizing wrt. $\mathcal{L}$, it follows from $D' \subset D_1$ that $D' \sqsubset_C D_1$. This contradicts the assumption that D' is a solution to $(D, I_1, \emptyset, \sqsubset_C)$. Therefore, D'' cannot exist, i.e., $D' \cup I_2$ is a solution to $(D, I_1 \cup I_2, \emptyset, \sqsubset_C)$. $\square$

Eventually, **U6** fails to hold even for strongly non-minimizing $\sqsubset_C$ if just consistency is required $C = \emptyset$: Let $D = \emptyset$, $\sqsubset_C{=}\subset$, $I_1 = \{$**caused** F **after** $A \wedge F\}$, $I_2 = \{$**caused** F **after** $A\}$, and $C_1 = C_2 = \emptyset$. Then I_1 subsumes I_2 and vice versa, but $T(I_1) \neq T(I_2)$.

4.2.2 Discussion

Summing up, we observe that even in the simple setting without unmodifiable laws ($D_u = \emptyset$), without constraints ($C = \emptyset$), and with set inclusion as preference relation ($\sqsubset_C = \subset$), not all postulates are satisfied. Concerning the revision postulates, apart from an additional consistency requirement for solution existence in **R5**, the only postulate that completely fails is Irrelevance of Syntax (**R4**/**U4**). This is intuitive, however, given that the causal information in an action description depends on its syntactical representation in terms of causal laws. While different sets of causal laws, i.e., knowledge bases, may represent the same transition diagram, when (the same) new information is incorporated, this no longer needs to be the case.

Concerning the update postulates, in addition to the failure of **U4**, postulate **U2** does not hold in general. The reason is that solutions must be consistent, a property which has been discussed as one of the discriminating properties between update and revision. In this respect, our approach certainly acts like a revision operator. Moreover, **U6** fails to hold even in this simple setting.

Let us turn to more sophisticated ADU problems, where more than (static) consistency is required for solutions, and dynamic requirements need to hold after changing the knowledge base. Recall that in general such requirements cannot be expressed in terms of causal laws. (With the latter, one can represent action domains that satisfy the respective requirement, which would amount to specify the solution as the new information to be incorporated, however, rather than expressing the requirement itself.) As soon as dynamic requirements can be demanded ($C \neq \emptyset$), several postulates cease to hold: **R3**/**U3**, **R5**/**U5**, and **R6**. For **R3**/**U3**, the reason is that the solution space is constrained to causal laws occurring in $D \cup I$ (which we consider a reasonable assumption for practical change operations in our setting). In case of **R5**/**U5**, and **R6**, which are related to the *supplementary AGM postulates* (i.e., AGM postulates K$*$7 and K$*$8 [57]), the simple counterexamples reveal that the main reason for this failure is due to the non-monotonicity of the action language and that it is rather independent of the problem definition.

4.3 Disjoint Factorization

We next consider a structural property of solutions and pre-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 5 (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, there exists a 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 2 *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) $S = \{s_0 \cup s_1 \mid s_0 \in S^0, s_1 \in S^1\}$;
(ii) *If $R^0 \neq \emptyset$ and $R^1 \neq \emptyset$ then, for $\langle s_0^0, A^0, s_1^0\rangle \in R^0$ and $\langle s_0^1, A^1, s_1^1\rangle \in R^1$, $\langle s_0^0 \cup s_0^1, A^0 \cup A^1, s_1^0 \cup s_1^1\rangle \in R$;*
(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 constraints (cf. Lemma 2 *(ii)*). This can be guaranteed by the NOP property, which will in fact be sufficient for composing pre-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 constraints.

Definition 6 (AD/Constraint 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 constraints 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 constraints in separate parts while disregarding preference.

Definition 7 (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 pre-solutions of an ADU problem can be obtained from those of a near-decomposition, provided that some ramifying conditions hold. We say that a constraint c occurs *positively* (resp. *negatively*) in a set C of constraints, if c occurs in the scope of an even (resp. odd) number of negations in a constraint 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 pre-solution to $(D^i, I^i, C^i, \sqsubset_{C^i})$ such that X^i has NOP if some dynamic constraint occurs negatively in C^{1-i}, for $i = 0, 1$. Then $X^0 \cup X^1$ is a pre-solution to $(D, I, C, \sqsubset_C)$.

(ii) Let X be a pre-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 constraint occurs positively in C^{1-i}, for $i = 0, 1$. Then, X^i is a pre-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 constraint 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 constraint 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 constraint $c \in \mathcal{L}_{C^i} \subseteq \mathcal{L}^i$ 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 2 *(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 constraints, then $X^0 \cup X^1$ satisfies C iff X^0 satisfies C^0 and X^1 satisfies C^1.

We next consider dynamic constraints c of the form **necessarily** Q **after** $A_1; \ldots; A_n$ or ¬**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 constraint; (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 constraint. We proceed by induction on the nesting depth k of the constraint.

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 2 *(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 constraints. 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 2 *(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 constraints. 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 $h = s^i, A_1, s_1^i, \ldots, s_{n-1}^i, A_n, s_n^i$, such that $s_n^i \not\models Q$. Since X^{1-i} has NOP, there exists a sequence of $n+1$ states, such that $h^{1-i} = s^{1-i}, \emptyset, s_1^{1-i}, \ldots, s_{n-1}^{1-i}, \emptyset, s_n^{1-i}$ is a history of X^{1-i}. By Lemma 2 *(ii)*, $h = s^i \cup s^{1-i}, A_1, \ldots, A_n, s_n^i \cup s_n^{1-i}$ is a history of $X^0 \cup X^1$. Furthermore, $s_n^i \not\models Q$ implies $s_n^i \cup s_n^{1-i} \not\models Q$ because $c \in X^i$ and Q contains only static constraints. 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 2 *(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 constraints. Contradiction. This proves (2) for $k = 0$.

Induction Step: Let (1) and (2) be true for dynamic constraints of nesting depth at most $k-1$ and consider a dynamic constraint c of nesting depth k. Then, Q contains only static constraints and dynamic constraints 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 constraints with a respective justification that Q contains only static constraints and dynamic constraints of nesting depth at most $k-1$.

So far, we have shown that (1) and (2) hold for any open constraint. By the structure of S (cf. Lemma 2 *(i)*), we conclude for any existential or universal constraint 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 constraint, 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 constraint. Therefore, $X^i \models C^i$ and X^{1-i} has NOP if C^i contains a negative dynamic constraint, 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 constraint 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 pre-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 constraint occurs negatively in C^{1-i}. We show that $X^0 \cup X^1$ is a pre-solution to $(D, I, C, \sqsubset_C)$. By Lemma 2 *(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 pre-solution to $(D, I, C, \sqsubset_C)$.

Case *(ii)*: Let X be a pre-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 constraint occurs positively in C^{1-i}. We prove that for $i = 0, 1$, X^i is a pre-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

$D^1_u \cup I^1 \subseteq X^1 \subseteq D^1 \cup I^1$. 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. $\square$

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

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 8 (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 ($\sqsubset_C = \subset$) is always factorizable (e.g., taking $\subset$ as the preference relations of the factorization). We also remark that if the set of constraints C is decomposable with respect to $(\mathcal{L}^0, \mathcal{L}^1)$, then the constraint weight preference $<_{weight_q}$ 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 9 (ADU Decomposition) *A* decomposition *of a given 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 constraint occurs in C, or (ii) no dynamic constraint 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 constraints (combining the transition graphs of the components is then unproblem-

atic). Item *(ii)* accounts for possible dynamic constraints 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 $X^0 = D \cup I \setminus \{$**caused** $TvON$ **if** $PowerON\}$ 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)* $X^0 \cup X^1 = (D \cup D' \cup I) \setminus \{$**caused** $TvON$**if** $PowerON\}$ 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 $X^0 \cup X^1 = (D \cup D' \cup I) \setminus \{$**caused** $TvON$**if** $PowerON\}$ is a solution to $(D \cup D', I, C, <_{weight_q})$, as $X^0 = D \cup I \setminus \{$**caused** $TvON$**if** $PowerON\}$ 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 pre-solution to this ADU problem since $D_u \cup I$ is a pre-solution to $(D, I, C, <_{weight_q})$. Moreover, setting the weight of constraint (13)

ALWAYS holds $PowerON \wedge TvON \supset$

¬**necessarily** (**holds** $TvON$) **after** $\{PushPB_{TV}\}$

to 0 (which amounts to assigning the preferred constraints 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, [11] and Theorem 2 shows some possible exploitation. These results can be integrated into algorithms for computing solutions, which we consider in Section 6 below; their effectiveness is demonstrated on a practical example in Section 7.2. 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 a given action description is a solution. The complexity of these tasks strongly

[11] For similar and stronger results in classical propositional logic see [54].

$D \models C_o$ & $D \sqsubset_C D'$	**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 $\sqsubset_C$).

depends on the complexity of deciding whether a given action description satisfies a set of (obligatory) constraints (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 constraints 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 constraints considered. Note that the class of admitted constraints is the main source of complexity in most concrete settings, in particular when deciding $D \sqsubset_C D'$ reduces to deciding constraint fulfillment.

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 contains 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 [53].

Informally, the results show that modulo the cost of deciding the satisfaction of constraints 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 pre-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 pre-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 pre-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 prob-

lems with different classes (restricted sets) of constraints 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, summarizing the main arguments and constructions, while full proofs are given in Appendix A.

