*

INTERNATIONAL COMPUTER SCIENCE INSTITUTE

1947 Center St. • Suite 600 • Berkeley, California 94704-1198 • (510) 643-9153 • FAX (510) 643-7684

Ramification and Causality

Michael Thielscher

TR-96-003

January 1996

Abstract

The ramification problem in the context of commonsense reasoning about actions and change names the challenge to accommodate actions whose execution causes indirect effects. Not being part of the respective action specification, such effects are consequences of general laws describing dependencies between components of the world description. We present a general approach to this problem which incorporates causality, formalized by directed relations between two single effects stating that, under specific circumstances, the occurrence of the first causes the second. Moreover, necessity of exploiting causal information in this way or a similar is argued by elaborating the limitations of common paradigms employed to handle ramifications, namely, the principle of categorization and the policy of minimal change. Our abstract solution is exemplarily integrated into a specific calculus based on the logic programming paradigm.

* On leave from FG Intellektik, TH Darmstadt

1 Introduction

The ability to reason about causality, which involves predicting the effects of own actions and explaining observed phenomena, serves humans as basis for understanding the world to an extent sufficient to survive and for acting intelligently in their habitat. Formal approaches to model this ability have therefore a long tradition in philosophy and, especially, AI, where this research area was initiated by the paper [McCarthy, 1959], which claimed reasoning about actions play a fundamental role in common sense.

Drawing conclusions about dynamically changing environments is grounded on formal specifications of what effects are caused by the execution of a particular action. Since it is obviously impossible to provide an exhaustive description, which defines the result of executing an action in each possible state of the world, action specifications are to be restricted to the part of the world which they possibly affect—while the rest of the world is subject to the *law of persistence*, i.e., is assumed to remain stable. But, even this focusing on what effects an action may possibly cause becomes intractable in complex domains if one tries to put all effects into a single, complete specification. For actions often only at first glance appear to admit a most simple description since their execution causes merely a small amount of *direct* changes, which, however, may in turn initiate a long chain of *indirect* effects hard to entirely oversee. For instance, consider the action of toggling a switch, which in the first place causes nothing but a change of the switch's position. However, the switch is probably part of an electric circuit so that, say, some light bulb is turned off as side effect, which in turn may cause someone to hurt himself in a suddenly darkened room by running against a chair that, as a consequence, falls into a television set whose implosion activates the fire alarm and so on and so forth.[1]

The task of designing a framework to formalize action scenarios where action specifications are not assumed to completely describe all possible effects, constitutes the *ramification problem*.[2] A satisfactory solution to this problem requires successful treatment of two major issues:

First, an appropriately weakened version of the aforementioned law of persistence is needed which applies only to those parts of the world description that are not affected as regards direct effects *nor* regarding indirect effects. As a solution to this problem, in this article we suggest to keep the law of persistence as it stands but to consider the world description obtained through its application merely as intermediate result. Indirect effects are then accommodated by further investigation until an overall satisfactory successor state obtains. This method perfectly accounts for both rigorous persistence of unaffected parts on the one hand, which avoids uncaused changes, and arbitrarily complex chains of indirect effects on the other hand.

Second, indirect effects typically are consequences of additional, general knowledge of domain-specific dependencies between world description components—but not all effects suggested from this perspective are desirable from the standpoint of causality. As an example, consider the

[1] A crucial question in this context concerns the distinction between indirect effects occurring during a single world's state transition step and those which deserve separate state transitions (also called *delayed* effects). E.g., the above may not only be described as "the fire alarm is activated in the successor state after having toggled the switch," but also as, say, "the chair is falling in the successor state (and presumably hits the television set during the next state transition)." As a reasonable, albeit informal, guidance we suggest a single state transition should summarize what happens until someone, e.g. the reasoning agent himself, has the possibility to intervene by acting again (stopping the chair from falling, for instance).

[2] The naming was suggested in [Ginsberg and Smith, 1988b] being inspired by [Finger, 1987]. The latter, however, was not devoted to the ramification problem in exactly the above sense, contrary to what is often claimed; rather, this thesis describes how to exploit logical consequences (called ramifications) of goal specifications in planning problems, with the aim to restrict search space.

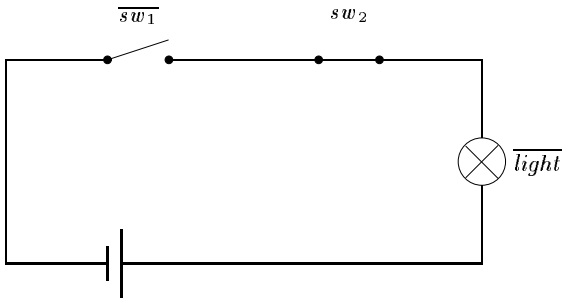

Figure 1: An electric circuit consisting of a battery, two switches and a light bulb, which is on if and only if both switches are on. The respective state of each of the three dynamic components involved is described by a unique propositional constant, where negation is denoted by a bar on top of the respective symbol.

simple electric circuit depicted in Figure 1, which consists of a battery, two switches and a light bulb. The obvious connection between these components may formally be described by the logic expression $sw_1 \wedge sw_2 \equiv light$, i.e., the light is on if and only if both switches are on. Now, if we toggle the first switch in the particular state displayed, where both sw_1 and $light$ are false while sw_2 is true, then, aside from the direct effect of sw_1 becoming true, we also expect that the light bulb turns on. This indirect effect is inspired by the formula just mentioned, which includes the implication $sw_1 \wedge sw_2 \supset light$. However, despite this being the intuitively expected result, the mere knowledge of the connection between the switches and the bulb is not sufficient: Note that the above formula, $sw_1 \wedge sw_2 \equiv light$, logically also entails the implication $sw_1 \wedge \overline{light} \supset \overline{sw_2}$, which suggests that instead of the light being turned on, the indirect effect of toggling the first switch is that the second one jumps its position—a result which contradicts our intuition.

The reason for the inadequacy of merely taking into account formalizations of dependencies as pure logical formulas—usually called *domain constraints*—is that these formulas do not include causal information. More precisely, $sw_1 \wedge \overline{light} \supset \overline{sw_2}$ is clearly true in any state and, therefore, contains *evidential* information, i.e., if one observes sw_1 be true and $light$ be false then it is safe to conclude sw_2 must be false. But, exploiting this implication for reasoning about *causality* is misleading.[3]

As a solution to the second problem, we propose to incorporate causality in form of so-called *causal relationships*, which formalize statements like

A change of $\overline{sw_1}$ to sw_1 causes a change of $\overline{light}$ to $light$, provided sw_2 is true.

We intend to, after having computed the direct effects of executing an action in a particular state of the world, apply suitable causal relationships, one by one, to accommodate indirect effects until a satisfactory result obtains. Employing a collection of single causal relationships, each of which only relates two particular effects, accounts for both several indirect effects caused by a direct one and also indirect effects in turn causing further indirect effects. To illustrate the latter, consider the relationship

[3] See [Pearl, 1988a] for a general discussion on the different nature of causal compared to evidential implications.

A change of $\overline{light}$ to *light* causes a change of $\overline{light\text{-}detector}$ to *light-detector*, provided *detector-activated* is true.

in addition to the one above. Since we do not expect the designer of a formal domain specification to create a complete set of suitable causal relationships, we will moreover present an automated procedure to extract them from given domain constraints plus some general information on which components of a state description—usually called *fluents*—may possibly influence other fluents. Causal relationships and their automatic generation are formally introduced in Section 3.

Yet the purpose of this article is not only to suggest incorporating causal information by means of our causal relationships but also to argue this, or employing similar concepts, be inevitable when trying to cope with the ramification problem in general. To this end, an entire section, 5, will be devoted to illustrating the limited expressiveness of existing paradigms aiming at handling ramifications, namely, the principle of *categorization* and the policy of *minimal change*. Roughly, categorization-based approaches distinguish fluents that are more likely to change than other fluents (e.g., a change of *light* is considered more likely than a change of sw_2). While this relies on the existence of an appropriate categorization, we will illustrate that some fluents may strive against falling into a single category (Subsection 5.1). Even more common is the assumption of minimal change, which amounts to rejecting a resulting state if, informally spoken, it is obtained by changing the values of strictly more fluents than necessary. While we will offer a formal proof that our method captures all reasonable resulting states with minimal distance to the original state (Section 4), in Subsection 5.2 we will illustrate that we are also able to find non-minimal resulting states, which are perfectly well conceivable if only all changes are reasonable from the standpoint of causality. We have intentionally omitted concrete references to approaches based on either paradigm so far since a detailed comparison with related work can be found in the discussion, Section 7.

In the second part of the paper, Section 6, we integrate the concept of causal relationships into a particular action calculus which is based on reifying entire state descriptions and which employs the logic programming paradigm [Hölldobler and Schneeberger, 1990; Hölldobler and Thielscher, 1995]. While for sake of simplicity states are described via a set of propositional constants in the first part (see Section 2), the calculus itself employs more complex a notion of fluent, which comes along with fluent formulas involving quantifications. The extended calculus will be proved sound and complete wrt the formal semantics induced by causal relationships and their application.

2 A Basic Theory of Actions

In the first part of the paper, we employ a most elementary theory of actions and change, which enables us to stress solely on the problem of domain constraints and ramifications. The basic entities of action scenarios are *states*, each of which is a snapshot of the underlying dynamic system, i.e., the part of the world being modeled, at a particular instant of time. Formally, a state shall be described by assigning truth-values to a fixed set of propositional constants.

Definition 1 Let $\mathcal{F}$ be a finite set of symbols called *fluent names*. A *fluent literal* is either a fluent name $f \in \mathcal{F}$ or its negation, denoted by $\overline{f}$. A set of fluent literals is *inconsistent* iff it contains some $f \in \mathcal{F}$ along with $\overline{f}$. A *state* is a maximal consistent set of fluent literals. ■

Note that formally each combination of truth-values denotes a state, some of which, however, might be considered impossible due to domain-specific dependencies between some fluents (see

below). Throughout the paper we assume the following notational conventions: If ℓ is a fluent literal then by $|\ell|$ we denote its affirmative component, i.e, $|f| = |\overline{f}| = f$ where $f \in \mathcal{F}$. This notation extends to sets of fluent literals S as follows: $|S| = \{|\ell| : \ell \in S\}$; e.g., for each state S we have $|S| = \mathcal{F}$. Furthermore, if ℓ is a negative fluent literal then $\overline{\ell}$ should be interpreted as $|\ell|$ (in other words, $\overline{\overline{f}} = f$). Finally, if S is a set of fluent literals then by $\overline{S}$ we denote the set $\{\overline{\ell} : \ell \in S\}$; e.g., $\overline{\mathcal{F}}$ contains all negative fluent literals given a set $\mathcal{F}$ of fluent names.

Example 1 To model the Electric Circuit example depicted in Figure 1, we use these three fluents: $\mathcal{F} = \{sw_1, sw_2, light\}$ to denote the state of the two switches and the light bulb, respectively. The current state displayed in Figure 1 is then represented by $\{\overline{sw_1}, sw_2, \overline{light}\}$. ■

Given an underlying set of fluent names, these can be considered atoms for constructing (propositional) formulas, using the standard logical connectives, to allow for statements about states. Truth and falsity, respectively, of such formulas wrt a particular state S are based on defining a literal ℓ to be true if and only if $\ell \in S$.

Definition 2 Let $\mathcal{F}$ be a set of fluent names. The set of *fluent formulas* is inductively defined as follows: Each fluent literal in $\mathcal{F} \cup \overline{\mathcal{F}}$ and $\top$ (*true*) and $\bot$ (*false*) are fluent formulas, and if F and G are fluent formulas so are $F \wedge G$, $F \vee G$, $F \supset G$, and $F \equiv G$.[4]

Let S be a state and F a fluent formula, then the notion of F being *true* in S, written $S \models F$, is inductively defined as follows:

1. $S \models \top$ and $S \not\models \bot$;
2. $S \models \ell$ iff $\ell \in S$, for each fluent literal ℓ;
3. $S \models F \wedge G$ iff $S \models F$ and $S \models G$;
4. $S \models F \vee G$ iff $S \models F$ or $S \models G$;
5. $S \models F \supset G$ iff $S \not\models F$ or $S \models G$;
6. $S \models F \equiv G$ iff $S \models F$ and $S \models G$, or $S \not\models F$ and $S \not\models G$.

■

Fluent formulas provide means to distinguish states that cannot occur due to specific dependencies between particular fluents. Formulas which have to be satisfied in all states that are conceivable in a domain are also called *domain constraints*.

Example 1 (continued) Given the fluent names of Example 1, we can employ fluent formula $sw_1 \wedge sw_2 \equiv light$ as domain constraint in order to model the intended relation between the two switches and the light bulb. This formula holds, for instance, in the state depicted in Figure 1, $\{\overline{sw_1}, sw_2, \overline{light}\}$, but is violated in, say, $\{sw_1, sw_2, \overline{light}\}$. ■

The second basic entity in frameworks to reason about dynamic environments are *actions*, whose execution causes state transitions. Again, since stress shall lie on the ramification problem rather than on sophisticated methods of specifying the *direct* effects of actions, we employ a fairly simple, STRIPS-style [Fikes and Nilsson, 1971; Lifschitz, 1986] notion of action specification. Each *action description* consists of

[4] As negation can be expressed through negative literals, we omit the basic connective "$\neg$"—this is just for sake of readability as it avoids too many different forms of negation, especially in view of Section 6, where logic programs including the principle of negation-as-failure are used.

- A *condition* C, which is a set of fluent literals, all of which must be contained in a state in case the action description is intended to be applied.
- A (direct) *effect* E, also which is a set of fluent literals, all of which shall hold in the resulting state after having applied the action description.

For simplicity, we assume $|C| = |E|$ (i.e., condition and effect refer to the very same set of fluent names), which enables us to obtain the state resulting from the direct effect by simply removing set C from the state at hand and adding set E to it. This assumption does not impose a severe restriction of expressiveness as several descriptions for a single action are supported (e.g., specifying the action "toggle the first switch" in our Electric Circuit scenario requires a description for both sw_1 being false and true in the current state—see below). Yet, we assume at most one of the available action descriptions for a particular action be applicable wrt any particular state, for we want to avoid non-deterministic behavior regarding direct effects.