5.2 Constraint Fulfillment

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

Theorem 4 *Given an action description D and a set C of constraints, 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 constraints in C, and (iii) $\mathrm{P}^{\mathrm{NP}}_{\parallel}$-complete if C does not involve dynamic constraints.*

Here $\mathrm{P}^{\mathrm{NP}}_{\parallel}$ means polynomial-time with a single parallel evaluation of calls to an NP oracle. Similarly for $i > 1$, Θ^P_i is the class of problems that can be decided in polynomial time with parallel calls to a Σ^P_{i-1} oracle (alternatively, this class is often characterized by allowing $O(\log n)$ many oracle calls) [65].

Proof. Concerning *(i)* the result has been shown in [18]. Membership in Case *(iii)* follows from the fact that checking the truth of a negated universal constraint of the form ¬**ALWAYS** Q, where Q is a conjunction of clauses over static constraints of the form **holds** F or ¬**holds** F, is in NP. Hence, the complementary task, i.e., checking the truth of a positive universal constraint, **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 constraints 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^P_{k+2}-oracle calls.

As for hardness, the problem in *(iii)* is reduced to the following $\mathrm{P}^{\mathrm{NP}}_{\parallel}$-hard decision version of *Maximum CNF Satisfiability* [40]: 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:

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

$$\textbf{caused } F_{1,1} \textbf{ if } C_1, \quad \textbf{caused } \neg F_{1,1} \textbf{ if } \neg C_1,$$

$$\textbf{caused } F_{1,0} \textbf{ if } \neg C_1, \quad \textbf{caused } \neg F_{1,0} \textbf{ if } C_1,$$

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

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

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 constraint:

ALWAYS holds $F_{n,0} \vee$

SOMETIMES holds $F_{n,k} \wedge$ **ALWAYS** $(\neg$**holds** $F_{n,k+1} \wedge \ldots \wedge \neg$**holds** $F_{n,n}) \vee$

$\ldots$

SOMETIMES holds $F_{n,lk} \wedge$ **ALWAYS** $(\neg$**holds** $F_{n,lk+1} \wedge \ldots \wedge \neg$**holds** $F_{n,n})$.

For hardness in Case *(ii)*, consider m *Quantified Boolean Formulas (QBFs)* $\Phi_l = Q_1 X_1^l\, Q_2\, X_2^l \cdots Q_n X_n^l\, E^l$, $1 \leq\ l \leq m$, 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 [65]. The problem of deciding $D \models c$ for a constraint c with nesting depth k of dynamic constraints 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 constraint:

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

where

$$g^m = \textbf{SOMETIMES}\ f^m, \text{and}$$
$$f^l = p_1\, N\, p_1\, (\ldots (p_{n-1}\, N\, p_{n-1}\, \textbf{holds}\ E^l\ \textbf{after}\ \{A_{n-1}\}) \ldots)\ \textbf{after}\ \{A_1\},$$

where $N =$ **necessarily**, and 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 constraints, we aim to characterize exactly the complexity of the solution finding tasks for these classes of constraints 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 pre-solution) is (i) PSPACE-complete in general, (ii) Σ^P_{k+3}-complete, if k is the maximal nesting depth of dynamic constraints in C_o, (iii) Σ^P_2-complete, if C_o does not involve dynamic constraints, 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 $C_o = \{c_o\}$, where

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

and where $N =$ **necessarily**, and $p_i = \neg$ if i is odd and 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. □

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 constraints 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 constraint 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 constraints in C_o, (ii) Π^P_{k+3}-complete if k is the maximal nesting depth of dynamic constraints in C_o, (iii) Π^P_2-complete if C_o does not involve dynamic constraints, 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

$$f = p_1\, N\, p_1\, (\ldots (p_{n-1}\, N\, \bar{p}_{n-1}\, \textbf{holds}\ E\ \ \textbf{after}\ \{A_{n-1}\})\ldots)\ \textbf{after}\ \{A_1\},$$

$$g = \bigwedge\nolimits_{Y_i \in Y} \textbf{SOMETIMES holds}\ Y_i \wedge \textbf{SOMETIMES holds}\ \neg Y_i,$$

where $N =$ **necessarily**, $p_i = \neg$ if i is odd and 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. □

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 constraints 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 pre-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 constraints in C, (ii) Π^P_{k+3}-complete if k is the maximal nesting depth of dynamic constraints in C, (iii) Π^P_2-complete if C does not involve dynamic constraints, 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 $<_{\mathit{weight}_q}$, 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 $C_p = \{\textbf{ALWAYS holds}\ Y_i, \textbf{ALWAYS holds}\ \neg Y_i \mid Y_i \in Y\}$ and consider a weight of 1 for each $c \in C_p$. Then, D_u is a solution to $(D_u \cup D_m, I, C_o \cup C_p, <_{\mathit{weight}_q})$ 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 $C_p = \{\textbf{ALWAYS holds}\ Y_i, \textbf{ALWAYS holds}\ \neg Y_i \mid Y_i \in Y\}$ with weight 1 for each $c \in C_p$. Then, for the same reason as above, D_u is a solution to $(D_u \cup D_m, I, C_o \cup C_p, <_{\mathit{weight}_q})$ 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 $D_u = \{\textbf{caused}\ Y_1\ \textbf{if}\ \neg E, \textbf{caused}\ \neg Y_1\ \textbf{if}\ \neg E\}$, $D_m = \emptyset$, $I = \emptyset$, and $C = \emptyset$. Then, D_u is a solution to $(D_u \cup D_m, I, C, <_{\mathit{weight}_q})$ iff E is satisfiable. □

Hence, even recognizing solutions is quite hard. However, recognizing pre-solutions is easier for restricted sets of constraints (Θ^P_{k+3}-complete if the maximal nesting depth of dynamic constraints in C is k, $\mathrm{P}^{\mathrm{NP}}_{\parallel}$-complete if C has no dynamic constraints, 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 pre-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 constraints is restricted, since computing a solution needs the power of a Σ^P_{k+3} oracle. If the existence test for a pre-solution of $(D_u \cup D_m, I, C, \subset)$ in Step 1 or Step 2 in fact returns some pre-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 $<_{\mathit{weight}_q}$, 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 pre-solutions can be used effectively in a similar way. For instance, whenever the constraints in C_p can be strictly ordered according to their (non-negative) weights, such that no subset of constraints that are before a constraint 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 constraints C_p once, using the oracle to determine whether pre-solutions exist to the slightly modified problem where certain constraints from C_p are added to C_o in order to determine the set of constraints from C_p satisfied by

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 pre-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 pre-solution **then** $D' := D' \cup \{\ell\}$;
endwhile;
Step 3 output D'. □

Fig. 5. Algorithm to compute some solution preferred by set-inclusion

an optimal solution. Once this set is known, any pre-solution of the problem where these constraints are added to C_o, is a solution to the original problem.

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

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 $<_{\mathit{weight}_q}$ to a branch and bound algorithm that aims at enumerating pre-solutions (for which e.g. techniques as in [13] are useful) and cuts branches in the search tree if no better pre-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) pre-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 pre-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 Pre-Solutions

Pre-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 then amounts to determine an appropriate update fluent set, which is semantically defined and may be computed by constraint satisfaction and state set generation algorithms.

Definition 10 *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 $H_i \in \mathbf{H}$:

$$\textbf{caused } L \textbf{ if } G \wedge H_i, \tag{15}$$

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

$$\textbf{caused } L \textbf{ if } G \textbf{ after } H \wedge H_i, \tag{16}$$

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:

$$\textbf{inertial } H_i, \neg H_i. \tag{17}$$

We next define update fluent sets. To this end, we define, given an action description $D_u \cup D_m$ and a set of constraints 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 constraint 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 (9), $E, s' \models Q$ holds at some $s' \in S^U_{\mathbf{H},s}$; *(ii)* c is universal (10), $E, s' \models Q$ holds at all $s' \in S^U_{\mathbf{H},s}$; *(iii)* c is a Boolean combination of existential and universal constraints c_i, the combination evaluates to true if each c_i has the value at s wrt. $S^U_{\mathbf{H},s}$. Then, $S^U_C = \{s \in S^U \mid c \text{ holds at } s \text{ wrt. } S^U_{\mathbf{H},s}, \text{ for all } c \in C\}$. Furthermore, in the rest of this section, we identify states with the sets of fluents which are true at that state.

Definition 11 *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 pre-solution to an ADU problem $(D, I, C, \sqsubset_C)$, where $D = D_u \cup D_m$, with the algorithm PRE-SOLUTION shown in Figure 6. The key to its correctness is the following proposition.

Proposition 4 *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 pre-solution to $(D, I, C, \sqsubset_C)$ iff M is an update set for U relative to C_o.*

Algorithm PRE-SOLUTION$(D, I, C, \sqsubset_C)$
Input: an ADU problem $(D, I, C, \sqsubset_C)$
Output: some pre-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 pre-solution exists*
Step 4 identify the set W of causal laws in D_m labeled by the elements of $\mathbf{M}$;
Step 5 output $D_u \cup W \cup I$.

Fig. 6. Algorithm to compute some pre-solution

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 11 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 11 guarantees that D' is consistent. Vice versa, to every pre-solution corresponds an update set M, given by the labels of the modifiable laws included in the pre-solution.

From Proposition 4, the correctness of algorithm PRE-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* PRE-SOLUTION *outputs some pre-solution of $(D, I, C, \sqsubset_C)$ if and only if some pre-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 pre-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 pre-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:

caused $TvON$ **if** $PowerON \wedge H_1$,

caused $\neg TvON$ **if** $\neg PowerON \wedge H_2$,

inertial $H_i, \neg H_i \qquad (1 \leq i \leq 2)$.

According to the transition diagram described by U, we have that action $PushPB_{RC}$ is not executable, i.e., constraint (3): **ALWAYS executable** $\{PushPB_{RC}\}$ is violated at any state $s \supseteq \{PowerON, TvON, H_1\}$. Moreover, at any state $s \supseteq \{PowerON, TvON\}$ such that $H_2 \notin s$, constraint (13):

$$\textbf{ALWAYS holds } PowerON \wedge TvON \supset$$
$$\neg\textbf{necessarily } (\textbf{holds } TvON) \textbf{ after } \{PushPB_{TV}\}$$

is not satisfied due to missing causation for $\neg TvON$. At every state of U, however, constraint (14): **ALWAYS executable** $\{PushPB_{TV}\}$ is satisfied. We thus obtain

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

and, for instance, $\{PowerON, TvON, H_2\} \in S^U_C$. 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 **caused** $TvON$ **if** $PowerON$, which is labeled by H_1, from $D \cup I$, we obtain a pre-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:

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

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:

$$\textbf{caused } PowerON \textbf{ after } PushPB_{TV} \wedge \neg PowerON \wedge H_3,$$
$$\textbf{caused } \neg PowerON \textbf{ after } PushPB_{TV} \wedge PowerON \wedge H_4,$$
$$\textbf{caused } TvON \textbf{ after } PushPB_{TV} \wedge \neg PowerON \wedge H_5,$$
$$\textbf{caused } \neg TvON \textbf{ after } PushPB_{TV} \wedge PowerON \wedge H_6,$$
$$\textbf{inertial } H_i, \neg H_i \qquad (3 \leq i \leq 6).$$

Constraint (3): **ALWAYS executable** $\{PushPB_{RC}\}$ is still violated according to the transition diagram described by U, since the action $PushPB_{RC}$ is not executable whenever $s \supseteq \{PowerON, TvON, H_1\}$. Let us consider the remaining states s of U, i.e., only those such that $H_1 \notin s$. We first observe that a violation of constraint

(13):

ALWAYS holds $PowerON \wedge TvON \supset$

$\neg$**necessarily** (**holds** $TvON$) **after** $\{PushPB_{TV}\}$

is witnessed by any such state where $s \supseteq \{PowerON, TvON\}$, $H_6 \notin s$, and either $H_2 \notin s$ or $H_4 \notin s$ (or both), since there is no causation for $\neg TvON$ when executing $PushPB_{TV}$. Finally, constraint (14): **ALWAYS executable** $\{PushPB_{TV}\}$ does not hold at any such state s where the power and the TV are off, i.e., $s \cap \{PowerON, TvON\} = \emptyset$, 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 pre-solution to the problem. Note however, that in case of $\sqsubset_C = \subset$, for instance, none of the pre-solutions is a solution, as removing **caused** $TvON$ **if** $PowerON$ is sufficient. This is reflected by the (maximal) update set $\{H_2, H_3, H_4, H_5, H_6\}$. □

Algorithm PRE-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 constraint fulfillment check in Step 1, the bulk of the work is with Step 3, i.e., to compute an update set **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 **M** for which condition *(ii)* $S^U_{\mathbf{H},s} \subseteq S^U_C$ is tested using constraint satisfaction; 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 constraint c of form (9), define $c_s =$ **SOMETIMES holds** $F_s \wedge Q$, and for each universal constraint c of form (10), define $c_s =$ **ALWAYS holds** $F_s \supset Q$. For a Boolean combination c of existential and universal constraints, we define c_s as the constraint obtained by rewriting each occurrence of an existential or universal constraint as described above. Then $S^U_{\mathbf{H},s} \subseteq S^U_C$ is equivalent to $U \models c_s$ for each constraint c in C.

Thus, one can build algorithms to compute pre-solutions of an ADU on top of basic reasoning services for action descriptions that generate sets of states and allow for checking the satisfaction of constraints (as supported e.g. in AD-Constraint [21], under some limitations), which are applied to the update description $U(D)$. Compared to a simple search over the pre-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 effi-

cient processing of transition-based formalisms), and perform state generation and check constraint fulfillment 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 (historyless) updates under lazy evaluation, and more generally for realizing non-Markovian semantics of sequences of updates $I_1, \ldots, I_k$, in analogy to update programs in the context of logic programming updates [5,20]. In the Markovian case, the result of updating an action description D is obtained by incorporating the I_i, $i = 1, \ldots, k$ 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 Examples: Updating the Zoo World into a Circus

The Zoo World is an action domain proposed by Erik Sandewall in his Logic Modelling Workshop. It consists of several cages and the exterior, gates between them, and animals of several species, including humans. Actions in this domain include moving within and between cages, opening and closing gates, and mounting and riding animals. This domain was described in the action language $\mathcal{C}+$ in [1].

We present two examples for updating the action description of the Zoo World in $\mathcal{C}$ (derived from the one in [1]) such that we obtain a description for a Circus. The first example illustrates the applicability of our method for computing pre-solutions; the second example illustrates the usefulness of the decomposability theorem.

7.1 Singing and Mounting in the Circus

Suppose that we would like to update the action description of the Zoo World in $\mathcal{C}$ in such a way to obtain a description for a Circus by taking into account the following new information: a human can sing; and when he does, he becomes happy if he is also mounted on an animal. We also want to ensure the following condition: different from the Zoo World, in a Circus, the humans are expected to mount on each other, who further can mount on a large animal.

First, we transform the description of [1] into the action language $\mathcal{C}$; the modified

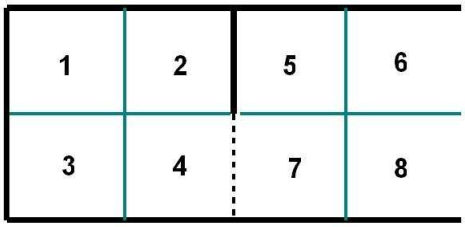

Fig. 7. The landscape of the little zoo of [1]: positions 1–4 are inside the cage; positions 5–8 are outside the cage, the dashed lines denote the gate.

description is available in Appendix C.

Next, we describe the new information I by the following causal laws. Suppose that h ranges over constants denoting humans, and $anml$ ranges over constants denoting animals in the zoo:

$$\textbf{caused } Happy(h) \textbf{ if } True \textbf{ after } Sing(h) \wedge Mounted(h, anml).$$

Note that both h and $anml$ are schematic variables; so the above expression stands for a set of "ground" causal laws.

Next, we identify the causal laws D_m that could be modified. The modifications we desire are about the mounting action in particular, so let D_m consist of the following causal laws. Suppose that $h, h1$ range over constants denoting humans, $anml$ ranges over constants denoting animals in the zoo, and p ranges over positions in the zoo.

- If a human tries to mount an animal that doesn't change position, mounting is successful:

$$\textbf{caused } Mounted(h, anml) \textbf{ if } Pos(anml, p) \textbf{ after } Pos(anml, p) \wedge Mount(h, anml). \tag{18}$$

- A human cannot attempt to mount a human who is mounted:

$$\textbf{caused } False \textbf{ if } True \textbf{ after } Mount(h, h1) \wedge Mounted(h1, anml). \tag{19}$$

- A human cannot be mounted on a human who is mounted:

$$\textbf{caused } False \textbf{ if } Mounted(h, h1) \wedge Mounted(h1, anml). \tag{20}$$

We assume that our little Circus has two humans (a small boy named Bart and an adult named Homer) and an elephant (Jumbo). We assume that our Circus has the same landscape as the little Zoo as in [1]: there is a cage, with four positions inside; outside the cage are four positions as well (Figure 7).

We can express the desired conditions (or scenarios) in this little Circus by constraints. For instance, consider the following scenario of three steps: Initially, Jumbo

and Bart are at different positions in the cage, and Homer is outside the cage; Homer is not happy. It should be possible at some point that first Homer mounts on Jumbo and next Bart mounts on Homer, so that in the end Homer is mounted on Jumbo, Bart is mounted on Homer, and Homer is happy. Suppose also that Jumbo does not change its location during the whole scenario. We can describe this scenario by the following constraint C:

SOMETIMES

$\bigvee_{l \neq l'', l, l'' < 5, l' > 4}$ **holds** $Pos(Bart, l) \wedge Pos(Homer, l') \wedge Pos(Jumbo, l'') \wedge$

$\neg Happy(Homer) \wedge$

(**possibly** $Mounted(Bart, Homer) \wedge Mounted(Homer, Jumbo) \wedge$

$Happy(Homer)$ **after** $Mount(Homer, Jumbo); Mount(Bart, Homer) \vee$

possibly $Mounted(Bart, Homer) \wedge Mounted(Homer, Jumbo) \wedge$

$Happy(Homer)$ **after** $True; Mount(Homer, Jumbo); Mount(Bart, Homer)) \wedge$

$\bigvee_{l'' < 5}$ **evolves** $Pos(Jumbo, l''); True; Pos(Jumbo, l''); True; Pos(Jumbo, l'');$

$True; Pos(Jumbo, l''); True; Pos(Jumbo, l''); True; Pos(Jumbo, l''); True;$

$Pos(Jumbo, l''); True; Pos(Jumbo, l'')$

(21)

We can present this constraint to CCALC as (as in Figure C.7 of Appendix C); and CCALC finds out that this scenario is not possible within the Zoo World.