Definition 3 Let $\mathcal{F}$ be a set of fluent names. An *action description* is a triple $\langle C, a, E\rangle$ where C, called *condition*, and E, called *effect*, are consistent sets of fluent literals; and a, called *action name*, is a symbol not occurring in $\mathcal{F}$. It is assumed that $|C| = |E|$.

If S is a state then an action description $\alpha = \langle C, a, E\rangle$ is *applicable* in S iff $C \subseteq S$. The *application* of α to S yields $(S \setminus C) \cup E$. Any set $\mathcal{A}$ of action descriptions is assumed to contain at most one applicable description for each action name a and state S. ■

Note that S being a state, C and E being consistent, and $|C| = |E|$ guarantee $(S \setminus C) \cup E$ to be a state again—not necessarily, however, one which is possible according to domain constraints.

Example 1 (continued) Our Electric Circuit domain allows for two actions, namely, toggling either switch, which is why we consider the action names $toggle_1$ and $toggle_2$, respectively, along with the four action descriptions

$$\begin{array}{ll} \langle\, \{\overline{sw_1}\}\,,\, toggle_1\,,\, \{sw_1\}\,\rangle & \langle\, \{\overline{sw_2}\}\,,\, toggle_2\,,\, \{sw_2\}\,\rangle \\ \langle\, \{sw_1\}\,,\, toggle_1\,,\, \{\overline{sw_1}\}\,\rangle & \langle\, \{sw_2\}\,,\, toggle_2\,,\, \{\overline{sw_2}\}\,\rangle \end{array} \tag{1}$$

When executing, say, $toggle_1$ in state $S = \{\overline{sw_1}, sw_2, \overline{light}\}$, the first of these descriptions is applicable due to $\{\overline{sw_1}\} \subseteq S$; its application yields

$$(S \setminus \{\overline{sw_1}\}) \cup \{sw_1\} \;=\; \{sw_1, sw_2, \overline{light}\}$$

■

This example illustrates that the state obtained via applying an action description may violate the underlying domain constraint(s) since only direct effects have been specified. In the next section, we develop a method to further modify such a preliminarily resulting, impossible state in order to account for additional, indirect effects of having executed an action.

3 Causal Relationships

The ramification problem arises as soon as it does not suffice to compute the direct effects of actions only. The resulting collection of fluent literals may violate underlying domain constraints, which in turn give rise to additional, indirect effects. Theoretically, we could of

course compile all conceivable indirect effects into action descriptions by exploiting the fact that an arbitrary number of different descriptions for a single action can be formulated. For instance, replacing $\langle\{\overline{sw_1}\}, toggle_1, \{sw_1\}\rangle$ by both $\langle\{\overline{sw_1}, sw_2, \overline{light}\}, toggle_1, \{sw_1, sw_2, light\}\rangle$ and $\langle\{\overline{sw_1}, \overline{sw_2}, \overline{light}\}, toggle_1, \{sw_1, \overline{sw_2}, \overline{light}\}\rangle$ helps to obtain the intended result when applying $toggle_1$ to state $\{\overline{sw_1}, sw_2, \overline{light}\}$, viz $\{sw_1, sw_2, light\}$. However, this procedure bears two major disadvantages demonstrating its inadequacy. First, generally an enormous number of action descriptions is needed to account for every possible combination of indirect effects. Consider, for example, a model of an electric circuit where a distinguished switch is involved in n domain constraints each of which contains a switch-bulb pair in a similar fashion as in Example 1. Defining the effect of toggling the separate switch solely by means of action descriptions would then require 2^{n+1} different descriptions, one for each possible combination of truth-values assigned to the switch being operated and the other n switches. Second, adding a new domain constraint may require, in the worst case, a re-definition of the entire set of action descriptions used before.

3.1 Applying Causal Relationships

As a consequence of the above observation, we keep a given set of action descriptions which concentrates on direct effects only and regard the resulting collection of fluent literals, obtained after having applied such a description, merely as an intermediate state, which requires additional computation accounting for possible indirect effects. A single indirect effect is obtained according to a directed *causal* relation between specific fluents. For instance, having as direct effect a change of the first switch into its lower position in the state depicted in Figure 1, this is regarded as additionally *causing* the light bulb to change its state also, for the second switch is on. We will formalize such causal relationships by expressions like

$$sw_1 \ \texttt{causes} \ light \ \texttt{if} \ sw_2 \tag{2}$$

Formally, such expressions operate on state-effect pairs (S, E) where S is the current collection of fluent literals and E contains all (direct or indirect) effects that have been considered so far. E.g., let $S = \{sw_1, sw_2, \overline{light}\}$ be the state obtained after having computed the direct effect of $toggle_1$, as described in the preceding section, then $E = \{sw_1\}$. The causal relationship formalized in (2) gives rise to indirect effect $light$, which supersedes $\overline{light}$ in S. Moreover, this new effect is added to E; altogether, this results in the new state-effect pair $(\{sw_1, sw_2, light\}, \{sw_1, light\})$.

The reason for employing and manipulating the second component, E, is that identical intermediate states S can be reached by different effects E, each of which may require a different, sometimes opposite treatment. Consider, as an example, two switches, sw_1 and sw_2, being mechanically connected (say, through a spring) such that they cannot be simultaneously on. The corresponding domain constraint, $\overline{sw_1} \vee \overline{sw_2}$, gives rise to two causal relationships, viz

$$sw_1 \ \texttt{causes} \ \overline{sw_2} \ \texttt{if} \ \top \tag{3}$$

$$sw_2 \ \texttt{causes} \ \overline{sw_1} \ \texttt{if} \ \top \tag{4}$$

Now, toggling the first switch in state $\{\overline{sw_1}, sw_2\}$ yields, following action descriptions (1), intermediate state $\{sw_1, sw_2\}$. But the very same intermediate state is also obtained by toggling the second switch in state $\{sw_1, \overline{sw_2}\}$. Yet the intended outcomes differ considerably: In the

first case, the final result should be $\{sw_1, \overline{sw_2}\}$, while $\{\overline{sw_1}, sw_2\}$ is expected in the second. This distinction can only be achieved by referring to the differing effects, $E_1 = \{sw_1\}$ and $E_2 = \{sw_2\}$. The former enables application only of (3) to the intermediate result, $\{sw_1, sw_2\}$, the latter only of (4), which leads to the respective desired successor state.[5]

The formal definition of causal relationships and their application is as follows:

Definition 4 Let $\mathcal{F}$ be a set of fluent names. A *causal relationship* is an expression of the form e `causes` r `if` Φ where Φ is a fluent formula based on $\mathcal{F}$ and e and r are fluent literals.

Let (S, E) be a pair consisting of a state S and a set of fluent literals E, then a causal relationship e `causes` r `if` Φ is *applicable* to (S, E) iff $S \models \Phi \wedge e \wedge \overline{r}$ and $e \in E$. Its application yields the pair $((S \setminus \{\overline{r}\}) \cup \{r\}, E \cup \{r\})$.

Given a set $\mathcal{R}$ of causal relationships, by $(S, E) \rightsquigarrow_{\mathcal{R}} (S', E')$ we denote the existence of a causal relationship in $\mathcal{R}$ whose application to (S, E) yields (S', E'). ■

In words, a causal relationship is applicable if the associated condition Φ holds, the particular indirect effect r is currently false, and its cause e has been observed and is currently true.[6]

Now, assume we are given a suitable set of causal relationships and have obtained a set of fluent literals S after having computed the direct effects of an action via Definition 3. State S might violate the underlying domain constraints. In order to obtain a satisfactory resulting state, we then compute additional, indirect effects by (non-deterministically) selecting and (serially) applying causal relationships. If this procedure eventually results in a state satisfying the domain constraints then such a state is considered *successor* state:[7]

Definition 5 Let $\mathcal{F}$ be a set of fluent names, $\mathcal{A}$ a set of action descriptions, $\mathcal{D}$ a set of domain constraints, and $\mathcal{R}$ a set of causal relationships. Furthermore, let S be a state satisfying $\mathcal{D}$ and a an action name. If there exists an applicable (wrt S) action description $\langle C, a, E\rangle \in \mathcal{A}$ then a state S' is *successor state* iff

1. $((S \setminus C) \cup E, E) \overset{*}{\rightsquigarrow}_{\mathcal{R}} (S', E')$ for some E' and
2. S' satisfies $\mathcal{D}$.

■

Example 1 (continued) An adequate set of causal relationships for our Electric Circuit domain consists in the following four elements:

$$\begin{array}{llll} sw_1 & \texttt{causes} \; light & \texttt{if} \; sw_2 & \qquad \overline{sw_1} \; \texttt{causes} \; \overline{light} \; \texttt{if} \; \top \\ sw_2 & \texttt{causes} \; light & \texttt{if} \; sw_1 & \qquad \overline{sw_2} \; \texttt{causes} \; \overline{light} \; \texttt{if} \; \top \end{array} \tag{5}$$

In words, if either switch is switched on then this causes the light bulb to turn on provided the other switch is already on. Conversely, switching either switch off results in a dark bulb regardless of the other switch's position.

[5] Other examples that require taking into account how an intermediate state was obtained can be found in Subsections 3.3 and 5.3.

[6] The reason for the very last stipulation, formally expressed by $S \models e$ in addition to $e \in E$, is that some other indirect effect might occur that causes a withdrawal of the previously computed e! After having withdrawn this effect, it is of course no longer available as cause for other effects. Such situations are perfectly reasonable, as will be shown by Example 4 in Section 5.

[7] We adopt a standard notation in writing $(S, E) \overset{*}{\rightsquigarrow}_{\mathcal{R}} (S', E')$ to indicate the existence of a (possibly empty) sequence of causal relationships in $\mathcal{R}$ whose successive application to (S, E) yields (S', E').

Applying the first action description in (1) to the state depicted in Figure 1, $\{\overline{sw_1}, sw_2, \overline{light}\}$, yields the pair $(\{sw_1, sw_2, \overline{light}\}, \{sw_1\})$, to which the first of the given causal relationships is applicable, resulting in $(\{sw_1, sw_2, light\}, \{sw_1, light\})$. The first component of this pair satisfies the underlying domain constraint, $sw_1 \wedge sw_2 \equiv light$; hence, it is a successor state. Moreover, it is the only successor state according to Definition 5 as no other causal relationship in (5) is applicable. ■

While the application ordering might be crucial in so far as a different ordering may allow for more causal relationships be applied (or less, respectively; see Subsection 5.2 for a key example), we can prove the following important result of order independence in case a unique set of relationships is applied:

Proposition 6 *Let $\mathcal{F}$ be a set of fluent names, S a state, and E a set of fluent literals. Furthermore, let $\rho_1, \ldots, \rho_n$ be a sequence of causal relationships ($n \geq 0$) applicable to (S, E) yielding*

$$(S, E) \overset{\rho_1}{\rightsquigarrow} (S_1, E_1) \overset{\rho_2}{\rightsquigarrow} \ldots \overset{\rho_n}{\rightsquigarrow} (S_n, E_n) \tag{6}$$

Then, for any permutation $\rho_{\pi(1)}, \ldots, \rho_{\pi(n)}$ which is also applicable to (S, E), i.e.,

$$(S, E) \overset{\rho_{\pi(1)}}{\rightsquigarrow} (S'_1, E'_1) \overset{\rho_{\pi(2)}}{\rightsquigarrow} \ldots \overset{\rho_{\pi(n)}}{\rightsquigarrow} (S'_n, E'_n) \tag{7}$$

we have $S_n = S'_n$ and $E_n = E'_n$.

> **Proof:** Since the indirect effects, r_i, of the applied causal relationships $\rho_i = e_i$ `causes` r_i `if` Φ_i ($1 \leq i \leq n$) are monotonically added to the second component, E, we obviously have $E_n = E'_n$.
>
> To show $S_n = S'_n$, let, for each $f \in \mathcal{F}$,
>
> $$\mathcal{R}_f := \{ e_i \texttt{ causes } r_i \texttt{ if } \Phi_i : |r_i| = f,\ 1 \leq i \leq n \}$$
>
> be the set of all relationships which change the truth-value of f in the course of (6) or (7), respectively. Since applying a causal relationship requires the indirect effect, r_i, being explicitly false in the current state, the finally obtained truth-value of f depends only on the number of relationships in $\mathcal{R}_f$ causing f compared to the number of relationships causing $\overline{f}$.[8] Since (6) and (7) do not differ in this respect, we know $f \in S_n$ iff $f \in S'_n$ and $\overline{f} \in S_n$ iff $\overline{f} \in S'_n$.
>
> ■

It is important to realize that neither uniqueness of a successor state nor even its existence is guaranteed in general. The former characterizes actions with non-deterministic behavior—a concept which will be discussed in detail later, in Subsection 5.2; the meaning of the latter will be elucidated at the end of the current section, in Subsection 3.3. First of all, however, we raise another crucial issue, namely, how to obtain an adequate set of causal relationships on the basis of given domain constraints.

[8] More precisely, if $f \in S$ then $\mathcal{R}_f$ contains either equally many elements causing f and $\overline{f}$, respectively, or one more causing $\overline{f}$. In the former case, we have $f \in S_n$ (and $f \in S'_n$), in the latter $\overline{f} \in S_n$ (and $\overline{f} \in S'_n$). The analogue holds in case $\overline{f} \in S$.

3.2 Influence Information

Obtaining the intended result by applying causal relationships to compute indirect effects of actions relies, of course, on a suitable set of these relationships. Such a set should be sound in so far as each element represents an intuitively reasonable causal relation, and it should also be complete in so far as it covers all conceivable indirect effects.[9] The four causal relationships (5) used in the Electric Circuit domain constitute such a suitable set. There is obviously a close correspondence between the elements of this set and the domain constraint underlying this exemplary scenario, which is why the following analysis is devoted to the problem of how an adequate set of causal relationships can be automatically extracted from given domain constraints.

There is, however, a crucial, well-known obstacle to be considered towards this end: Despite the observation that the causal relationships in (5) are inspired by domain constraint $sw_1 \wedge sw_2 \equiv light$, this formula would also give rise to more, unintended causal relationships if evaluated from a purely syntactical point of view. For instance, the fact that if sw_1 becomes true then it is impossible to have both sw_2 and $\overline{light}$, equally well suggests the following causal relationship:

$$sw_1 \ \texttt{causes} \ \overline{sw_2} \ \texttt{if} \ \overline{light} \tag{8}$$

aside from sw_1 `causes` $light$ `if` sw_2. This additional causal relationship would, however, sanction state $\{sw_1, \overline{sw_2}, \overline{light}\}$ be possible successor state of turning on sw_1 in the state depicted in Figure 1, i.e., the second switch would magically jump its position in order to satisfy the domain constraint. Though this is an unintuitive outcome, the mere domain constraint does not provide enough information to rule out (8). Hence, there must be some additional domain knowledge one employs here for claiming a preference amongst the two possible changes.