Let us find a pre-solution D' to the ADU problem $(D, I, C, \subset)$, by applying Algorithm 6. For that, first we construct the update description U of the Zoo World:

(1) we introduce update fluents as auxiliary fluents of the following three forms $Aux1(h, anml, p)$, $Aux2(h, h1, anml)$, and $Aux3(h, h1, anml)$.
(2) we add new causal laws to make them inertial

inertial $Aux1(h, anml, p), Aux2(h, h1, anml), Aux3(h, h1, anml)$

inertial $\neg Aux1(h, anml, p), \neg Aux2(h, h1, anml), \neg Aux3(h, h1, anml)$

(3) we replace the causal laws (18)–(20) with the following causal laws respec-

tively

caused $Mounted(h, anml)$ **if** $Pos(anml, p)$ **after** $Pos(anml, p) \wedge$
$Mount(h, anml) \wedge Aux1(h, anml, p)$

caused $False$ **if** $True$ **after** $Mount(h, h1) \wedge Mounted(h1, anml) \wedge$
$Aux2(h, h1, anml)$

caused $False$ **if** $Mounted(h, h1) \wedge Mounted(h1, anml) \wedge$
$Aux3(h, h1, anml)$

After that, we can check whether the scenario represented by the constraint (21) is possible if we keep all the causal laws, except for those labeled by $Aux2(Bart, Homer, Jumbo)$ and $Aux3(Bart, Homer, Jumbo)$. For that, we just need to modify the CCALC constraint above by adding several lines, as shown in Figure C.8 of Appendix C. Then, CCALC finds a possible execution of this scenario as presented in Figure C.9 of Appendix C. It suggests dropping from D_m the causal laws

caused $False$ **if** $True$ **after** $Mount(Bart, Homer) \wedge Mounted(Homer, Jumbo)$

caused $False$ **if** $Mounted(Bart, Homer) \wedge Mounted(Homer, Jumbo)$

to update the Zoo World description into a little Circus.

7.2 Exchanging Hats in the Circus

Consider a world, which involves monkeys and dogs among other animals, where only monkeys can wear hats. We can obtain a $\mathcal{C}$ description D^0 of such a world, from the $\mathcal{C}+$ description of missionaries and cannibals exchanging hats [50]; it can be presented to CCALC as in Figure D.1 (Appendix D).

Now consider a variation of the Zoo World described in Section 7.1, which involves also monkeys and dogs, where only monkeys can wear hats. This variation of the Zoo World can be described by the union of the Zoo World description D^1 discussed in Section 7.1 (Figures C.1–C.6, Appendix C) and the description D^0 mentioned above.

Suppose that we would like to update the action description $D^0 \cup D^1$ of this extended Zoo World, to obtain a description of a Circus where not only humans can mount on each other who further can mount on a large animal, but also animals can exchange hats with each other. Assume that the modifiable part D^1_m of D^1 is the same as in Section 7.1, and the modifiable part of D^0_m of D^0 consists of the

following causal laws:

$$\textbf{caused } \mathit{False} \textbf{ if } \mathit{Owner}(\mathit{ha}, \mathit{anml})$$

where ha ranges over hats, and $anml$ ranges over animals except monkeys.

We assume that our little Circus has the same landscape as in Figure 7; and it contains two humans (a small boy Bart and an adult Homer), an elephant (Jumbo), a dog (Snoopy), three monkeys (a small monkey Abu and two large monkeys), and two hats. In this little Circus, in addition to the desired conditions (or scenarios) presented in Section 7.1 by the set C^1 of constraints (21), we also consider the following scenario: initially, Snoopy and Abu are wearing hats; they exchange hats at least once. We can express this condition by the constraints C^0:

SOMETIMES

($\textbf{evolves } \mathit{Owner}(\mathit{ha1}, \mathit{Abu}) \wedge \mathit{Owner}(\mathit{ha2}, \mathit{Snoopy}); \mathit{exchange}(\mathit{ha1}, \mathit{ha2}); \mathit{True} \vee$

$\textbf{evolves } \mathit{Owner}(\mathit{ha1}, \mathit{Abu}) \wedge \mathit{Owner}(\mathit{ha2}, \mathit{Snoopy}); \mathit{True}; \mathit{True};$

$\mathit{exchange}(\mathit{ha1}, \mathit{ha2}); \mathit{True} \vee$

$\textbf{evolves } \mathit{Owner}(\mathit{ha1}, \mathit{Abu}) \wedge \mathit{Owner}(\mathit{ha2}, \mathit{Snoopy}); \mathit{True}; \mathit{True};$

$\mathit{True}; \mathit{True}; \mathit{True}; \mathit{exchange}(\mathit{ha1}, \mathit{ha2}); \mathit{True}).$

where $ha1$ and $ha2$ range over hats. This constraint can be presented to CCALC as in Figure D.2 (Appendix D).

Here, we can update $D^0 \cup D^1$ relative to $C^0 \cup C^1$. On the other hand, since $((D^0, \emptyset, C^0), (D^1, \emptyset, C^1))$ is a near-decomposition of $(D^0 \cup D^1, \emptyset, C^0 \cup C^1, \subset)$, by Theorem 1, we can update D^0 and D^1 separately, in parallel. Considering the computation time CCALC takes to verify given constraints, the latter approach takes much less time. With the former approach, CCALC verifies constraints $C^0 \cup C^1$ with respect to a propositional theory of size 20450 atoms and 398430 clauses (obtained from the update description of $D^0 \cup D^1$) in about 9 minutes (including the grounding and completion time). With the latter approach, CCALC verifies C^0 with respect to a propositional theory of size 164 atoms and 766 clauses (obtained from the update description of D^0) in less than a second (including the grounding and completion time); and it verifies C^1 with respect to a propositional theory of size 5462 atoms and 60567 clauses (obtained from the update description of D^1) in less than 30 seconds (including the grounding and completion time).

8 Discussion

8.1 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, see e.g. [67,34,57] and references therein. The relation of this problem to reasoning about actions has been identified earlier [66,59,56], 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 [61] is similar to our work in that it also studies update problems in a nonmonotonic framework (yet in logic programming) and considers the same criterion of minimal change. It deals with three kinds of updates to a knowledge base D: theory update of D by some new information 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 [61] that checking for solution existence is NP-hard for each problem; this complies with Theorem 5 *(iii)*. An important difference to [61] is that in an ADU problem $(D, I, C, \subset)$, the constraints 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 [44] and Liberatore [47] 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 constraint language by formulas of the shapes

$$E \textbf{ occurs at } t_i, \tag{22}$$
$$P \textbf{ holds at } t_i, \tag{23}$$

where E is an action name, P is a fluent name, and t_i is a time stamp; a state s satisfies a constraint (22) resp. (23) if, for some history (11) 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 [68]. 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 constraint language discussed above, other forms of consistency studied in [68] can be achieved in our framework, by describing possible scenarios or formulas as constraints.

Some of the related work mentioned above, like [6,49,3,37], 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? [37] studies the update problem in the context of dynamic logic [35]. 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., [38,62]), 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. A respective elaboration is described by a causal law, like $Blackout \rightarrow [Toggle]Light$, which is to be contracted from the action description. 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 ϕ does not hold: first a still has the effect ψ; and second a has no effect except on 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 [37] 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 constraints (which may describe some de-

sired/preferred conditions on the domain) are satisfied by the updated description. In other words, [37] is less suitable for the incorporation of new information compared to our approach. For instance in the example given above, elaborating wrt. the effect law $Blackout \rightarrow [ReplaceFuse]Blackout$ will not serve the intended purpose to incorporate the effects of replacing a broken fuse, while in our approach we simply update with the causal law **caused** $\neg Blackout$ **after** $ReplaceFuse \wedge Blackout$ for this purpose.

Another related work that studies action description updates, for elaboration tolerance, is [3]. The authors introduce an action description language, called Evolp Action Programs (EAPs), built upon the update language Evolp [4]. 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 and the switch is toggled at step n, 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. The semantics of an EAP (and thus the $assert$ construct) is given by means of stable models [29]. Adding $assert(D)$ to the initial description is different from adding D: like our approach it ensures static consistency of the resulting action description (if the update itself is consistent); preference is implicitly given by set inclusion, i.e., maximal consistent subsets of the initial laws are retained. Another similarity to our work is that updates that consist of static/dynamic rules are described in the same language as the action description. Additionally, the language of [3] allows to specify changes of rules, as a part of an update (using the assert construct). 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. However, apart from rather cumbersome language extensions for handling the frame problem, EAPs do not provide a means to specify certain dynamic requirements that an update might have to satisfy (in particular universal properties quantifying over all states), which is a main feature of our approach. For instance, in the setting of Example 1, translating D and I into a respective EAP would represent an update equivalent to the action description $D \cup I$, i.e., one that does not satisfy the constraints C. Since the constraints cannot be expressed in the language, additional analysis is needed to identify an update I' (different from I), which would enforce the required behaviour when asserted to the initial action description D.

The works by Lifschitz [49] and by Balduccini and Gelfond [6] are similar to [3] 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 [52,55]. Lifschitz describes in [49] 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 [6], 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 (no causal laws are modified). The approaches provide tools for the user to enact updates (by defeating causal laws, respectively by applying consistency restoring rules), but different from our approach, no particular modifications are characterized from first principles as "intended" solutions of an update problem, which remains with the user. While adding abnormality predicates [49] is a simple technique that does not support preference constraints, [6] (which is more geared towards diagnosis) requires to anticipate all possible updates in order to encode a priori solutions for potential inconsistencies with subsequent updates into the initial domain description at design time; the support for preferences on consistency restoring rules is limited, e.g., cardinality based preferences are difficult to represent. Furthermore, as the result of updating an action description is not an action description, adjustments for iterated updates are necessary.

Concerning results on the computational complexity, Eiter and Gottlob [23] 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 [36] improved these complexity bounds for restricted settings under Winslett's Possible Models Approach. Liberatore [46] considers further approaches for belief update from the literature, derived corresponding complexity results, and extended them to the problem of iterated update. Baral and Zhang [7] 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 [45] establishes, for instance, NP-completeness of consistency checking and coNP-completeness for entailment, which essentially amounts to checking whether $D \models$ **ALWAYS necessarily** (**holds** F) **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. [42] investigated the computational complexity of the progression problem 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 constraints c in our setting. Moreover, to the best of our knowledge, the complexity of deciding constraint fulfillment has not been addressed so far (apart from the PSPACE result for the general case for the constraint language we considered, which has been proven in [18]), let alone the problem of updating action descriptions in the presence of constraints.

8.2 Nature of Change

As stated in the problem description, our approach is intended to also allow for designed (normative) worlds that are represented by means of action descriptions, where changes thus are considered to be updates rather than revisions. However, 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 [41,58]. Lang [41] describes a scope for revision and for update, and he notices that, as pointed out by [27,28], the scope can not be simply decided by whether the theory is about static vs. dynamic worlds. Then, as also pointed out by [10,14], Lang relates revision and update by means of backward-forward reasoning, in particular, by means of action progression. According to [41], 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.

However, update and revision behave for our problem technically not much different: while informally, revision operators aim at selecting models of the new information that are closest to the models of the knowledge base globally, update operators change each model of the knowledge base locally (this is intuitively captured by the axiom **U8** of the KM postulates). As each action description D has a unique associated model given by $T(D)$, the two methods yield the same result. The main difference remaining is the behavior on inconsistent action descriptions. Revision with consistent information must make an inconsistent knowledge base consistent

(as done in our approach), while update must preserve inconsistency. Clearly, our method can be easily adapted to this behavior, and thus show an update flavor.

The AGM and KM postulates [2,39] are based on several assumptions that do not hold for the action language $\mathcal{C}$. One requirement which is not met is that of an underlying formal language which is governed by a logic, i.e., which is closed under Boolean connectives. Other requirements, including superclassicality, modus ponens, and the deduction theorem, essentially restrict to formalisms with an underlying monotonic logic (an explicit restriction for instance in Hansson's work [33]). However, the action language $\mathcal{C}$ is is nonmonotonic. For instance, if D consists of the single law

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

where P is the single fluent, then the transition diagram described by D, $T(D)$, has two states $s_1 = \{P\}$, $s_2 = \{\neg P\}$, and two transitions $\langle s_1, \emptyset, s_1\rangle$ and $\langle s_2, \emptyset, s_1\rangle$. Thus the causal law **caused** P **after** $\emptyset$ is satisfied by $T(D)$ (equivalently, $D \models$ **ALWAYS necessarily holds** P **after** $\emptyset$), 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 additional transitions ($\langle s_1, \emptyset, s_2\rangle$ and $\langle s_2, \emptyset, s_2\rangle$) and $D \not\models$ **ALWAYS necessarily holds** P **after** $\emptyset$; thus **caused** P **after** $\emptyset$ 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 [20] and for defeasible logic in [9]. This has been confirmed by our study of KM-style properties in Section 4, where nonmonotonicity turned out to be the reason for several properties to fail. Thus governing our action description updates with the AGM or KM postulates is not meaningful; and intuitively the same is true for postulates for contraction developed in monotonic settings. We refrained from a formal investigation in this direction due to another reason however: action language $\mathcal{C}$ is not closed under complement, more precisely it is neither defined nor clear what the complement of an action language should be, or how it is represented. As a consequence, it does not constitute a logic and well-known identities, like the Levi Identity used in classical belief change settings to relate contraction, expansion, and revision, cannot be applied.

By the counterexamples for KM postulates given in Section 4, it also becomes clear that the same results are obtained for less general, alternative definitions. For instance, one may consider an initial action description D_0, and a set of constraints C_0, as the initial knowledge to be modified by new information, which consists of a set of causal laws D_1, and a set of constraints C_1, which are considered to hold for sure in a solution. Preference is given to solutions that keep a maximal sets of the original laws and constraints (wrt. set inclusion), such that the resulting action description is consistent and satisfies—in addition to all constraints in C_1—also all constraints from C_0 that are kept. Note that in our setting, this is amounts to a

particular case where $D = D_m = D_0$, $C_o = C_1$, $C_p = C_0$, and $\sqsubset_C$ is defined by $D' \sqsubset_C D''$ iff $D' \subseteq D''$, $D'_{C_p} \subseteq D''_{C_p}$, and one of the inclusions is strict, where $D_{C_p} = \{c \in C_p \mid D \models c\}$, for any action description D. Note that all counterexamples stated in Section 4 are also counterexamples for this setting. We further remark that the other properties (except for those that require strongly minimizing $\sqsubset_C$, which is not the case for the above preference relation), and in particular results on computational complexity, hold for this particular setting as well.

An AGM- respectively KM-style theory for non-monotonic logics with significant attention is, to our knowledge, still missing. We note that [38], 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. Another notable work is [26], which considers the revision of rational preference orderings that underly certain (nonmonotonic) consequence operators. However, in order to avoid shortcomings concerning the general principles of success and minimality of change, which are impossible to adhere in general for the nonmonotonic setting, restrictions are imposed concerning the knowledge bases and the conditionals (akin to laws in our setting) admissible for revision. More closely related to our setting is a very recent approach to belief revision for answer-set programs [15] with an operator that satisfies the majority of the AGM postulates. This is achieved by building on a strong underlying notion of equivalence (so-called *strong equivalence*), using a respective monotonic formal characterization of answer-set programs called SE-models, and by applying well-known techniques from classical belief revision. Applying similar methods to action language $\mathcal{C}$ in order to come up with a theory-revision operator is an interesting subject for future work. Work by Turner [63] on SE-models for causal theories may serve as a starting point. However, several issues are not immediate and need further consideration. For instance, a direct application of Turner's SE-models to laws in $\mathcal{C}$ is achieved for static laws only, while it is the dynamic laws which we are mainly interested in for revision. Hence, the concept of SE-model has to be adapted appropriately. Note that any revision operator, respectively update operator, obtained this way is characterized by semantic structures which is orthogonal to our aims in this article. It is not clear how the resulting semantic structures could be syntactically represented (something which could be achieved due to a characterization of SE-models in terms of answer-set programs in [15]). Even if a suitable representation by means of causal laws is developed, it is not likely that the resulting action description after the change is reminiscent of the original description (see also discussions in [17] and comments on this work in the following subsection).

We can sometimes improve solutions (and pre-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 pre-solution D' to $(D, I, C, \sqsubset_C)$ there exists some pre-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 $\sqsubset_C^s$, the solutions of $(D, I, C, \sqsubset_C)$ can then be recovered from the ones of $(D^s, I, C, \sqsubset_C^s)$. Therefore, such a replacement method can be useful to prevent "complete removal" of some laws from the given action description. Furthermore, solutions of $(D^s, I, C, \sqsubset_C^s)$ 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, [17] took a slightly different, semantics-oriented view for resolving conflicts between an action description and a set of constraints, 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 constraints. 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 [17] to propose support for the user in terms of reasoning services on an action description given constraints, 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 [21,22]. 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.

9 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 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 constraints C (expressed as formulas 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 constraints and preference relations. Furthermore, we presented some algorithms for computing solutions and pre-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 systems like the causal calculator [51] or AD-Constraint [21], 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 [30]). 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 $\mathcal{C}+$ [43], or the action language $\mathcal{K}$ [19] (into which the language considered here maps naturally) may be studied.

Further issues are to consider richer forms of constraints (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 [8,11], or to semantic-based preference that uses other weight assignments like those in [18] (which are computable in polynomial space) or preference based on state- and transition-rankings, inspired by approaches e.g. in conditional reasoning (see [24]).

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 [5,20], 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.

Acknowledgments

We would like to thank the anonymous referees for their comments, which helped improve this paper considerably.

This work was supported by the Austrian Science Fund (FWF) grant P16536-N04, the European Commission IST programme grants FET-2001-37004 WASP and IST-2001-33123 CologNeT, and by the Vienna Science and Technology Fund (WWTF) grant ICT08-020.

References

[1] V. Akman, S. T. Erdogan, J. Lee, V. Lifschitz, and H. Turner. Representing the zoo world and the traffic world in the language of the causal calculator. *Artificial Intelligence*, 153(1–2):105–140, 2004.

[2] 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.

[3] 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 *Lecture Notes in Computer Science*, pages 52–77. Springer, 2004.

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

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

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

[11] 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.

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

[13] 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.

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

[15] J. P. Delgrande, T. Schaub, H. Tompits, and S. Woltran. Merging logic programs under answer set semantics. In P. M. Hill and D. S. Warren, editors, *ICLP*, volume 5649 of *Lecture Notes in Computer Science*, pages 160–174. Springer, 2009.

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

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

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

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

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

[25] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 995–1072. Elsevier, 1990.

[26] M. Freund. On the revision of preferences and rational inference processes. *Artificial Intelligence*, 152(1):105–137, 2004.

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

[28] 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.

[29] 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.

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

[31] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153(1-2):49–104, 2004.

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

[33] S. O. Hansson. Knowledge-level analysis of belief base operations. *Artificial Intelligence*, 82(1-2):215–235, 1996.

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

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

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

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

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

[39] 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.