More precisely, we appear to exploit general information of potential influence of some fluents upon others. For instance, it is simply known that changing a switch's position might influence the state of a light bulb rather than directly causing other switches to move.[10] It is therefore inevitable to provide this kind of domain knowledge in addition to domain constraints when seeking an adequate set of causal relationships. This information of possible influences is formalized as follows:

Definition 7 Let $\mathcal{F}$ be a set of fluent names. A binary relation $\mathcal{I} \subseteq \mathcal{F} \times \mathcal{F}$ on these fluent names is called *influence information*. ■

If $(f_1, f_2) \in \mathcal{I}$ then this is intended to denote that a change of f_1's truth-value might possibly influence the truth-value of f_2. Hence, regarding the Electric Circuit domain we have $\mathcal{I} = \{(sw_1, light), (sw_2, light)\}$, i.e., both switches might influence the light but not vice versa nor do they mutually interfere.

Domain constraints plus influence information constitute an amount of knowledge suitable for automatically generating an adequate set of causal relationships. The basic idea is to investigate all possible violations of a domain constraint and formalize various causal relationships each of which helps to 'correct' such a violation while accommodating the influence information. More precisely, if $\mathcal{D}$ is the set of underlying domain constraints then we construct the conjunctive normal form (CNF, for short) of $\bigwedge \mathcal{D}$, i.e, of the conjunction of all constraints. Then $\mathcal{D}$ is

[9] Here, "conceivable" can—to state the obvious—refer only to what is potentially derivable given the domain constraints. One cannot expect computation of an indirect effect desired in some scenario if the scenario's formalization does not include a piece of knowledge hinting at the possible existence of this effect.

[10] The use of word "directly" is crucial since switches do have the ability to influence the position of other switches indirectly, e.g., through activating a relay (see Example 3 below).

violated iff a conjunct $\ell_1 \vee \ldots \vee \ell_m$ in this CNF is violated, which in turn is true iff $\overline{\ell_1} \wedge \ldots \wedge \overline{\ell_m}$ holds. The reason for such a violation must be some (direct or indirect) effect $\overline{\ell_j}$, and this 'flaw' can be 'corrected' by changing some other $\overline{\ell_k}$ to ℓ_k via a causal relationship—but only in case fluent $|\ell_j|$ can possibly influence fluent $|\ell_k|$ according to $\mathcal{I}$. All this is formalized as follows:

Definition 8 Let $\mathcal{F}$ be a set of fluent names and $\mathcal{D}$ a set of domain constraints. An influence information $\mathcal{I}$ then determines a set of causal relationships $\mathcal{R}$ following this procedure:

1. Let $\mathcal{R} := \{\}$.
2. Let $D_1 \wedge \ldots \wedge D_n$ ($n \geq 0$) be the CNF of $\bigwedge \mathcal{D}$. For each $D_i = \ell_1 \vee \ldots \vee \ell_{m_i}$ ($i = 1, \ldots, n$) do the following:
3. For each $j = 1, \ldots, m_i$ do the following:
4. For each $k = 1, \ldots, m_i$ such that $(|\ell_j|, |\ell_k|) \in \mathcal{I}$, add this causal relationship to $\mathcal{R}$:

$$\overline{\ell_j} \ \texttt{causes} \ \ell_k \ \texttt{if} \bigwedge_{\substack{l = 1, \ldots, m_i \\ l \neq j, l \neq k}} \overline{\ell_l} \tag{9}$$

■

Note that the causal relationships e `causes` r `if` Φ generated by this procedure do not exploit the general expressiveness in so far as condition Φ is, in any case, a conjunction of literals (c.f. (9)) rather than an arbitrary fluent formula. On the one hand, this does not imply that some causal information otherwise representable cannot be obtained by applying Definition 8, because any causal relationship can be transformed into an equivalent set of in this way restricted relationships. On the other hand, employing the general notion may lead to considerably more compact, albeit equivalent, representations. More sophisticated means to extract causal relationships from domain constraints without constructing normal forms should be developed to this end; yet, this goes beyond the scope of this paper.[11,12]

Example 1 (continued) Given domain constraint $D = sw_1 \wedge sw_2 \equiv \mathit{light}$ and influence information $\mathcal{I} = \{(sw_1, \mathit{light}), (sw_2, \mathit{light})\}$, by applying Definition 8 we obtain causal relationships as follows:

- The CNF of D is $(\overline{sw_1} \vee \overline{sw_2} \vee \mathit{light}) \wedge (sw_1 \vee \overline{\mathit{light}}) \wedge (sw_2 \vee \overline{\mathit{light}})$.
- As regards the first disjunct, $\overline{sw_1} \vee \overline{sw_2} \vee \mathit{light}$, we obtain the following:
  - While $(sw_1, sw_2) \notin \mathcal{I}$, we have $(sw_1, \mathit{light}) \in \mathcal{I}$, which yields

$$sw_1 \ \texttt{causes} \ \mathit{light} \ \texttt{if} \ sw_2$$

[11] To be more precise, the task would be to construct, for any two literals e and r such that $(|e|, |r|) \in \mathcal{I}$, a causal relationship e `causes` r `if` Ψ where Ψ is most simple in describing the circumstances under which effect e gives rise to indirect effect r. This could be achieved by collecting all causal relationships obtained via Definition 8 for e and r, i.e., e `causes` r `if` Φ_1, ..., e `causes` r `if` Φ_m, and taking as Ψ a most compact formula equivalent to $\Phi_1 \vee \ldots \vee \Phi_m$.

[12] A related challenge would be to find suitable deductive representations of the way domain constraints in conjunction with influence information give rise to causal relationships. This may, roughly and without going into details, look like $\forall S\,[\mathit{holds}(\Phi, S) \wedge \mathit{holds}(e, S) \supset \mathit{holds}(r, S)] \wedge \mathit{Infl}(|e|, |r|) \supset \mathit{causes}(e, r, \Phi)$. A detailed analysis of how to exploit such a characterization in particular action calculi is also left as future work.

– Analogously, while $(sw_2, sw_1) \notin \mathcal{I}$, we have $(sw_2, light) \in \mathcal{I}$, which yields

$$sw_2 \texttt{ causes } light \texttt{ if } sw_1$$

– Both $(light, sw_1) \notin \mathcal{I}$ and $(light, sw_2) \notin \mathcal{I}$.

- As regards the second disjunct, $sw_1 \vee \overline{light}$, we have
  – Since $(sw_1, light) \in \mathcal{I}$, we obtain

$$\overline{sw_1} \texttt{ causes } \overline{light} \texttt{ if } \top$$

  – $(light, sw_1) \notin \mathcal{I}$.
- Analogously, the third disjunct, $sw_2 \vee \overline{light}$, yields

$$\overline{sw_2} \texttt{ causes } \overline{light} \texttt{ if } \top$$

Altogether, we obtain exactly the four causal relationships listed in (5), which we have granted in the previous subsection to obtain the desired solution. ■

An important, general issue regarding the concept of causal relationships and their automatic generation is of course its complexity. Note that in the worst case exponentially many causal relationships have to be generated due to the potential explosion of the size of the domain constraints during the CNF construction and since up to quadratic many relationships exist for each resulting conjunct. Despite these pathological cases, there is, however, a decisive characteristic due to which especially in large domains the number of causal relationships is small compared to the worst case: The domain constraints do not interfere during this process. Generally, large domains tend to be locally structured in so far as each single domain constraint relates only a small amount of fluents. If we assume the size of a domain constraint (i.e., the number of fluent names involved) be fixed and small compared to their overall number then the number of causal relationships being constructed is *linear* wrt the number of constraints. For instance, recall the situation discussed at the beginning of this section, where a distinguished switch, sw_0, is assumed to affect n different sub-circuits each containing another switch, sw_i, and a bulb, $light_i$ ($1 \leq i \leq n$). The dependencies are described by n domain constraints $sw_0 \wedge sw_i \equiv light_i$. While compiling all indirect effects into action descriptions requires 2^{n+1} different descriptions, as argued above, only $4n$ causal relationships have to be generated for this domain. Note that it still pays regarding the computational effort when actually computing successor states since in any case at most n causal relationships have to be applied when toggling sw_0. Hence, we have avoided the exponential factor of this example in any respect.

Moreover, the fact that domain constraints do not interfere in determining causal relationships avoids the second crucial problem mentioned at the beginning: No existing causal relationship has to be modified or removed when new domain constraints are added.

3.3 Indirect Effects vs. Implicit Qualification

Thus far we have seen that and how domain constraints give rise to additional, indirect effects of actions. On the other hand, as observed e.g. in [Ginsberg and Smith, 1988b; Lin and Reiter, 1994], domain constraints might instead give rise to additional, implicit *qualifications* of actions. In the following, we illustrate that the concept of causal relationships along with the notion of potential influence perfectly accounts for this distinction.

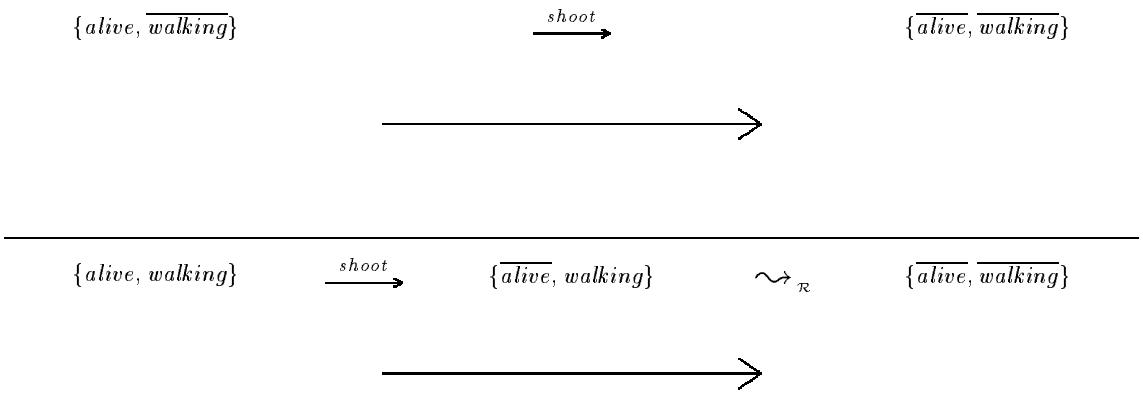

Figure 2: The application of $\langle\{alive\}, shoot, \{\overline{alive}\}\rangle$ to the first state depicted, where the turkey is not walking, results directly in a state that satisfies domain constraint $walking \supset alive$. In case the action description is applied to the second state, an additional ramification step based on (10) has to be computed in order to ensure the turkey stops walking when shot dead.

Example 2 Consider the following adaption, which is due to [Baker, 1991], of the Yale Shooting scenario [Hanks and McDermott, 1987]. We intend to hunt a turkey which is either alive or not (described via fluent *alive*) and which is walking around or not (fluent *walking*). Domain constraint $walking \supset alive$ restricts walking turkeys to vivid ones. Given the information that a change of *alive* might possibly influence the truth-value of *walking* but not vice versa, i.e., $\mathcal{I} = \{(alive, walking)\}$, this determines a single causal relationship, viz

$$\overline{alive} \texttt{ causes } \overline{walking} \texttt{ if } \top \tag{10}$$

Now, consider the action description $\langle\{alive\}, shoot, \{\overline{alive}\}\rangle$. Figure 2 illustrates the respective results of executing this action in the two states where its condition holds. While it is sufficient to compute only the direct effect in case $\{alive, \overline{walking}\}$, the underlying domain constraint gives rise to an additional, indirect effect (via (10)) in case $\{alive, walking\}$—not only does the turkey drop dead, it also stops walking.

On the contrary, consider action description $\langle\{\overline{walking}\}, entice, \{walking\}\rangle$, whose respective results when executed in the two states satisfying its condition are depicted in Figure 3. Again, in case $\{alive, \overline{walking}\}$ computing the direct effect suffices to obtain a state satisfying the domain constraint. Case $\{\overline{alive}, \overline{walking}\}$ is different: Applying the action description yields $\{\overline{alive}, walking\}$, which violates the domain constraint. Moreover, (10) is not applicable to the corresponding state-effect pair $(\{\overline{alive}, walking\}, \{walking\})$ since $\overline{alive}$ does not occur as effect. Hence, no successor state can be obtained according to Definition 5. In other words, our domain constraint gives rise to the additional, *implicit* qualification that a turkey must be alive in order to successfully entice it—which is exactly the intended solution. ■

In general, whenever no successor state exists according to Definition 5 then this hints at implicit qualifications of the action under consideration (c.f. the remark at the end of Subsection 3.1). Hence, providing the adequate information regarding potential causal influences between fluents automatically induces, by means of causal relationships, the distinction between ramification and qualification.

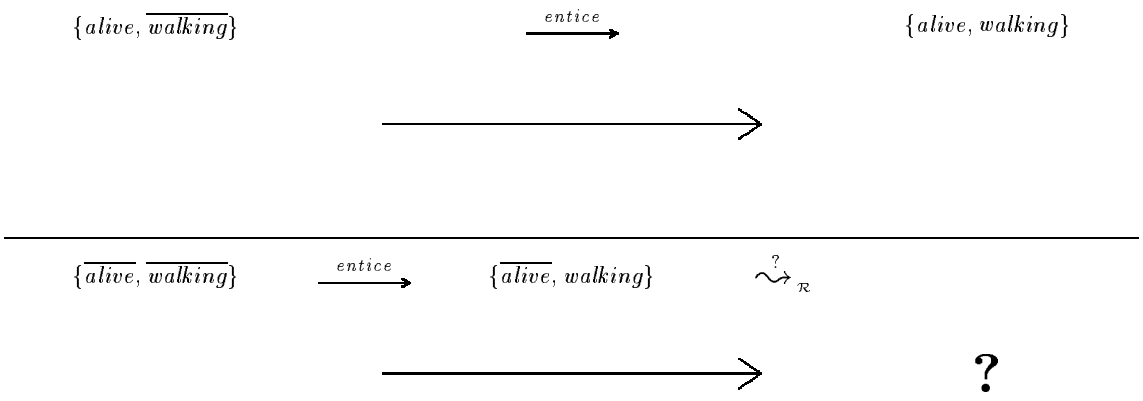

Figure 3: The application of $\langle\{\overline{walking}\}, entice, \{walking\}\rangle$ to the first state depicted, where the turkey is alive, results directly in a state that satisfies domain constraint $walking \supset alive$. In case the action description is applied to the second state, the result violates our domain constraint, which, moreover, cannot be 'corrected' by means of causal relationships. Hence, action *entice* cannot be successfully executed in a state where the turkey is not alive.