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

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

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

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

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

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

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

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

[48] V. Lifschitz. Answer Set Planning. In D. D. Schreye, editor, *Proceedings of the 16th International Conference on Logic Programming (ICLP'99)*, pages 23–37, Las Cruces, New Mexico, USA, Nov. 1999. The MIT Press.

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

[50] V. Lifschitz and W. Ren. Irrelevant actions in plan generation (extended abstract). In *Proc. Ninth Ibero-American Workshops on Artificial Intelligence (IBERAMIA 2004)*, pages 71–78, 2004.

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

[52] J. McCarthy. Elaboration tolerance. In *Proc. 1998 Symposium on Logical Formalizations of Commonsense Reasoning (CommonSense 98), Jan 7-9, 1998, London*, pages 198–216, 1998. Available at `www.ida.liu.se/ext/etai/nj/fcs-98/198/paper.ps` (accessed Jun 3, 2010).

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

[54] R. Parikh. Beliefs, belief revision, and splitting languages. *Journal of Logic, Language and Information*, 2:266–278, 1999.

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

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

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

[62] 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.

[63] H. Turner. Strong equivalence for causal theories. In V. Lifschitz and I. Niemelä, editors, *LPNMR*, volume 2923 of *Lecture Notes in Computer Science*, pages 289–301. Springer, 2004.

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

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

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

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

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

Electronic Appendix

A Proofs for Section 5

Theorem 4 *Given an action description D and a set C of constraints, 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 constraints in C, and (iii) $\mathrm{P}^{\mathrm{NP}}_{\parallel}$-complete if C does not involve dynamic constraints.*

Proof. Concerning *(i)* the result has been shown in [18]. We proceed with the proof of *(ii)* and *(iii)*.

Membership: W.l.o.g. C contains a single constraint c. Let us consider *(iii)* first. Then, c is a conjunction of clauses over universal constraints of the following form: **ALWAYS** Q or ¬**ALWAYS** Q, where Q is a conjunction of clauses over static constraints of the form **holds** F or ¬**holds** F. Checking truth of a negated universal (sub-)constraint of this form is in NP. To do so, 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 constraints is satisfied at s). Hence, the complementary task, i.e., checking the truth of a positive universal constraint, **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 constraints in c. This proves $\mathrm{P}^{\mathrm{NP}}_{\parallel}$-membership.

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

Base case ($k = 0$): For static Q, by *(iii)* deciding $D \models c_i$, is in $\mathrm{P}^{\mathrm{NP}}_{\parallel}$, for $1 \leq i \leq 4$. Hence, let $Q =$ **necessarily** Q_{k-1} **after** $A_1; \ldots; A_n$ be a dynamic constraint. 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 constraints, witnessing $D \not\models c_1$. Deciding $D \models c_3$ is in Π^P_2 and the complementary problem $D \models c_4$ is in Σ^P_2. 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 constraint 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 constraints be $k > 0$, and assume that deciding $D \models Q_{k-1}$ is in Θ_{k+2}^P for any subconstraint 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 constraint $Q \in c$. 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:

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

$$\textbf{caused } F_{1,1} \textbf{ if } C_1, \qquad \textbf{caused } \neg F_{1,1} \textbf{ if } \neg C_1,$$

$$\textbf{caused } F_{1,0} \textbf{ if } \neg C_1, \quad \textbf{caused } \neg F_{1,0} \textbf{ if } C_1,$$

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

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

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 constraint c_k:

ALWAYS holds $F_{n,0} \vee$

SOMETIMES holds $F_{n,k} \wedge$ **ALWAYS** ($\neg$**holds** $F_{n,k+1} \wedge \ldots \wedge \neg$**holds** $F_{n,n}) \vee$

$\ldots$

SOMETIMES holds $F_{n,lk} \wedge$ **ALWAYS** ($\neg$**holds** $F_{n,lk+1} \wedge \ldots \wedge \neg$**holds** $F_{n,n}$),

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 satisfied 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 constraints 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 satisfied 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 $\Phi_l = Q_1 X_1^l \, Q_2 \, X_2^l \cdots Q_n X_n^l \, E^l$, $1 \leq l \leq m$, 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 constraint c with nesting depth k of dynamic constraints 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 $X_{i+1}^1 \cup \cdots \cup X_{i+1}^m$.

Consider the constraint:

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

where

$$g^m = \textbf{SOMETIMES } f^m, \text{and}$$
$$f^l = p_1\, N\, p_1\, (\ldots (p_{n-1}\, N\, p_{n-1} \textbf{ holds } E^l \textbf{ after } \{A_{n-1}\}) \ldots) \textbf{ after } \{A_1\},$$

where $N =$ **necessarily**, and where $p_i = \neg$ if i is even and p_i is 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_1^l 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 $X_1^l \cup \cdots \cup X_{n-1}^l$. 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_n^l will turn the assignment on $X_1^l \cup \cdots \cup X_{n-1}^l$ 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_n^l that,

together with the assignment on $X_1^l \cup \cdots \cup X_{n-1}^l$ 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_1^l, 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 $X_1^l \cup \cdots \cup X_{n-1}^l$, 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_n^l will turn the assignment on $X_1^l \cup \cdots \cup X_{n-1}^l$ 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_n^l that will turn the assignment on $X_1^l \cup \cdots \cup X_{n-1}^l$ 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 $D_2 \models g^m$, 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 satisfied, whereas f^{2l+2} is not satisfied 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_{n+1}^P-hardness, i.e., Θ_{k+3}^P-hardness. □

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

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

and where $N =$ **necessarily**, and $p_i = \neg$ if i is odd and 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 pre-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 pre-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 pre-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 constraints in C_o, (ii) Π^P_{k+3}-complete if k is the maximal nesting depth of dynamic constraints in C_o, (iii) Π^P_2-complete if C_o does not involve dynamic constraints, 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

$$f = p_1\, N\, p_1\, (\ldots (p_{n-1}\, N\, \bar{p}_{n-1}\ \textbf{holds}\ E\ \ \textbf{after}\ \{A_{n-1}\}) \ldots)\ \textbf{after}\ \{A_1\},$$

$$g = \bigwedge\nolimits_{Y_i \in Y} \textbf{SOMETIMES}\ \ \textbf{holds}\ Y_i \wedge \textbf{SOMETIMES}\ \ \textbf{holds}\ \neg Y_i,$$

where $N =$ **necessarily**, $p_i = \neg$ if i is odd and 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 pre-solution. We show that it is a maximum pre-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 pre-solution. This contradicts the maximality of D_u.

For the if-direction, towards a contradiction assume that D_u is not maximal. Then, all states 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 pre-solution. We show that it is a maximum pre-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 pre-solution, which contradicts the maximality of D_u.

For the if-direction, towards a contradiction assume that D_u is not maximal. Then, all states 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 constraints in C, (ii) Π^P_{k+3}-complete if k is the maximal nesting depth of dynamic constraints in C, (iii) Π^P_2-complete if C does not involve dynamic constraints, 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 constraints 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 $C_p = \{$**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, $D_u <_{weight_q} D'$ 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 constraints 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 $C_p = \{$**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, $D_u <_{weight_q} D'$ 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 $D_u = \{$**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.

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 4, 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 3 *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 (15) 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 (15) in U such that $H_i \notin \mathbf{M}$. 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 (15) in U. Since, for every corresponding static law (15) 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 (17), 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 (16) in U, that is not of the form (17), 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 (17) 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 (17), 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 (16) 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 (17), 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 4 *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 pre-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 pre-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 pre-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 3 *(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 3 *(i)*, $s \cup \mathbf{M}$ is in S^U.

We show for any static or dynamic constraint 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 constraint c first. Since no element of $\mathbf{H}$ appears in c, and the constraint is static, $s \cup \mathbf{M} \not\models c$ follows. However, this contradicts the assumption. So let c be a dynamic constraint and h a history (11) 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 constraint and, since no element of $\mathbf{H}$ appears in c, it follows that $s_n \cup \mathbf{M} \not\models Q$. Moreover, by Lemma 3 *(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 constraints with maximum nesting depth $k-1$, and consider a dynamic constraint of nesting depth k. Then, Q contains only static constraints and dynamic constraints 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 3 *(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 constraint c, and thus also for any open constraint c.

We continue considering existential and universal constraints 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 open constraint 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 3 *(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 open, it follows that $D', s' \setminus \mathbf{H} \models Q$, and hence $D' \models c$. So let c be a universally quantified open constraint 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 3 *(i)* $s' \cup \mathbf{M} \in S^U$. Moreover, since Q is open 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 constraint c; the same follows for any Boolean combination of existential and universal constraints. 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 constraint 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 pre-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 3 *(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 constraint 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 constraint c first. Since no element of $\mathbf{H}$ appears in c, and the constraint is static, $s \not\models c$ follows. However, this contradicts the assumption $D', s \models c$. So let c be a dynamic constraint and h^U a history (11) 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 constraint and, since no element of $\mathbf{H}$ appears in c, it follows that $s_n \setminus \mathbf{H} \not\models Q$. Furthermore, by Lemma 3 *(ii)*, $s_i =_{\mathbf{H}} s_0$ for $1 \leq i \leq n$. Therefore, by Lemma 3 *(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 constraints with maximum nesting depth $k-1$, and consider a dynamic constraint of nesting depth k. Then, Q contains only static constraints and dynamic constraints 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 3 *(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 constraint c, and thus also for any open constraint c.

We continue considering existential and universal constraints 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 open constraint Q, the claim follows from the fact that then there exists a state $s' \in S$, such that $D', s' \models Q$. By Lemma 3 *(i)* $s' \cup \mathbf{M}$ is a state in S^U, and since Q is open, 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 open constraint 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 3 *(i)* $s' \setminus \mathbf{M}$ is a state of D', and since Q is open, $D', s \not\models Q$ 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 constraint c; the same follows for any Boolean combination of existential and universal constraints. This proves that $D' \models c$ implies that c holds at s wrt. $S^U_{\mathbf{H},s}$.

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

C The Zoo World in $\mathcal{C}$

The Zoo World was described in the action language $\mathcal{C}+$ and presented in the language of CCALC in [1], in five parts: animals (`zoo-animals`), movement (`zoo-movement`), actions (`zoo-actions`), landscape (`zoo-landscape`), as well as their union (`zoo`). We have transformed the first three components into $\mathcal{C}$ (in the language of CCALC as well) [12] as shown in Figures C.1–C.6, by replacing the non-Boolean fluents of the form `pos(Animal) = Position` with Boolean fluents `pos(Animal,Position)`, and by adding several causal laws to make sure that they express the same fluent: [13]

```
constraint [\/P | pos(ANML,P)].
caused -pos(ANML,P) if  pos(ANML,P1) & P\=P1.
```

The first three forms of causal laws in Figure C.6 constitute the modifiable part D_m of this description. The sample constraint given in Section 7.1 can be represented by the CCALC query given in Figure C.7.

D Exchanging Hats in the Circus

Consider a Circus world including monkeys and dogs, where only monkeys can exchange hats with each other. We can obtain a $\mathcal{C}$ description of this world, from the $\mathcal{C}+$ description of missionaries and cannibals exchanging hats [50], and present it to CCALC as in Figure D.1.

Now consider a variation of the Zoo World described in Section 7.1, which involves also monkeys and dogs, where only monkeys can exchange hats. This variation of the Zoo World can be described by the union of the Zoo World description D^1 discussed in Section 7.1 (Figures C.1–C.6) and the description D^0 mentioned above (Figure D.1).

Suppose that we would like to update the action description $D^0 \cup D^1$ of this extended Zoo World, to obtain a description of a Circus where not only humans can mount on each other who further can mount on a large animal, but also animals can exchange hats with each other. Assume that the modifiable part D^1_m of D^1 is the same as in Section 7.1, and the modifiable part of D^0_m of D^0 consists of the last causal law in Figure D.1. The sample constraint given in Section 7.2 can be represented by the CCALC query in Figure D.2.

---

[12] The input language of CCALC is explained at its manual at `http://www.cs.utexas.edu/users/tag/cc/`, with further examples.

[13] In CCALC an expression of the form `constraint G` is called a constraint; it is shorthand for the causal law **caused** False **if** $\neg G$.

```
:- sorts
  animal >> human;
  species.

:- variables
  ANML,ANML1                          :: animal;
  H,H1                                :: human;
  SP                                  :: species.

:- objects
% One of the species is human (lmw)
  humanSpecies                        :: species.

:- constants
% Each animal belongs to exactly one of a number of species. (lmw)
% Membership of an animal in a species does not change over time (lmw)
  sp(animal)                          :: species;
% Some species are large, some are not. (lmw)
  largeSpecies(species)               :: boolean;
% Each animal has a position at each point in time. (lmw)
  pos(animal,position)                :: inertialFluent;
% Boolean properties of animals (lmw)
  adult(animal)                       :: boolean;
  mounted(human,animal)               :: inertialFluent.

constraint [\/P | pos(ANML,P)].
caused -pos(ANML,P) if  pos(ANML,P1) & P\=P1.

default largeSpecies(SP).
default adult(ANML).

% Humans are a species called humanSpecies
caused sp(H)=humanSpecies.
constraint sp(ANML)=humanSpecies ->> [\/H | ANML=H].

:- macros
% Adult members of large species are large animals (lmw)
  large(#1) -> adult(#1) & largeSpecies(sp(#1)).

% There is at least one human-species animal in each scenario
%  (lmw)
constraint [\/H | true].

% Two large animals can not occupy the same position,
%  except if one of them rides on the other (lmw)
constraint pos(ANML,P) & pos(ANML1,P) & large(ANML) & large(ANML1)
     ->> [\/H | (H=ANML & mounted(H,ANML1)) ++
                (H=ANML1 & mounted(H,ANML))] where ANML@<ANML1.
```

Fig. C.1. The Zoo World: Animals

```
:- constants
accessible(position,position)       :: sdFluent.

caused accessible(P,P1)
   if neighbor(P,P1) & -[\/G | sides(P,P1,G) & -opened(G)].
default -accessible(P,P1).

% In one unit of time, an animal can move to one of the posi-
%  tions accessible from its present one, or stay in the posi-
%  tion where it is.  Moves to non-accessible positions are
%  never possible (lmw)
constraint -pos(ANML,P1)
   after pos(ANML,P) & -(P=P1 ++ accessible(P,P1)).

% A concurrent move where animal A moves into a position at the
%  same time as animal B moves out of it, is only possible if
%  at least one of A and B is a small animal. (lmw)
%  Exceptions for (failed) mount actions and certain occurrences
%  of throwOff -- when thrown human ends up where another large
%  animal was (see the first two propositions in '%%% ACTIONS %%%')
constraint -(pos(ANML,P) & -pos(ANML1,P))
   after -pos(ANML,P) & pos(ANML1,P)
         & large(ANML) & large(ANML1) unless ab(ANML).

% Two large animals cannot pass through a gate at the same time
% (neither in the same direction nor opposite directions) (lmw)
constraint -(pos(ANML,P1) & pos(ANML1,P1))
   after pos(ANML,P) & pos(ANML1,P) & sides(P,P1,G)
         & large(ANML) & large(ANML1) where ANML@<ANML1.
constraint -(pos(ANML,P) & pos(ANML1,P1))
   after pos(ANML,P1) & pos(ANML1,P) & sides(P,P1,G)
         & large(ANML) & large(ANML1) where ANML@<ANML1.

% While a gate is closing, an animal cannot pass through it
constraint -opened(G) ->> -pos(ANML,P1)
   after pos(ANML,P) & sides(P,P1,G) & opened(G).
```

Fig. C.2. The Zoo World: Movement

```
:- variables
  A,A1                                :: exogenousAction.

:- constants
  move(animal,position),
  open(human,gate),
  close(human,gate),
  mount(human,animal),
  getOff(human,animal,position),
  throwOff(animal,human)              :: exogenousAction.

:- macros
% Action #1 is executed by animal #2
  doneBy(#1,#2) ->
    ([\/P | #1==move(#2,P)] ++
     [\/G | #1==open(#2,G) ++ #1==close(#2,G)] ++
     [\/ANML | #1==mount(#2,ANML)] ++
     [\/ANML \/P | #1==getOff(#2,ANML,P)] ++
     [\/H | #1==throwOff(#2,H)]).

% A failed mount is not subject to the usual, rather strict,
%  movement restriction on large animals
mount(H,ANML) causes ab(H).

% If the position a large human is thrown into was previously
%  occupied by another large animal, the usual movement restriction
%  doesn't apply
throwOff(ANML,H) causes ab(H).

% Every animal can execute only one action at a time
nonexecutable A & A1 if doneBy(A,ANML1) & doneBy(A1,ANML1)
                                                 where A@<A1.
```

Fig. C.3. The Zoo World: Actions, Part 1

```
% Direct effect of move action
move(ANML,P) causes pos(ANML,P).

% An animal can't move to the position where it is now
nonexecutable move(ANML,P) & pos(ANML,P).

% A human riding an animal cannot perform the move action (lmw)
nonexecutable move(H,P) if mounted(H,ANML).

% Effect of opening a gate
open(H,G) causes opened(G).

% A human cannot open a gate if he is not located at a position
%  to the side of the gate (lmw)
nonexecutable open(H,G)
           if pos(H,P) & side1(G)\=P & side2(G)\=P.

% A human cannot open a gate if he is mounted on an animal
nonexecutable open(H,G) if mounted(H,ANML).

% A human cannot open a gate if it is already opened
nonexecutable open(H,G) if opened(G).

% Effect of closing a gate
close(H,G) causes -opened(G).

% A human cannot close a gate if he is not located at a position
%   to the side of the gate (lmw)
nonexecutable close(H,G)
           if pos(H,P) & side1(G)\=P & side2(G)\=P.

% A human cannot close a gate if he is mounted on an animal
nonexecutable close(H,G) if mounted(H,ANML).

% A human cannot close a gate if it is already closed
nonexecutable close(H,G) if -opened(G).

% When a human rides an animal, his position is the same as the
%  animal's position while the animal moves (lmw)
caused pos(H,P) if mounted(H,ANML) & pos(ANML,P).
```

Fig. C.4. The Zoo World: Actions, Part 2

```
% If a human tries to mount an animal that doesn't change position,
%    mounting is successful

caused mounted(H,ANML) if pos(ANML,P)
                       after pos(ANML,P) & mount(H,ANML).

% A human cannot attempt to mount a human who is mounted
nonexecutable mount(H,H1) if mounted(H1,ANML).

% A human cannot be mounted on a human who is mounted
caused false if mounted(H,H1) & mounted(H1,ANML).

% The action fails if the animal changes position, and in this
%  case the result of the action is that the human ends up in
%  the position where the animal was (lmw)
caused pos(H,P) if -pos(ANML,P)
                after pos(ANML,P) & mount(H,ANML).

% A human already mounted on some animal cannot attempt to mount
nonexecutable mount(H,ANML) if mounted(H,ANML1).

% A human can only be mounted on a large animal
constraint mounted(H,ANML) ->> large(ANML).

% A human cannot attempt to mount a small animal (lmw)
nonexecutable mount(H,ANML) if -large(ANML).

% A large human cannot be mounted on a human
constraint mounted(H,H1) ->> -large(H).

% A large human cannot attempt to mount a human
nonexecutable mount(H,H1) if large(H).

% An animal can be mounted by at most one human at a time
constraint -(mounted(H,ANML) & mounted(H1,ANML)) where H@<H1.

% A human cannot attempt to mount an animal already mounted by
%  a human
nonexecutable mount(H,ANML) if mounted(H1,ANML).

% A human cannot attempt to mount an animal if the human is
%  already mounted by a human
nonexecutable mount(H,ANML) if mounted(H1,H).
```