4 A Fixpoint Characterization of Ramifications

Causal relationships and their successive application after having computed the direct effects of an action constitute a mostly operational way to tackle the ramification problem. In this section, we relate our approach to a more static, fixpoint oriented characterization of successor states, which accounts for indirect effects, is based on the idea of minimizing change in conjunction with causal information, and which has been introduced in [McCain and Turner, 1995]. The aim is to prove that all successor states in the sense of the latter framework are also obtained by applying causal relationships.

As we have argued, an adequate solution to the ramification problem requires more sophisticated information of causal dependencies than provided by the mere domain constraints. This is why it is insufficient to simply consider all states that are as close as possible to the original one while satisfying both the direct effect of the action under consideration and the underlying domain constraints. Neither does this allow for preventing unintended changes nor does it enable us to distinguish between ramifications and qualifications (see Subsection 3.3). Based on this observation, the authors of [McCain and Turner, 1995] suggest to replace domain constraints by a suitable set of directed rules, called *causal rules*, which serve as deduction rules and are therefore weaker than the corresponding implications:

Definition 9 [McCain and Turner, 1995] Let $\mathcal{F}$ be a set of fluent names. A *causal rule* is an expression $\Phi \Rightarrow \Psi$ where Φ and Ψ are fluent formulas.

Let $\mathcal{C}$ be a finite set of causal rules. If Θ is a set of fluent formulas then by $\mathcal{T}_{\mathcal{C}}(\Theta)$ we denote the smallest set of fluent formulas containing Θ and being deductively closed under $\mathcal{C}$, i.e.,

1. $\Theta \subseteq \mathcal{T}_{\mathcal{C}}(\Theta)$;
2. for any θ such that $\mathcal{T}_{\mathcal{C}}(\Theta) \models \theta$ we have $\theta \in \mathcal{T}_{\mathcal{C}}(\Theta)$; and
3. if $\Phi \Rightarrow \Psi \in \mathcal{C}$ and $\Phi \in \mathcal{T}_{\mathcal{C}}(\Theta)$ then $\Psi \in \mathcal{T}_{\mathcal{C}}(\Theta)$.

If $\theta \in \mathcal{T}_{\mathcal{C}}(\Theta)$ then this denoted by $\Theta \vdash_{\mathcal{C}} \theta$. ■

Example 1 (continued) Consider causal rule $\mathcal{C} = \{sw_1 \wedge sw_2 \Rightarrow light\}$. If $\Theta = \{sw_1, sw_2\}$ then $\mathcal{T}_{\mathcal{C}}(\Theta)$ includes *light* since the given causal rule is applicable. In contrast, $\overline{sw_2} \notin \mathcal{T}_{\mathcal{C}}(\{sw_1, \overline{light}\})$ although $\overline{sw_2}$ follows classically from $sw_1 \wedge \overline{light} \wedge (sw_1 \wedge sw_2 \supset light)$. ■

Causal rules serve as basis for a fixpoint characterization of successor states which accounts for possible indirect effects. Informally, after having executed, in state S, an action with direct effect E, each possible successor state T must satisfy E, must be consistent with the set of causal rules, and each changing truth-value from S to T must be grounded on some causal rule. Thereby, the last requirement encodes the notion of minimal change:

Definition 10 [McCain and Turner, 1995] Let $\mathcal{F}$ be a set of fluent names, $\mathcal{A}$ a set of action descriptions, and $\mathcal{C}$ a set of causal rules. Furthermore, let S be a state and a an action name. If $\mathcal{A}$ contains an applicable (wrt S) action description for a and with effect E, then a state T is called *minimal change successor* iff

$$T = \{\ell : (S \cap T) \cup E \vdash_{\mathcal{C}} \ell\} \tag{11}$$

i.e., T is fixpoint of the function $\lambda T.\{\ell : (S \cap T) \cup E \vdash_{\mathcal{C}} \ell\}$ given S and E. ■

Example 1 (continued) An adequate set of causal rules for the Electric Circuit domain consists of these three elements:

$$\begin{array}{l} sw_1 \wedge sw_2 \Rightarrow light \\ \overline{sw_1} \Rightarrow \overline{light} \\ \overline{sw_2} \Rightarrow \overline{light} \end{array} \tag{12}$$

Let $S = \{\overline{sw_1}, sw_2, \overline{light}\}$ as depicted in Figure 1 then the only minimal change successor of applying $\langle\{\overline{sw_1}, toggle_1, \{sw_1\}\}\rangle$ is $T = \{sw_1, sw_2, light\}$: We have $(S \cap T) \cup E = \{sw_2\} \cup \{sw_1\}$, and the given causal rules allow to additionally derive *light*. In contrast, the unintended state $T' = \{sw_1, \overline{sw_2}, \overline{light}\}$ does not satisfy (11): $(S \cap T') \cup E = \{\overline{light}\} \cup \{sw_1\}$, which does not allow for deriving the missing literal, $\overline{sw_2}$.[13] ■

The following observation justifies the term "minimal change successor;" each state T satisfying (11) is as close as possible to S, i.e., there is no state with less (wrt set inclusion) changes that also satisfies E and $\mathcal{C}$:[14]

Observation 11 *Each minimal change successor is as close as possible to its predecessor.*

> **Proof:** Let $\mathcal{C}$ be a set of causal rules, S a state, E a set of fluent literals, and T, T' two minimal change successors. We have to show that if there exists some $\ell \in S \cap T$ such that $\overline{\ell} \in T'$ then also $S \cap T'$ contains an element not in T. Let $\ell \in S \cap T$ then (11) and T being consistent imply $(S \cap T) \cup E \nvdash_{\mathcal{C}} \overline{\ell}$. Given $\overline{\ell} \in T'$,

[13] It is also interesting to see why $T'' = \{sw_1, sw_2, \overline{light}\}$, where only the direct effect is computed, also does not satisfy (11): $(S \cap T'') \cup E = \{sw_2, \overline{light}\} \cup \{sw_1\}$ allows to additionally derive *light* given (12), i.e., T'' is not fixpoint. This illustrates that all formulas $\Phi \supset \Psi$ induced by causal relationships $\Phi \Rightarrow \Psi$ hold in minimal change successors.

[14] Observation 11 is a consequence of a theorem stated and proved in [McCain and Turner, 1995], which essentially relates Definition 10 to the basic definition of the so-called *possible models approach* [Winslett, 1988]. Below, we provide a more direct proof.

this together with the fact that $(S \cap T') \cup E \vdash_{\mathcal{C}} \overline{\ell}$ proves $S \cap T' \not\subseteq S \cap T$ since otherwise $\mathcal{T}_{\mathcal{C}}(S \cap T') \subseteq \mathcal{T}_{\mathcal{C}}(S \cap T)$. Hence, $S \cap T'$ includes an element that does not occur in T.

■

In what follows, we restrict attention to *non-disjunctive* causal rules $\Phi \Rightarrow \Psi$, where Ψ is a conjunction of literals.[15] Since $\Phi \Rightarrow \ell_1 \wedge \ldots \wedge \ell_n$ is equivalent to the n rules $\Phi \Rightarrow \ell_1, \ldots, \Phi \Rightarrow \ell_n$, we assume without loss of generality the consequent of a rule be a fluent literal. Moreover, when employing Equation (11) we only consider $\mathcal{T}_{\mathcal{C}}(\Theta)$ with Θ being a set of literals; hence, each causal rule of the form $\Phi_1 \vee \Phi_2 \Rightarrow \ell$ can equivalently be replaced by $\Phi_1 \Rightarrow \ell$ plus $\Phi_2 \Rightarrow \ell$, which allows us to assume each causal rule be of the form $\ell_1 \wedge \ldots \wedge \ell_m \Rightarrow \ell$ where $\ell, \ell_1, \ldots, \ell_m$ are fluent literals. In what follows, for notational convenience we will formally treat the antecedent of a causal rule, Φ, a set of literals and denote their conjunction by $\bigwedge \Phi$.

The main result of this section shall be a proof that each minimal change successor can be obtained by applying the approach developed in Section 3. Not only does this verify that our method covers all reasonable successor states with minimal distance from the original state, it also provides a means to actually *compute* minimal successor states—note that, following Equation (11), these states have to be guessed prior to testing whether they satisfy the condition of Definition 10.

In view of the intended result, we first present a pseudo-iterative characterization of minimal change successors and prove its adequacy:

Theorem 12 *Let $\mathcal{F}$ be a set of fluent names, $\mathcal{C}$ a set of causal rules, S a state, and E a set of literals. For each state T we define*

1. $\Gamma_0(S,T) := (S \cap T) \cup E$
2. $\Gamma_i(S,T) := \Gamma_{i-1}(S,T) \cup \{\ell : \Phi \Rightarrow \ell \in \mathcal{C}$ *and* $\Phi \subseteq \Gamma_i(S,T)\}$, *for* $i = 1, 2, \ldots$[16]

Then T is minimal change successor iff $T = \bigcup_{i=0}^{\infty} \Gamma_i(S,T)$.

Proof: We have to show $\{\ell : (S \cap T) \cup E \vdash_{\mathcal{C}} \ell\} = \bigcup_{i=0}^{\infty} \Gamma_i(S,T)$.

"*LHS*$\subseteq$*RHS*":

Let $\ell \in LHS$. In case $\ell \in (S \cap T) \cup E$, we find that $\ell \in \Gamma_0(S,T) \subseteq \bigcup_{i=0}^{\infty} \Gamma_i(S,T)$. Otherwise, $(S \cap T) \cup E \vdash_{\mathcal{C}} \ell$ implies the existence of a finite sequence of inference rules $\Phi_1 \Rightarrow \ell_1, \ldots, \Phi_n \Rightarrow \ell_n$ in $\mathcal{C}$ ($n \geq 1$) such that $\ell = \ell_n$ and, for each $1 \leq i \leq n$, $\Phi_i \subseteq (S \cap T) \cup E \cup \{\ell_1, \ldots, \ell_{i-1}\}$. Consequently, $\ell \in \Gamma_n(S,T) \subseteq \bigcup_{i=0}^{\infty} \Gamma_i(S,T)$.

"*LHS*$\supseteq$*RHS*":

By induction on i, we show $\Gamma_i(S,T) \subseteq \{\ell : (S \cap T) \cup E \vdash_{\mathcal{C}} \ell\}$. The base case, $i = 0$, holds by definition since $\Gamma_0(S,T) = (S \cap T) \cup E$. Now, let $\ell \in \Gamma_{i+1}(S,T)$ such that there exists some $\Phi \Rightarrow \ell \in \mathcal{C}$ where $\Phi \subseteq \Gamma_i(S,T)$. The induction hypothesis for $\Gamma_i(S,T)$ implies $(S \cap T) \cup E \vdash_{\mathcal{C}} \bigwedge \Phi$; hence, $(S \cap T) \cup E \vdash_{\mathcal{C}} \ell$.

■

This alternative characterization of minimal change successors forms the basis for proving the formal relation between this concept and the notion of causal relationships. To this end,

[15] A brief discussion concerning the nature of disjunctive causal rules can be found at the end of this section.
[16] Recall that Φ is considered a set of literals.

each causal rule $\Phi \Rightarrow \ell$ induces corresponding causal relationships and also domain constraint $\bigwedge \Phi \supset \ell$, which has to be satisfied by each state resulting from a successful application of a series of causal relationships. Aside from exploiting Theorem 12, the crucial point in the following proof is to ensure that whenever a causal rule is actually applied to justify an indirect effect then a corresponding causal relationship e `causes` r `if` Φ is also applicable, which especially requires e to occur in the respective set of already obtained (direct or indirect) effects E (recall that causal relationships operate on state-effect pairs (S, E)):

Theorem 13 *Let $\mathcal{F}$ be a set of fluent names, $\mathcal{A}$ a set of action descriptions, and $\mathcal{C}$ a set of causal rules. Furthermore, let $\mathcal{D} = \{\bigwedge \Phi \supset \ell : \Phi \Rightarrow \ell \in \mathcal{C}\}$ be a set of domain constraints and let $\mathcal{R}$ be a set of causal relationships containing for each $\{\varphi_1, \ldots, \varphi_n\} \Rightarrow \ell \in \mathcal{C}$ and each $1 \leq i \leq n$ the element*

$$\varphi_i \quad \texttt{causes} \quad \ell \quad \texttt{if} \quad \varphi_1 \wedge \ldots \wedge \varphi_{i-1} \wedge \varphi_{i+1} \wedge \ldots \wedge \varphi_n \tag{13}$$

Let S be a state which satisfies $\mathcal{D}$, and let $\langle C, a, E\rangle \in \mathcal{A}$ be an applicable action description then each minimal change successor T (wrt $\mathcal{C}$) is successor state (wrt $\mathcal{R}$ and $\mathcal{D}$).

Proof: We prove by induction that for each $i \in \mathbb{N}_0$ there exists a pair (S_i, E_i) such that $((S \setminus C) \cup E, E) \overset{*}{\rightsquigarrow}_{\mathcal{R}} (S_i, E_i)$ and $\Gamma_i(S, T) \subseteq S_i$ and $\Gamma_i(S, T) \setminus S \subseteq E_i$.[17] In what follows, for sake of readability we abbreviate $\Gamma(S, T)$ by Γ.

In case $i = 0$, $S_0 := (S \setminus C) \cup E$ and $E_0 := E$ satisfy the conditions: We have $\Gamma_0 = (S \cap T) \cup E \subseteq (S \setminus C) \cup E$ since $|C| = |E|$ and Γ_0 is consistent; furthermore, $\Gamma_0 \setminus S = ((S \cap T) \cup E) \setminus S \subseteq E$.

For the induction step let (S_i, E_i) satisfy the claim. Then, let

$$\{\ell_1, \ldots, \ell_n\} \;:=\; \{\ell : \ell \in \Gamma_{i+1} \text{ and } \ell \notin \Gamma_i\} \tag{14}$$

be the set of all literals that are added to Γ_i to obtain Γ_{i+1}. Hence, there exist n causal rules $\Phi_1 \Rightarrow \ell_1, \ldots, \Phi_n \Rightarrow \ell_n \in \mathcal{C}$ such that, for each $1 \leq j \leq n$, $\Phi_j \subseteq \Gamma_i$. Let us consider the first rule, $\Phi_1 \Rightarrow \ell_1$. From the induction hypothesis we conclude $\Gamma_i \subseteq S_i$ and, consequently, $\Phi_1 \subseteq S_i$. Moreover, we can find some $\varphi \in \Phi_1$ such that $\varphi \in E_i$: Assuming the contrary, i.e., $E_i \cap \Phi_1 = \{\}$, the induction hypothesis $\Gamma_i \setminus S \subseteq E_i$ implies $(\Gamma_i \setminus S) \cap \Phi_1 = \{\}$. But since $\Phi_1 \subseteq \Gamma_i$, this implies $\Phi_1 \subseteq S$; hence, $\ell_1 \in S$ (since S satisfies $\mathcal{D}$), i.e., $\ell_1 \in S \cap T \subseteq \Gamma_0$, which contradicts $\ell_1 \notin \Gamma_i$ (see (14)).

Thus, causal relationship φ `causes` ℓ_1 `if` $\bigwedge(\Phi_1 \setminus \{\varphi\})$ in $\mathcal{R}$ is applicable to (S_i, E_i) provided $\overline{\ell_1} \in S_i$. But S_i is a state, i.e., if it does not contain $\overline{\ell_1}$ it already contains ℓ_1 and the causal relationship need not be applied. Hence, either we can obtain $((S_i \setminus \{\overline{\ell_1}\}) \cup \{\ell_1\}, E_i \cup \{\ell_1\})$, or else we keep (S_i, E_i) and know $\ell_1 \in S_i$. To see that we can likewise successively proceed with all other literals, $\ell_2, \ldots, \ell_n$, observe first that the component E_i is grows monotonically. Thus, the only possible obstacle preventing us from applying the respective causal relationships for $j = 2, \ldots, n$, is the existence of some $\varphi \in \Phi_k$ $(2 \leq k \leq n)$ such that $\varphi \in \Gamma_i$ but $\varphi \notin (S_i \setminus \{\overline{\ell_1}, \ldots, \overline{\ell_{k-1}}\}) \cup \{\ell_1, \ldots, \ell_{k-1}\}$. In other words, causal relationship

[17] The last condition is used to ensure the aforementioned applicability of all relevant causal relationships as regards the set of currently obtained (direct or indirect) effects, E_i.

φ causes ℓ_k if $\bigwedge(\Phi_k \setminus \{\varphi\})$ was applicable to (S_i, E_i) but is not after first having computed $\ell_1, \ldots, \ell_{k-1}$. Yet $\varphi \notin (S_i \setminus \{\overline{\ell_1}, \ldots, \overline{\ell_{k-1}}\}) \cup \{\ell_1, \ldots, \ell_{k-1}\}$ means $\varphi \in \{\overline{\ell_1}, \ldots, \overline{\ell_{k-1}}\}$ since $\varphi \in \Gamma_i$ and, by induction hypothesis, $\Gamma_i \subseteq S_i$; hence, there exists some $\overline{\ell_j} \in \Gamma_i$ ($1 \leq j < k \leq n$). Due to $\ell_j \in \Gamma_{i+1}$, this contradicts $T \supseteq \Gamma_i \cup \Gamma_{i+1}$ being consistent.

To summarize, having successfully applied all n causal relationships (whenever necessary), we obtain the two sets $S_{i+1} := (S_i \setminus \{\overline{\ell_1}, \ldots, \overline{\ell_n}\}) \cup \{\ell_1, \ldots, \ell_n\}$ and $E_{i+1} := E_i \cup (\{\ell_1, \ldots, \ell_n\} \setminus S)$—which satisfy the claim: $\Gamma_{i+1} \subseteq S_{i+1}$ (due to $\Gamma_i \subseteq S_i$ and (14)) and $\Gamma_{i+1} \setminus S \subseteq E_{i+1}$ (due to $\Gamma_i \setminus S \subseteq E_i$ and (14)).

Now, since there exists only a finite number of changes from S to T, we have $\bigcup_{i=0}^{\infty} \Gamma_i = \Gamma_n$ for some $n \in \mathbb{N}_0$. Since $\Gamma_n = T$ is a state, $T \subseteq S_n$ implies $T = S_n$ and, consequently, $((S \setminus C) \cup E, E) \overset{*}{\rightsquigarrow}_{\mathcal{R}} (T, E_n)$, i.e., T is successor state.

■

Interestingly, the converse of this theorem does not hold, i.e., successor states might exist (in the sense of Definition 5) that cannot be obtained using the fixpoint-based approach. In Subsection 5.3, we argue that such states are perfectly reasonable, and failing to detect them with the approach discussed in this section is due to the policy of minimizing change, which thus might be too restrictive.

Finally, recall that our result is restricted to non-disjunctive causal rules. In the remainder of this section, we briefly discuss the nature of rules involving disjunctions in their consequent, such as in

$$\top \Rightarrow a \vee c \tag{15}$$

Typically, disjunctive rules are used to express non-deterministic behavior. For instance, given $S = \{\overline{a}, \overline{c}\}$, there exist two different minimal change successors wrt (15) and Definition 10 (assume $E = \{\}$), viz $T_1 = \{a, \overline{c}\}$ and $T_2 = \{\overline{a}, c\}$, respectively.[18] Note that $\{a, c\}$ is not a possible successor since merely having $a \vee c$ does not allow for concluding a nor c. The latter observation suggests that (15) could equivalently be replaced by these two non-disjunctive rules:

$$\begin{aligned} \overline{a} &\Rightarrow c \\ \overline{c} &\Rightarrow a \end{aligned} \tag{16}$$

which indeed yield the identical result when applied to state S from above. While this indicates that disjunctive rules can often be adequately represented by non-disjunctive ones, (15) and (16) are not generally equivalent when additional causal rules are considered. For example, if we add these two rules to (15):

$$\begin{aligned} a \vee c &\Rightarrow a \\ a \vee c &\Rightarrow c \end{aligned} \tag{17}$$

then $T = \{a, c\}$ is minimal change successor of $S = \{\overline{a}, \overline{c}\}$ since $\{\} \vdash_{\{(15),(17)\}} a \wedge c$. In contrast, no minimal change successor of S wrt $\{(16), (17)\}$ exists. Yet, this example lacks significance because antecedent and consequent of the causal rules in (17) share fluent names, which generally seems odd. Moreover, it is hard to imagine a more meaningful example since adding, say, $a \vee c \Rightarrow d$ instead of (17) does *not* cause a difference between (15) and (16). We tend

[18] To see why, take $S \cap T_1 = \{\overline{c}\}$, say, which, in conjunction with $a \vee c$ derived via (15), entails the missing literal, a.

to consider the latter observation hinting at the fact that even in principal each set consisting of arbitrary causal rules can easily be replaced by an equivalent set involving only non-disjunctive rules (obviously, $\{(15),(17)\}$ is equivalent to $\{\top \Rightarrow a \wedge c\}$). In any case, it should have become obvious that requiring non-disjunctive causal rules means no severe restriction.

5 The Necessity of Causal Relationships ...

We have seen that considering mere domain constraints when dealing with the ramification problem is generally not sufficient to avoid intuitively unexpected changes. As a solution, we have proposed causal relationships, which directly reflect the intuitive notion of causality in the domain being modeled. In this section, we contrast our proposal with other abstract concepts that are most widely used (often in slightly different variants) to tackle the problem of undesired indirect effects. Our aim is to illustrate the restrictive expressiveness of these concepts compared to our method.

5.1 ... Compared to Categorization-Based Approaches

The standard approach to avoid syntactically reasonable yet intuitively unexpected indirect effects, is to introduce some sort of categorization amongst the underlying set of fluent names. Such a distinction between different, typically two or three, kinds of fluents comes along with some specific notion of preference as regards changes in one category compared to changes in other ones when computing ramifications—or, less sophisticated, only a particular category is subject to the law of persistence etc. While a variety of names for such fluent classes circulate in literature,[19] the following, from our viewpoint artificial and questionable, assumption is fundamental for all methods based on categorization: Each fluent name can be uniquely assigned a single category. With a simple extension of our Electric Circuit domain, we will illustrate that the role of a fluent might be less obvious in this respect, which causes difficulties in finding a single appropriate category it belongs to. To this end, we employ the following, prototypical category-based definition:

Definition 14 Let $\mathcal{F}$ be a set of fluent names and $\mathcal{F}_p$ (*primary* fluents) and $\mathcal{F}_s$ (*secondary* fluents)[20] be two disjoint subsets such that $\mathcal{F}_p \dot{\cup} \mathcal{F}_s = \mathcal{F}$. If S, T_1, T_2 are states (wrt $\mathcal{F}$) then T is *closer to* S than T', written $T \prec_S T'$, iff

1. either $|T \setminus S| \cap \mathcal{F}_p \subsetneqq |T' \setminus S| \cap \mathcal{F}_p$
2. or $|T \setminus S| \cap \mathcal{F}_p = |T' \setminus S| \cap \mathcal{F}_p$ and $|T \setminus S| \cap \mathcal{F}_s \subsetneqq |T' \setminus S| \cap \mathcal{F}_s$.

Let $\mathcal{D}$ be a set of domain constraints and $\mathcal{A}$ a set of action descriptions. If S is a state and a an action description then a state T is *successor* iff there exists some $\langle C, a, E\rangle \in \mathcal{A}$ such that $C \subseteq S$ and the following holds:

1. $E \subseteq T$;
2. T satisfies $\mathcal{D}$; and

[19] E.g., *frame* vs. *non-frame* fluents [Lifschitz, 1990]; *relevant* vs. *dependent* [Brewka and Hertzberg, 1993]; *persistent* vs. *non-persistent* [del Val and Shoham, 1993]; the latter augmented by *mutable* fluents [Zhang and Foo, 1993]; or *occluded*, *remanent*, and *dependent* [Sandewall, 1995a].

[20] The terms "primary" and "secondary," respectively, were inspired by [Sandewall, 1995b].

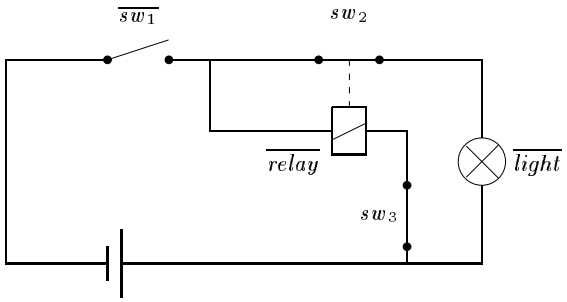

Figure 4: An extended electric circuit described by five fluents. The current state is denoted by $\overline{sw_1}$ (the first switch is off), sw_2 (the second switch is on), sw_3 (the third switch is on), $\overline{light}$ (the light bulb is off) and $\overline{relay}$ (the relay is deactivated).

---

3. there is no $T' \prec_S T$ such that $E \subseteq T'$ and T' satisfies $\mathcal{D}$.

■

In words, state T is closer to S than T' iff S and T differ in less (wrt set inclusion) primary fluents than S and T' do, or else S, T and S, T' differ in the same way on primary fluents but S and T differ in less secondary fluents than S and T' do. For instance, to prefer a change of the light bulb's state compared to a switch magically jumping its position in Example 1, we consider sw_1, sw_2 primary and $light$ secondary. Then the application of $\langle\{\overline{sw_1}\}, toggle_1, \{sw_1\}\rangle$ to state $S = \{\overline{sw_1}, sw_2, \overline{light}\}$ admits, as intended, $T = \{sw_1, sw_2, light\}$ as unique successor since $T \prec_S \{sw_1, \overline{sw_2}, \overline{light}\}$.

Consider, now, the following extension of our electric circuit (see also Figure 4):

Example 3 We augment Example 1 by introducing a third switch, named sw_3, plus a relay, named $relay$. If activated, the relay is intended to force the second switch (sw_2) to jump off. It is controlled by the first and third switch. Formally, the dependencies between all these components are described by the following domain constraints:

$$\begin{aligned} sw_1 \wedge sw_2 &\equiv light \\ sw_1 \wedge sw_3 &\equiv relay \\ relay &\supset \overline{sw_2} \end{aligned} \tag{18}$$

In order to find an adequate partition of all involved fluent names into primary and secondary fluents, respectively, observe first that we should have $sw_1, sw_2 \in \mathcal{F}_p$ and $light \in \mathcal{F}_s$ as above since whenever one of these two switches changes its position, we prefer a change of $light$ instead of a change of the other switch (as regards the first domain constraint). Analogously, we should have $sw_1, sw_3 \in \mathcal{F}_p$ and $relay \in \mathcal{F}_s$ since whenever one of these two switches changes its position, we prefer a change of $relay$ instead of a change of the other switch (as regards the second domain constraint). Hence, we obtain $\mathcal{F}_p = \{sw_1, sw_2, sw_3\}$ and $\mathcal{F}_s = \{light, relay\}$.

On this basis, let us investigate the particular state depicted in Figure 4. The expected result of toggling the first switch, sw_1, is that the relay becomes activated, which in turn

causes the second switch, sw_2, jumping its position; hence, the light bulb stays off.[21] Indeed, according to Definition 14, given state $S = \{\overline{sw_1}, sw_2, sw_3, \overline{relay}, \overline{light}\}$ and action description $\langle\{\overline{sw_1}\}, toggle_1, \{sw_1\}\rangle$, the corresponding state, $\{sw_1, \overline{sw_2}, sw_3, relay, \overline{light}\}$, is successor: Aside from direct effect $\{sw_1\}$, the above domain constraints suggest that a second primary fluent must change its state since any state with all switches on violates (18). However, the observation that a second primary fluent has to be changed suggests another successor state candidate, namely, where sw_3 changes its truth-value instead of sw_2: The reader is invited to verify that state $\{sw_1, sw_2, \overline{sw_3}, \overline{relay}, light\}$ also satisfies the conditions of Definition 14 as it does not violate the domain constraints and has minimal distance to S. Hence, we obtain a second successor state where the third switch magically opens, the relay remains deactivated and the light bulb turns on. ■

The reason for the unexpected second state to occur in this example is that we necessarily fail to assign a unique, appropriate category to fluent sw_2, whose role is twofolded: On the one hand, it should be considered primary (regarding the sub-circuit involving sw_1 and *light*), and on the other hand, it behaves like a secondary fluent (as regards the sub-circuit containing the relay). One might suggest that this particular example could be modeled by just introducing an additional category of, say, *tertiary* fluents, $\mathcal{F}_t$, which have even lower priority than secondary fluents. Then, assigning $\mathcal{F}_p = \{sw_1, sw_3\}$, $\mathcal{F}_s = \{sw_2, relay\}$, $\mathcal{F}_t = \{light\}$ and extending Definition 14 appropriately yields the expected unique resulting state. However, aside from the somehow strange categorization, where similar entities, namely switches, belong to different categories, this particular classification requires a deeper analysis of possible direct and indirect effects in the electric circuit and is far from being intuitively plausible. Moreover, it is not hard to imagine more complex domains requiring more and more categories, which heavily increases the difficulty of deciding to which class a particular fluent name should belong.