Fig. C.5. The Zoo World: Actions, Part 3

```
% The getOff action is successful provided that the animal does
%  not move at the same time.  It fails if the animal moves, and
%  in this case the rider stays on the animal (lmw)
caused pos(H,P) if pos(ANML,P1)
                after pos(ANML,P1) & getOff(H,ANML,P).

caused -mounted(H,ANML) if pos(ANML,P1)
                        after pos(ANML,P1) & getOff(H,ANML,P).

% The action cannot be performed by a human not riding an animal
%  (lmw)
nonexecutable getOff(H,ANML,P) if -mounted(H,ANML).

% A human cannot attempt to getOff to a position that is not
%  accessible from the current position
nonexecutable getOff(H,ANML,P) if -accessible(P1,P) & pos(ANML,P1).

% The throwOff action results in the human no longer riding the
%  animal and ending in a position adjacent to the animal's
%  present position. It is nondeterministic since the rider may
%  end up in any position adjacent to the animal's present position. (lmw)
throwOff(ANML,H) may cause pos(H,P).
throwOff(ANML,H) causes -mounted(H,ANML).

% If the resultant position is occupied by another large animal
%  then the human will result in riding that animal instead
%  (lmw)
caused mounted(H,ANML1) if pos(H,P) & pos(ANML1,P) & large(ANML1)
   after throwOff(ANML,H) where H\=ANML1.

% The action cannot be performed by an animal not ridden by a
%  human  (lmw)
nonexecutable throwOff(ANML,H) if -mounted(H,ANML).

% The actions getOff and throwOff cannot be executed
%  concurrently

nonexecutable getOff(H,ANML,P) & throwOff(ANML,H).
```

Fig. C.6. The Zoo World: Actions, Part 4

```
:- query
label :: 10;
maxstep :: 3;
% suppose that initially the gate is closed,
% Jumbo and Bart are at different positions in the cage,
% and Homer is outside the cage.
0: -opened(gateAO),
   [\/P \/P1 \/P2 | pos(bart,P), pos(jumbo,P2), P<5, P2<5, P=\=P2,
                    pos(homer,P1), P1>4];
% at some time T, Homer mounts on Jumbo and next Bart mounts on Homer;
% afterwards, Homer is mounted on Jumbo and Bart is mounted on Homer.
[\/T | T+1<maxstep && (T: mount(homer,jumbo)) && (T+1: mount(bart,homer))
       && (T+2: mounted(bart,homer)) && (T+2: mounted(homer,jumbo))];
% also Jumbo does not change its position in the cage.
[\/P3 | P3<5 && [/\T1 | T1=<maxstep ->> (T1: pos(jumbo,P3))]].
```

Fig. C.7. The sample constraint given in Section 7.1 for updating the Zoo World into a little Circus, expressed as a query in CCALC.

```
:- query
label :: 10;
maxstep :: 3;
% suppose that initially the gate is closed,
% Jumbo and Bart are at different positions in the cage,
% and Homer is outside the cage.
0: -opened(gateAO),
   [\/P \/P1 \/P2 | pos(bart,P), pos(jumbo,P2), P<5, P2<5, P=\=P2,
                    pos(homer,P1), P1>4];
% at some time T, Homer mounts on Jumbo and next Bart mounts on Homer;
% afterwards, Homer is mounted on Jumbo and Bart is mounted on Homer.
[\/T | T+1<maxstep && (T: mount(homer,jumbo)) && (T+1: mount(bart,homer))
       && (T+2: mounted(bart,homer)) && (T+2: mounted(homer,jumbo))];
% also Jumbo does not change its position in the cage.
[\/P3 | P3<5 && [/\T1 | T1=<maxstep ->> (T1: pos(jumbo,P3))]];
% suppose that in the update description of the Zoo World,
% the causal laws labeled by aux2(bart, homer, jumbo)
% and aux3(bart, homer, jumbo) are deleted:
(T2=<maxstep ->>
 ([/\H2 /\P /\ANML | (T2: aux1(H2,ANML,P))] &&
  [/\H /\H1 /\ANML | (H \= bart && H1 \= homer && ANML \= jumbo) ->>
                     ((T2: aux2(H,H1,ANML)) && (T2: aux3(H,H1,ANML)))]).
```

Fig. C.8. The constraint of Figure C.7 modified by adding further constraints as described in Section 7.1.

```
0:  pos(homer, 7)  pos(bart, 3)  pos(jumbo, 4)  happy(bart)
-aux2(bart, homer, jumbo)  -aux3(bart, homer, jumbo)

ACTIONS:  move(bart, 4)  open(homer, gateAO)

1:  opened(gateAO)  pos(homer, 7)  pos(bart, 4)  pos(jumbo, 4)
happy(bart)  -aux2(bart, homer, jumbo)  -aux3(bart, homer, jumbo)

ACTIONS:  mount(homer, jumbo)  mount(bart, homer)

2:  opened(gateAO)  pos(homer, 4)  pos(bart, 7)  pos(jumbo, 4)
mounted(homer, jumbo)  happy(bart)  -aux2(bart, homer, jumbo)
-aux3(bart, homer, jumbo)

ACTIONS:  mount(bart, homer)  sing(homer)

3:  opened(gateAO)  pos(homer, 4)  pos(bart, 4)  pos(jumbo, 4)
mounted(homer, jumbo)  mounted(bart, homer)  happy(homer)  happy(bart)
-aux2(bart, homer, jumbo)  -aux3(bart, homer, jumbo)
```

Fig. C.9. A possible scenario which shows that the constraint of Figure C.8 is satisfied by the Zoo World (i.e., the Zoo World can be updated into a little Circus) if we remove the causal laws labeled by `aux2(bart, homer, jumbo)` and `aux3(bart, homer, jumbo)`.

```
:- sorts
  species;
  animal >> (monkey; dog);
  hat.

:- variables
  D                 :: dog;
  ANML,ANML1        :: animal;
  SP                :: species;
  HA,HA1            :: hat;
  MO,MO1            :: monkey.

:- objects
  dogSpecies            :: species;
  monkeySpecies         :: species.

:- constants
  sp(animal)        :: species;
  exchange(hat,hat) :: exogenousAction;
  owner(hat,animal),
  aux4(hat,animal) :: inertialFluent.

caused sp(D)=dogSpecies.
constraint sp(ANML)=dogSpecies ->> [\/D | ANML=D].

caused sp(MO)=monkeySpecies.
constraint sp(ANML)=monkeySpecies ->> [\/MO | ANML=MO].

caused -owner(HA,ANML1) if owner(HA,ANML) & ANML\=ANML1.
caused -owner(HA1,ANML) if owner(HA,ANML) & HA\=HA1.

exchange(HA,HA1) causes owner(HA,ANML1) & owner(HA1,ANML)
  if owner(HA1,ANML1) & owner(HA,ANML).
nonexecutable exchange(HA,HA).

caused false if owner(HA,ANML) && sp(ANML)\=monkeySpecies.
```

Fig. D.1. Monkeys exchanging hats.

```
:- query
label ::e10;
maxstep :: 1;
[\/HA1 \/HA2 \/T4 | HA1\=HA2 && (0: owner(HA1,apu)) &&
   (0: owner(HA2,snoopy)) && T4<maxstep && (T4: exchange(HA1,HA2))].
```

Fig. D.2. The sample constraint to check that Snoopy and Abu can exchange hats.

```
:- query
label ::e11;
maxstep :: 1;
[\/HA1 \/HA2 \/T4 | HA1\=HA2 && (0: owner(HA1,apu)) &&
   (0: owner(HA2,snoopy)) && T4<maxstep && (T4: exchange(HA1,HA2))];
[/\HA /\ANML /\T | T =< maxstep ->> (T: -aux4(HA,ANML))].
```

Fig. D.3. The sample constraint to check that Snoopy and Abu can exchange hats, if we remove the causal laws labeled by aux4 from the description in Figure D.1.

```
?- query e11.

0:  owner(ha1, snoopy)  owner(ha2, apu)
-aux4(ha1, apu)  -aux4(ha1, mo1)  -aux4(ha1, mo2)
-aux4(ha1, snoopy)  -aux4(ha2, apu)  -aux4(ha2, mo1)
-aux4(ha2, mo2)  -aux4(ha2, snoopy)

ACTIONS:  exchange(ha2, ha1)

1:  owner(ha1, apu)  owner(ha2, snoopy)
-aux4(ha1, apu)  -aux4(ha1, mo1)  -aux4(ha1, mo2)
-aux4(ha1, snoopy)  -aux4(ha2, apu)
-aux4(ha2, mo1)  -aux4(ha2, mo2)  -aux4(ha2, snoopy)
```

Fig. D.4. A possible scenario to show that the sample constraint to check that Snoopy and Abu can exchange hats is satisfied by the description in Figure D.1 if we remove the causal laws labeled by aux4.