In contrast, since causal relationships in conjunction with influence information only describe local phenomena, they can easily deal with fluents who behave differently regarding different domain constraints:

Example 3 (continued) The possible influences in the electric circuit depicted in Figure 4 are represented by this relation:

$$\mathcal{I} = \{(sw_1, light), (sw_2, light), (sw_1, relay), (sw_3, relay), (relay, sw_2)\}$$

Most importantly, this information concentrates on direct influences only, which, as we shall see, is sufficient; nothing has to be stated about the possibility that sw_1 might indirectly influence sw_2 (through the relay). Note also that sw_2 occurs as first and second argument in $\mathcal{I}$, which encodes its twofolded nature in this example. Applying Definition 8 to domain constraints (18)

[21] It might however happen that the light bulb turns on for a very short period of time, depending on the time it takes to activate the relay and to affect the second switch. Nonetheless, the light is definitely off in the resulting state.

and $\mathcal{I}$ yields the following nine causal relationships:

$$\begin{array}{llllll} sw_1 & \texttt{causes} & light & \texttt{if} & sw_2 & \qquad sw_2 \ \texttt{causes} \ light \ \texttt{if} \ sw_1 \\ \overline{sw_1} & \texttt{causes} & \overline{light} & \texttt{if} & \top & \qquad \overline{sw_2} \ \texttt{causes} \ \overline{light} \ \texttt{if} \ \top \end{array}$$

$$\begin{array}{llllll} sw_1 & \texttt{causes} & relay & \texttt{if} & sw_3 & \qquad sw_3 \ \texttt{causes} \ relay \ \texttt{if} \ sw_1 \\ \overline{sw_1} & \texttt{causes} & \overline{relay} & \texttt{if} & \top & \qquad \overline{sw_3} \ \texttt{causes} \ \overline{relay} \ \texttt{if} \ \top \end{array}$$

$$relay \ \texttt{causes} \ \overline{sw_2} \ \texttt{if} \ \top$$

The topmost four relationships are obtained from domain constraint $sw_1 \wedge sw_2 \equiv light$ as described in Subsection 3.2; domain constraint $sw_1 \wedge sw_3 \equiv relay$ yields, in a similar way, the next four relationships; and, finally, from $relay \supset \overline{sw_2}$ we obtain the last relationship due to $(relay, sw_2) \in \mathcal{I}$.

Now, given the state depicted in Figure 4, $S = \{\overline{sw_1}, sw_2, sw_3, \overline{relay}, \overline{light}\}$, and action description $\langle\{\overline{sw_1}, toggle_1, \{sw_1\}\rangle$, the starting point for the application of causal relationships is the state-effect pair

$$(\,\{sw_1, sw_2, sw_3, \overline{relay}, \overline{light}\}\,,\,\{sw_1\}\,) \tag{19}$$

There are two possible directions to proceed. First, we can apply sw_1 `causes` $relay$ `if` sw_3 followed by $relay$ `causes` $\overline{sw_2}$ `if` $\top$, which results in

$$(\,\{sw_1, \overline{sw_2}, sw_3, relay, \overline{light}\}\,,\,\{sw_1, relay, \overline{sw_2}\}\,)$$

The first argument satisfies the underlying domain constraints and, consequently, denotes a successor state (which represents exactly what we expect in this example). In addition, there is a second, larger chain of applicable causal relationships, which, however, comes to the very same conclusion as regards the resulting state, namely,

$$\begin{array}{rlll} sw_1 & \texttt{causes} & light & \texttt{if} \ sw_2 \\ sw_1 & \texttt{causes} & relay & \texttt{if} \ sw_3 \\ relay & \texttt{causes} & \overline{sw_2} & \texttt{if} \ \top \\ \overline{sw_2} & \texttt{causes} & \overline{light} & \texttt{if} \ \top \end{array} \tag{20}$$

In words, we first conclude the light bulb turns on due to the second switch being on. However, since the activation of the relay causes sw_2 to become false, we have to 'turn off' the light bulb again via the finally applied causal relationship. Thus, we obtain the pair

$$(\,\{sw_1, \overline{sw_2}, sw_3, relay, \overline{light}\}\,,\,\{sw_1, light, relay, \overline{sw_2}, \overline{light}\}\,)$$

which also contains the intended resulting state. The two derivations correspond to the two possible ways the circuit might behave shortly after having toggled the first switch (see also Footnote 21). Both end in the same state, and there is no other way to apply causal relationships to (19). In particular, the unintended solution where the third switch magically opens is not producible. ■

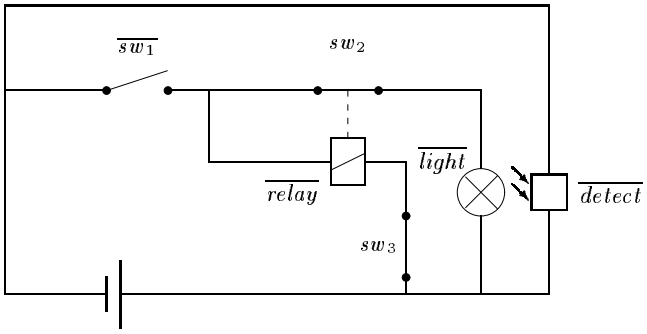

Figure 5: The extended electric circuit augmented by a device, represented by *detect*, which registers an activation of the light bulb (such a device combines a phototransistor and flipflop). The current state is described as in Figure 4 plus $\overline{detect}$ (no action of light has occurred).

5.2 ... Compared to the Minimal Change Policy

A widely accepted assumption concerning the ramification problem says that generating indirect effects ought to satisfy the property of minimal change. Regardless of possible categorizations as discussed in the previous section, whenever a 'proper' successor state is strictly closer to the original state than another 'proper' successor state then the latter is rejected. While the result proved in Section 4 shows that our method covers all states that confess to the minimal change policy, it is not restricted in this respect. The following example shows the importance of this in so far as requiring minimal changes might fail to obtain all intuitively expected possible successor states. It is motivated by the observation that the circuit in Example 3 behaves, internally, non-deterministically, though only a single successor state results. This aspect of indeterminism is now made explicit.

Example 4 The extended electric circuit from Example 3 is further augmented by a light detecting device (fluent name *detect*) that becomes (and stays) activated as soon as the light bulb turns on (c.f. Figure 5). This amounts to augmenting domain constraints (18) by

$$light \supset detect$$

Enhancing influence information $\mathcal{I}$ by $(light, detect)$ then results in this additional causal relationship:

$$light \texttt{ causes } detect \texttt{ if } \top \tag{21}$$

When the first switch, sw_1, is toggled in the state depicted in Figure 5 then we would expect two possible successor states due to the non-deterministic behavior of the circuit: In any case, we end up with the relay activated and both the second switch and the light bulb off. Yet, the complete outcome depends on whether or not the activation of the relay and its affecting the second switch is faster than the intermediate activation of the light bulb and, triggered by this, the activation of the photo-device. Since *detect* remains true once activated, it might happen that this fluent additionally becomes true. The available causal relationships support this conclusion. Let $S = \{\overline{sw_1}, sw_2, sw_3, \overline{relay}, \overline{light}, \overline{detect}\}$. Application of $\langle\{\overline{sw_1}\}, toggle_1, \{sw_1\}\rangle$ yields

$$(\,\{sw_1, sw_2, sw_3, \overline{relay}, \overline{light}, \overline{detect}\}\,,\,\{sw_1\}\,)$$

As in the preceding subsection, we may obtain

$$(\{sw_1, \overline{sw_2}, sw_3, relay, \overline{light}, \overline{detect}\} , \{sw_1, relay, \overline{sw_2}\})$$

or

$$(\{sw_1, \overline{sw_2}, sw_3, relay, \overline{light}, \overline{detect}\} , \{sw_1, light, relay, \overline{sw_2}, \overline{light}\})$$

both corresponding to the first of the expected successor states. But we can also integrate Relationship (21) into derivation sequence (20)—somewhere in between the first and last element—which results in the state-effect pair

$$(\{sw_1, \overline{sw_2}, sw_3, relay, \overline{light}, detect\} , \{sw_1, light, relay, detect, \overline{sw_2}, \overline{light}\})$$

whose first component satisfies all domain constraints and corresponds to the second expected successor state. Note that this state differs from S in strictly more fluent values than the first one does.

To see why no minimal change-based formalism can possibly obtain this, consider these causal rules $\mathcal{C}$:

$$\begin{array}{lll} sw_1 \wedge sw_2 \Rightarrow light & \overline{sw_1} \Rightarrow \overline{light} & \overline{sw_2} \Rightarrow \overline{light} \\ sw_1 \wedge sw_3 \Rightarrow relay & \overline{sw_1} \Rightarrow \overline{relay} & \overline{sw_3} \Rightarrow \overline{relay} \\ relay \Rightarrow \overline{sw_2} & light \Rightarrow detect & \end{array}$$

in conjunction with S as above and $E = \{sw_1\}$. While $T = \{sw_1, \overline{sw_2}, sw_3, relay, \overline{light}, \overline{detect}\}$ satisfies Equation (11), $T' = \{sw_1, \overline{sw_2}, sw_3, relay, \overline{light}, detect\}$ is not fixpoint since

$$(S \cap T') \cup E = \{sw_3, \overline{light}\} \cup \{sw_1\} \not\vdash_{\mathcal{C}} detect$$

■

The last observation suggests that generally minimization might not be a concept adequate to distinguish between possible indirect effect on the one hand, and unfounded changes on the other hand. In fact, the aim of calculating ramifications is not to minimize change but to avoid changes that are not caused, which, as we have seen, is not necessarily identical.

5.3 ... Compared to Causal Rules

Domain constraints, as argued in Subsection 3.3, may give rise to implicit qualifications rather than indirect effects. Example 2 illustrated that sometimes even a single constraint acts in both fashions, depending on which effect actually occurs. Both causal relationships and causal rules allow for modeling this distinction.[22] However, the expressiveness of the latter concept in this respect is limited compared to the former one. The reason is that it is impossible to restrict applicability of a causal rule like $\ell_1 \wedge \ell_2 \Rightarrow \ell$, say, to situations where ℓ_1 occurs as effect (causing ℓ as ramification in case ℓ_2 being true), while ℓ_2 occurring as effect (with ℓ_1 being true and ℓ being false) shall indicate an implicit qualification. In contrast, causal relationships support this sophistication,[23] which is required in the following example.

[22] Regarding Example 2 with domain constraint $walking \supset alive$, this is achieved by taking causal relationship $\overline{alive}$ `causes` $\overline{walking}$ `if` $\top$ but not $walking$ `causes` $alive$ `if` $\top$; similarly, one would employ causal rule $\overline{alive} \Rightarrow \overline{walking}$ but not $walking \Rightarrow alive$.

[23] This is why n different relationships are needed to represent a rule with n literals in its antecedent (c.f. (13) in Theorem 13).

Example 5 Let us consider a more subtle, ancient method to hunt the turkey, namely, by using a (manually activated) trapdoor. The state of this trapdoor is described via fluent name *trapdoor-open* and, aside from being alive or not, the turkey is either in the dangerous zone or not (fluent name *at-trap*). The ground underneath the trapdoor is designed such that if the turkey finds itself being *at-trap* and the trapdoor is open then it cannot be alive, which is represented by the following domain constraint:

$$\textit{at-trap} \wedge \textit{trapdoor-open} \supset \overline{\textit{alive}} \tag{22}$$

We can open the trapdoor via $\langle\{\overline{\textit{trapdoor-open}}\}, \textit{open}, \{\textit{trapdoor-open}\}\rangle$ and entice the turkey via $\langle\{\overline{\textit{at-trap}}, \textit{alive}\}, \textit{entice}, \{\textit{at-trap}, \textit{alive}\}\rangle$. While the state of the trapdoor can possibly influence the victim's state of being alive or not, the turkey is somehow alert in so far as it would never kill itself by moving towards the open trapdoor, i.e., *at-trap* shall not possibly influence *alive* . The latter is intended to give rise to implicit qualification $\overline{\textit{trapdoor-open}}$ for *entice* . Hence, the adequate influence information is $\mathcal{I} = \{(\textit{trapdoor-open}, \textit{alive})\}$,[24] which, in conjunction with (22), determines a single causal relationship according to Definition 8, namely,

$$\textit{trapdoor-open} \quad \texttt{causes} \quad \overline{\textit{alive}} \quad \texttt{if} \quad \textit{at-trap} \tag{23}$$

Given state $S = \{\textit{alive}, \textit{at-trap}, \overline{\textit{trapdoor-open}}\}$ (say, after having enticed the turkey), executing *open* yields $\{\textit{alive}, \textit{at-trap}, \textit{trapdoor-open}\}$ as intermediate state, which violates our domain constraint; yet, *trapdoor-open* occurred as effect, which is why (23) is applicable, resulting in the expected state $\{\overline{\textit{alive}}, \textit{at-trap}, \textit{trapdoor-open}\}$. In contrast, consider state $S = \{\textit{alive}, \overline{\textit{at-trap}}, \textit{trapdoor-open}\}$ and action *entice* , whose execution yields intermediate state $\{\textit{alive}, \textit{at-trap}, \textit{trapdoor-open}\}$ also. But now (23) is not applicable since *trapdoor-open* did not occur as effect, i.e., no successor state exists. In other words, $\overline{\textit{trapdoor-open}}$ is an additional, implicit qualification for *entice* , which is exactly the intended result. Note that we are only able to distinguish these two cases by employing (23) but not the analogous causal relationship *at-trap* `causes` $\overline{\textit{alive}}$ `if` *trapdoor-open* . For both correspond to an identical causal rule, namely $\textit{at-trap} \wedge \textit{trapdoor-open} \Rightarrow \overline{\textit{alive}}$, this distinction goes beyond the expressiveness of causal rules. ■

6 A Calculus

Having presented, thoroughly discussed, and demonstrated the benefits of our general approach to the ramification problem compared to others, the second part of the paper is devoted to the development of a suitable, concrete calculus. This calculus will be based on the logic programming paradigm. More precisely, we adapt and extend a method described in [Hölldobler and Schneeberger, 1990; Hölldobler and Thielscher, 1995], which applies the concept of *reification* to entire states, i.e., each of which is formally represented as single term and, thus, is manipulable by means of program clauses. The adequate treatment of these terms requires a (domain-independent) equational theory, which, essentially, formalizes crucial properties of the datastructure set. We assume the reader be familiar with basic concepts of logic programs and the negation-as-failure principle, as can be found e.g. in the textbook [Lloyd, 1987].

[24] Since the trapdoor does not work automatically, we do not consider $(\textit{at-trap}, \textit{trapdoor-open}) \in \mathcal{I}$, and $(\textit{trapdoor-open}, \textit{at-trap}) \notin \mathcal{I}$ is due to the assumption that the turkey has no time to escape during the process of opening the door.

6.1 Reified States

While the atomic elements of state descriptions have been restricted to propositional constants in the first part of the paper for sake of simplicity, we employ more complex a notion of fluents in the second part. A fluent is now an n-place relation over given objects (see, e.g., [Sandewall, 1994; Kartha and Lifschitz, 1994]), called *entities* here. This comes along with both a generalized concept of action descriptions and fluent formulas involving quantifications. Yet, by requiring underlying sets of entities be finite, we still guarantee decidability in any respect. The following definition extends Definition 1:

Definition 15 Let $\mathcal{O}$ be a finite set of symbols called *entities*. Let $\mathcal{F}$ denote a finite set of fluent names, each of which is associated with a natural number called *arity*. A *fluent* is an expression $f(o_1,\ldots,o_n)$ where $f \in \mathcal{F}$ of arity n and $o_1,\ldots,o_n \in \mathcal{O}$. A *fluent literal* is a fluent or its negation denoted by $\overline{f(o_1,\ldots,o_n)}$.

Let $\mathcal{V}$ be a denumerable set of *variables*. An expression $f(t_1,\ldots,t_n)$ and its negation $\overline{f(t_1,\ldots,t_n)}$ are called *fluent expressions* iff $f \in \mathcal{F}$ of arity n and $t_i \in \mathcal{O} \cup \mathcal{V}$ $(1 \leq i \leq n)$. ■

As before, a *state* is a maximal consistent set of fluent literals. For sake of readability, from now on we implicitly assume an arbitrary but fixed underlying set $\mathcal{O}$ of entities, a set $\mathcal{F}$ of fluent names, and a set $\mathcal{V}$ of variables, respectively.

We follow the PROLOG convention in denoting variables by uppercase letters, sometimes with sub- or superscripts. The expression $\widetilde{X}$ (resp. $\widetilde{o}$) denotes a finite sequence of variables (resp. entities) of arbitrary but fixed length. A *substitution* $\theta : \mathcal{V} \mapsto \mathcal{V} \cup \mathcal{O}$ maps variables to variables or entities. A substitution is often written $\{X_1 \mapsto t_1,\ldots,X_n \mapsto t_n\}$ expressing that $\theta(X_i) = t_i$ for $X_i \in \{X_1,\ldots,X_n\}$ and $\theta(X) = X$ otherwise. The *application* of a substitution θ to some expression ξ, written $\xi\theta$, amounts to replacing all variables X occurring in ξ by $\theta(X)$. If $\widetilde{X}$ is a sequence of all variables occurring free in expression ξ then this is written $\xi[\widetilde{X}]$. Let $\widetilde{X} = X_1,\ldots,X_n$ then a *ground instance* of such an expression is obtained by applying a substitution $\theta = \{X_1 \mapsto o_1,\ldots,X_n \mapsto o_n\}$ to ξ where $o_1,\ldots,o_n \in \mathcal{O}$, and if $\widetilde{o} = o_1,\ldots,o_n$ then $\xi[\widetilde{X}]\theta$ is also denoted by $\xi[\widetilde{o}]$.

Based on the extended notion of fluent, action descriptions may now contain variables and, then, are considered representatives for all of their ground instances:

Definition 16 An *action description* is a triple $\langle C[\widetilde{X}], a(\widetilde{X}), E[\widetilde{X}]\rangle$ where $C[\widetilde{X}]$ and $E[\widetilde{X}]$ are sets of fluent expressions and action name a is a symbol. It is assumed that $|C[\widetilde{o}]| = |E[\widetilde{o}]|$ for any sequence $\widetilde{o}$ of entities.

If S is a state then a ground instance $\alpha[\widetilde{o}]$ of action description $\alpha[\widetilde{X}] = \langle C[\widetilde{X}], a(\widetilde{X}), E[\widetilde{X}]\rangle$ is *applicable* in S iff $C[\widetilde{o}] \subseteq S$. The *application* of $\alpha[\widetilde{o}]$ to S yields $(S \setminus C[\widetilde{o}]) \cup E[\widetilde{o}]$. Any set $\mathcal{A}$ of action descriptions is assumed to contain at most one applicable description for each ground instance $\alpha(\widetilde{o})$ and each state S. ■

Example 1 (continued) Let us exploit the extended expressiveness to model the Electric Circuit domain using the two entities $\mathcal{O} = \{s_1, s_2\}$ representing the two switches, along with the unary fluent *on* denoting the position of its argument plus the nullary fluent *light* denoting the state of the light bulb as before. The current state displayed in Figure 1 is then encoded by $S = \{\overline{on(s_1)}, on(s_2), \overline{light}\}$.

On the basis of this representation, we define an action called $toggle(X)$ by the following two descriptions:

$$\langle\, \{\overline{on(X)}\}\,,\, toggle(X)\,,\, \{on(X)\}\,\rangle \qquad \langle\, \{on(X)\}\,,\, toggle(X)\,,\, \{\overline{on(X)}\}\,\rangle \tag{24}$$

When executing, say, $toggle(s_1)$ in S then the instance $\theta = \{X \mapsto s_1\}$ of the first action description is applicable due to $\{\overline{on(X)}\}\theta \subseteq S$; its application yields

$$(S \setminus \{\overline{on(s_1)}\}) \cup \{on(s_1)\} = \{on(s_1), on(s_2), \overline{light}\}$$

■

The approach defined in [Hölldobler and Schneeberger, 1990; Hölldobler and Thielscher, 1995] is grounded on the formal representation of an entire state by a single term. To this end, fluent expressions are reified and connected via a special binary function, which is illustratively denoted by $\circ$ and written in infix notation. As an example, the term representation of $S = \{\overline{on(s_1)}, on(s_2), \overline{light}\}$ is

$$(\,\overline{on(s_1)} \circ on(s_2)\,) \circ light$$

where the bar denoting negative fluent expressions should be taken formally a unary function. Intuitively, the order in which the various fluent expressions are connected is irrelevant as regards the state to be represented. Hence, our connection function obeys some special properties, which are formalized using the following equational theory [Hölldobler and Schneeberger, 1990]:

$$\begin{array}{rrcll} \forall X,Y,Z. & (X \circ Y) \circ Z & = & X \circ (Y \circ Z) & \text{(associativity)} \\ \forall X,Y. & X \circ Y & = & Y \circ X & \text{(commutativity)} \\ \forall X. & X \circ \emptyset & = & X & \text{(unit element)} \end{array}$$

where the special constant $\emptyset$ denotes a unit element for $\circ$. These three axioms (short: AC1) are used as equational theory underlying our logic program. In what follows, we write $=_{\mathrm{AC1}}$ to illustrate that equality should always be related to the axioms above. Due to the law of associativity, we are allowed to omit parenthesis on the level of $\circ$. Note that (AC1) models essential properties of the datastructure "set." For formal reasons, we introduce a function τ which maps sets of fluent expressions $F = \{t_1, \ldots, t_n\}$ to their term representation, $\tau_F = t_1 \circ \cdots \circ t_n$ (including $\tau_{\{\}} = \emptyset$).

The concept of reifying entire states allows for a most flexible way to manipulate them via first-order formulas and, more specifically, via logic program clauses. To this end, action descriptions are encoded as unary clauses based on a ternary predicate called *action*. Let $\mathcal{A} = \{\langle C_1[\widetilde{X_1}], a_1[\widetilde{X_1}], E_1[\widetilde{X_1}]\rangle, \ldots, \langle C_n[\widetilde{X_n}], a_n[\widetilde{X_n}], E_n[\widetilde{X_n}]\rangle\}$ be a finite set of action descriptions, then these are transformed into the following n program clauses:

$$\begin{array}{l} action(\tau_{C_1[\widetilde{X_1}]}, a_1[\widetilde{X_1}], \tau_{E_1[\widetilde{X_1}]}). \\ \qquad\vdots \\ action(\tau_{C_n[\widetilde{X_n}]}, a_n[\widetilde{X_n}], \tau_{E_n[\widetilde{X_n}]}). \end{array} \tag{25}$$

The application of action descriptions according to Definition 16 is then modeled by this clause, where $result(\tau_S, a[\widetilde{o}], \tau_{S'})$ shall be derivable iff executing $a[\widetilde{o}]$ in S yields S':

$$\begin{array}{rcl} result(S, A, S') & \leftarrow & action(C, A, E), \\ & & S =_{\mathrm{AC1}} C \circ V, \\ & & S' =_{\mathrm{AC1}} V \circ E. \end{array} \tag{26}$$

In words, $result(\tau_S, a[\widetilde{o}], \tau_{S'})$ is true if an action description for $a[\widetilde{o}]$ exists with condition $C[\widetilde{o}]$ and effect $E[\widetilde{o}]$ such that

1. The unification problem $\tau_S =_{\mathrm{AC1}} \tau_{C[\tilde{o}]} \circ V$, where V is a new variable, has a solution. This models testing $C[\tilde{o}] \subseteq S$.

2. Term $\tau_{S'}$ equals term $V \circ \tau_{E[\tilde{o}]}$ (wrt (AC1)). Since a side effect of solving the aforementioned unification problem is that, in case of success, V becomes bound to all sub-terms in S which are not amongst the condition, $C[\tilde{o}]$, this models testing $S' = (S \setminus C[\tilde{o}]) \cup E[\tilde{o}]$.

Note that this formalization avoids additional axioms (so-called *frame axioms* [McCarthy and Hayes, 1969; Green, 1969]) to express the law of persistence since all fluent literals unaffected by the execution of an action automatically continue to be true in the resulting state.

The adequate computation mechanism for the equational logic program $(25), (26); (\mathrm{AC1})$ [25] is *SLDE-resolution* [Plotkin, 1972; Jaffar *et al.*, 1984; Gallier and Raatz, 1989; Hölldobler, 1989] where, in contrast to ordinary SLD-resolution, the standard unification procedure is replaced by an algorithm that unifies wrt an underlying equational theory, i.e., in our case, wrt (AC1).[26] In conjunction with this resolution principle, the program is sound and complete as regards Definition 16 (see [Hölldobler and Thielscher, 1995]):

Theorem 17 *Let $\mathcal{A}$ be a set of action description, S, S' two states, and $a\theta$ be ground instance of an action. Then there exists an action description $\langle C, a, E\rangle \in \mathcal{A}$ such that $C\theta \subseteq S$ and $S' = (S \setminus C\theta) \cup E\theta$ iff*

$$\leftarrow \mathit{result}(\tau_S, a\theta, \tau_{S'})$$

has an SLDE-refutation wrt $(25)_{\mathcal{A}}, (26); (\mathrm{AC1})$.

In view of extending the basic program for to model our approach to the ramification problem, we first develop a suitable encoding of fluent formulas and the corresponding notion of validity wrt states. To this end, the following definition extends the basic concepts of fluent formulas used in the first part, allowing for more complex formulas involving quantifiers:

Definition 18 The set of *fluent formulas* is inductively defined as follows: Each fluent expression and $\top$ and $\bot$ are fluent formulas, and if F and G are fluent formulas so are $F \wedge G$, $F \vee G$, $F \supset G$, $F \equiv G$, $\exists X.\, F$, and $\forall X.\, F$ (where $X \in \mathcal{V}$).

A *closed* formula is a fluent formula without free variables, i.e., where each variable occurring in the formula is bound by some quantifier. For sake of simplicity, it is assumed that within a fluent formula different quantifiers always bind different variables. Let S be a state and F a closed fluent formula then the notion of F being *true* in S, written $S \models F$, is inductively defined as follows:

1. $S \models \top$ and $S \not\models \bot$;

2. $S \models \ell$ iff $\ell \in S$, for each fluent literal ℓ;

3. $S \models F \wedge G$ iff $S \models F$ and $S \models G$;

4. $S \models F \vee G$ iff $S \models F$ or $S \models G$;

5. $S \models F \supset G$ iff $S \not\models F$ or $S \models G$;

[25] To be precise, we always assume an additional program clause representing reflexivity, $X =_{\mathrm{AC1}} X$, which is employed to solve equality subgoals.

[26] AC1-unification is known to be decidable and finitary, i.e., two terms always admit a finite complete set of AC1-unifiers [Stickel, 1981]. For efficient unification algorithms see, e.g., [Bürckert *et al.*, 1988; Lincoln and Christian, 1990; Große *et al.*, 1992].

6. $S \models F \equiv G$ iff $S \models F$ and $S \models G$, or $S \not\models F$ and $S \not\models G$;

7. $S \models \exists X.\, F$ iff there exists some $o \in \mathcal{O}$ such that $S \models F\{X \mapsto o\}$;

8. $S \models \forall X.\, F$ iff $S \models F\{X \mapsto o\}$ for any $o \in \mathcal{O}$.

Here, $F\{X \mapsto o\}$ denotes the fluent formula resulting from replacing in F all occurrences of X by o. ■

When trying to formulate program clauses to decide validity of fluent formulas wrt states, we are facing the well-known problem of how to express universally quantified statements in bodies of program clauses. Ordinary, i.e. definite, clauses do not allow for encoding rules like "p if, for all X, $r(X)$." To overcome this lack of expressiveness, we follow [Lloyd, 1987] in employing the negation-as-failure principle [Clark, 1978] and re-formulate the above statement to "p if not q" in conjunction with "q if exists X such that *non-*$r(X)$." More precisely, to encode that $p(\tau_S)$ shall be true if formula $\forall X.\, F$ holds in state S, these two clauses are introduced:

$$\begin{array}{lcl} p(S) & \leftarrow & \neg q(S). \\ q(S) & \leftarrow & \mathit{entity}(X), \\ & & r(X, S). \end{array}$$

where $r(X, S)$ is assumed to be appropriately defined such that an instance $r(o, \tau_S)$ is derivable iff $F\{X \mapsto o\}$ does *not* hold in S. Generally, we first of all represent a given set of entities $\mathcal{O} = \{o_1, \ldots, o_n\}$ by these n facts:

$$\begin{array}{l} \mathit{entity}(o_1). \\ \quad \vdots \\ \mathit{entity}(o_n). \end{array} \tag{27}$$

Then the following transformation π (along with its dual $\overline{\pi}$) maps a fluent formula $F[\widetilde{X}]$ plus an atom $p(\widetilde{X})$ to a set of program clauses $\mathcal{C}$ which has the following property: An instance $\leftarrow p(\widetilde{o}, \tau_S)$ is refutable from $\mathcal{C}$ and (27) iff $F[\widetilde{o}]$ holds in S (resp. does not hold, in case the dual transformation is used, i.e., if $\mathcal{C} = \overline{\pi}(F, p(\widetilde{X}))$).[27]

$$\pi(\top, p(\widetilde{X})) := \{\ p(\widetilde{X}, S).\ \} \tag{28}$$

$$\pi(\bot, p(\widetilde{X})) := \{\ \} \tag{29}$$

$$\pi(\ell, p(\widetilde{X})) := \{\ p(\widetilde{X}, S) \leftarrow S =_{\mathrm{AC1}} \ell \circ V.\ \} \tag{30}$$

$$\begin{array}{rcl} \pi(F \wedge G, p(\widetilde{X})) & := & \{\ p(\widetilde{X}, S) \leftarrow q(\widetilde{X}, S), r(\widetilde{X}, S).\ \} \\ & & \cup\ \pi(F, q(\widetilde{X}))\ \cup\ \pi(G, r(\widetilde{X})) \end{array} \tag{31}$$

$$\pi(F \vee G, p(\widetilde{X})) := \pi(F, p(\widetilde{X}))\ \cup\ \pi(G, p(\widetilde{X})) \tag{32}$$

$$\begin{array}{rcl} \pi(\exists X.\, F, p(\widetilde{X})) & := & \{\ p(\widetilde{X}, S) \leftarrow \mathit{entity}(X), q(\widetilde{X}, X, S).\ \} \\ & & \cup\ \pi(F, q(\widetilde{X}, X)) \end{array} \tag{33}$$

$$\begin{array}{rcl} \pi(\forall X.\, F, p(\widetilde{X})) & := & \{\ p(\widetilde{X}, S) \leftarrow \neg q(\widetilde{X}, S). \\ & & \quad q(\widetilde{X}, S) \leftarrow \mathit{entity}(X), r'(\widetilde{X}, X, S).\ \} \\ & & \cup\ \overline{\pi}(F, r'(\widetilde{X}, X)) \end{array} \tag{34}$$

[27] For brevity, we omit the translations of connectives "$\supset$" and "$\equiv$", both of which can be expressed by the remaining ones.

The above transformation is straightforward and involves the dual $\overline{\pi}$, to be defined next, in case of universally quantified variables, (34), as discussed above.

$$
\begin{array}{rcll}
\overline{\pi}(\top, p'(\widetilde{X})) & := & \{\ \} & (35)\\
\overline{\pi}(\bot, p'(\widetilde{X})) & := & \{\ p'(\widetilde{X}, S).\ \} & (36)\\
\overline{\pi}(f, p'(\widetilde{X})) & := & \{\ p'(\widetilde{X}, S) \leftarrow S =_{\mathrm{AC1}} \overline{f} \circ V.\ \} & (37)\\
\overline{\pi}(\overline{f}, p'(\widetilde{X})) & := & \{\ p'(\widetilde{X}, S) \leftarrow S =_{\mathrm{AC1}} f \circ V.\ \} & (38)\\
\overline{\pi}(F \wedge G, p'(\widetilde{X})) & := & \overline{\pi}(F, p'(\widetilde{X})) \ \cup\ \overline{\pi}(G, p'(\widetilde{X})) & (39)\\
\overline{\pi}(F \vee G, p'(\widetilde{X})) & := & \{\ p'(\widetilde{X}, S) \leftarrow q'(\widetilde{X}, S), r'(\widetilde{X}, S).\ \} & (40)\\
 & & \cup\ \overline{\pi}(F, q'(\widetilde{X}))\ \cup\ \overline{\pi}(G, r'(\widetilde{X})) & \\
\overline{\pi}(\exists X.\, F, p'(\widetilde{X})) & := & \{\ p'(\widetilde{X}, S) \leftarrow \neg q'(\widetilde{X}, S). & (41)\\
 & & \quad q'(\widetilde{X}, S) \leftarrow \mathit{entity}(X), r(\widetilde{X}, X, S).\ \} & \\
 & & \cup\ \pi(F, r(\widetilde{X}, X)) & \\
\overline{\pi}(\forall X.\, F, p'(\widetilde{X})) & := & \{\ p'(\widetilde{X}, S) \leftarrow \mathit{entity}(X), q'(\widetilde{X}, X, S).\ \} & (42)\\
 & & \cup\ \overline{\pi}(F, q'(\widetilde{X}, X)) &
\end{array}
$$

As an example, consider domain constraint

$$(\ \forall X.\, \mathit{on}(X) \wedge \mathit{light}\ )\ \vee\ (\ \exists X.\, \overline{\mathit{on}(X)} \wedge \overline{\mathit{light}}\ ) \tag{43}$$

stating that the light bulb is on if and only if all switches are on. This formula should provably hold, say, in the state represented by

$$\mathit{on}(s_1) \circ \mathit{on}(s_2) \circ \mathit{light} \tag{44}$$

To verify this, we take the translation $\pi((43), p_1)$, augmented by the respective clauses (27) for the two entities s_1 and s_2, which results in this set of clauses:

$$
\begin{array}{lll}
p_1(S) \leftarrow q_1(S),\ r_1(S). & =:\ \pi(\forall X.\, \mathit{on}(X) \wedge \mathit{light}, p_1) & \\
 & & \\
q_1(S) \leftarrow \neg q_2(S). & =:\ \pi(\forall X.\, \mathit{on}(X), q_1) & \\
q_2(S) \leftarrow \mathit{entity}(X),\ r_2'(X, S). & & \\
 & & \\
r_2'(X, S) \leftarrow S =_{\mathrm{AC1}} \overline{\mathit{on}(X)} \circ V. & =:\ \overline{\pi}(\mathit{on}(X), r_2'(X)) & \\
 & & \\
r_1(S) \leftarrow S =_{\mathrm{AC1}} \mathit{light} \circ V. & =:\ \pi(\mathit{light}, r_1) & \\
 & & \\
p_1(S) \leftarrow q_3(S),\ r_3(S). & =:\ \pi(\exists X.\, \overline{\mathit{on}(X)} \wedge \overline{\mathit{light}}, p_1) & (45)\\
 & & \\
q_3(S) \leftarrow \mathit{entity}(X),\ q_4(X, S). & =:\ \pi(\exists X.\, \overline{\mathit{on}(X)}, q_3) & \\
 & & \\
q_4(X, S) \leftarrow S =_{\mathrm{AC1}} \overline{\mathit{on}(X)} \circ V. & =:\ \pi(\overline{\mathit{on}(X)}, q_4) & \\
 & & \\
r_3(S) \leftarrow S =_{\mathrm{AC1}} \overline{\mathit{light}} \circ V. & =:\ \pi(\overline{\mathit{light}}, r_3) & \\
 & & \\
\mathit{entity}(s_1). & & \\
\mathit{entity}(s_2). & &
\end{array}
$$

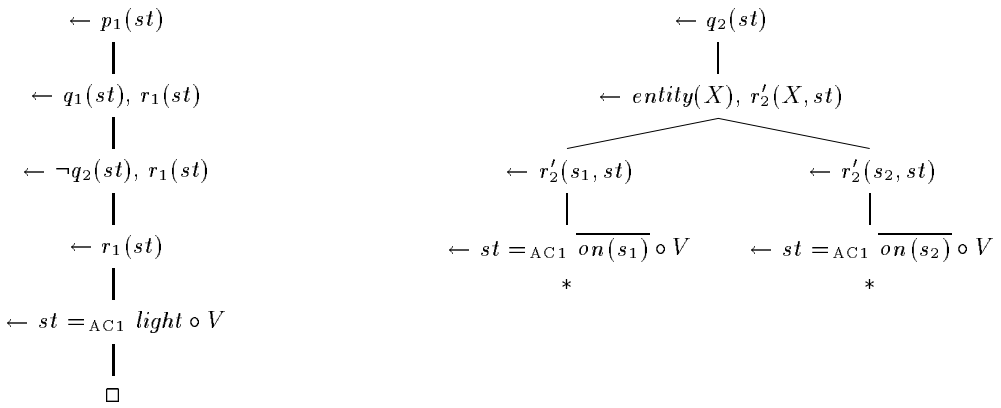

Figure 6: Let st abbreviate term $on(s_1) \circ on(s_2) \circ light$ then the leftmost derivation wrt the program listed in (45) verifies (43) be true in the state corresponding to st. In particular, success of subgoal $\leftarrow \neg q_2(st)$ (third step) is justified by the finitely failed SLDENF-tree on the right hand side. Its two branches fail because st is not AC1-unifiable with $\overline{on(s_1)} \circ V$ nor with $\overline{on(s_2)} \circ V$.

This equational logic program contains negative literals in clause bodies, which requires an extended resolution principle, namely *SLDENF-resolution* [Shepherdson, 1992; Thielscher, 1996], i.e., SLDE-resolution augmented by negation-as-failure. As usual, a negative literal as subgoal is solved by verifying that every derivation of the respective affirmative part fails, which in turn is shown by constructing a corresponding *finitely failed* derivation tree. Given the above clauses, Figure 6 depicts an SLDENF-derivation which shows that Formula (43) indeed holds in the state represented by (44).

In general, our encoding of fluent formulas and the associated notion of validity wrt states is correct in the sense of Definition 18:

Theorem 19 *Let S be a state, F a fluent formula with free variables $\widetilde{X}$, and $\widetilde{o}$ a sequence of entities. Then,*

1. $S \models F[\widetilde{o}]$ *iff* $\leftarrow p(\widetilde{o}, \tau_S)$ *has an SLDENF-refutation wrt* $\pi(F, p(\widetilde{X})), (27); (\mathrm{AC1})$.
2. $S \models F[\widetilde{o}]$ *iff* $\leftarrow p'(\widetilde{o}, \tau_S)$ *finitely fails wrt* $\overline{\pi}(F, p'(\widetilde{X})), (27); (\mathrm{AC1})$.

Proof: The proof is by simultaneous induction on the structure of F.

- In case $\top$,
  1. $S \models \top$ and $\leftarrow p(\widetilde{o}, \tau_S)$ is refutable using Clause (28).
  2. Analogously, following (35) there is no clause whose head unifies with $p'(\widetilde{o}, \tau_S)$, i.e., $\leftarrow p'(\widetilde{o}, \tau_S)$ finitely fails.
- In case $\bot$,
  1. $S \not\models \bot$ and, following (29), there is no clause whose head unifies with $p(\widetilde{o}, \tau_S)$.
  2. Analogously, $\leftarrow p'(\widetilde{o}, \tau_S)$ is refutable using Clause (36).
- In case ℓ,
  1. $S \models \ell[\widetilde{o}]$ iff $\ell[\widetilde{o}] \in S$. Correspondingly, $\leftarrow p(\widetilde{o}, \tau_S)$ can only be resolved using Clause (30) and only if τ_S and $\ell[\widetilde{o}] \circ V$ are AC1-unifiable, which is equivalent to $\ell[\widetilde{o}] \in S$.

2. $S \models f[\tilde{o}]$ (resp. $S \models \overline{f[\tilde{o}]}$) iff $\overline{f[\tilde{o}]} \notin S$ (resp. $f[\tilde{o}] \notin S$). Correspondingly, following (37) and (38), $\leftarrow p'(\tilde{o}, \tau_S)$ finitely fails iff τ_S and $\overline{f[\tilde{o}]} \circ V$ (resp. $f[\tilde{o}] \circ V$) are not AC1-unifiable.

- In case $F \wedge G$,
  1. $S \models F[\tilde{o}] \wedge G[\tilde{o}]$ iff $S \models F[\tilde{o}]$ and $S \models G[\tilde{o}]$. Correspondingly, following Clause (31), $\leftarrow p(\tilde{o}, \tau_S)$ has an SLDENF-refutation iff $\leftarrow q(\tilde{o}, \tau_S)$, $r(\tilde{o}, \tau_S)$ succeeds, which is equivalent to $S \models F[\tilde{o}]$ and $S \models G[\tilde{o}]$ according to the induction hypothesis.
  2. Analogously, following (39), $\leftarrow p'(\tilde{o}, \tau_S)$ finitely fails iff both $S \models F[\tilde{o}]$ and $S \models G[\tilde{o}]$, according to the induction hypothesis.
- In case $F \vee G$,
  1. $S \models F[\tilde{o}] \vee G[\tilde{o}]$ iff $S \models F[\tilde{o}]$ or $S \models G[\tilde{o}]$. Correspondingly, following (32), $\leftarrow p(\tilde{o}, \tau_S)$ has an SLDENF-refutation iff $S \models F[\tilde{o}]$ or $S \models G[\tilde{o}]$, according to the induction hypothesis.
  2. Analogously, $\leftarrow p'(\tilde{o}, \tau_S)$ finitely fails iff $\leftarrow q'(\tilde{o}, \tau_S)$, $r'(\tilde{o}, \tau_S)$ finitely fails, according to Clause (40), which is equivalent to $S \models F[\tilde{o}]$ or $S \models G[\tilde{o}]$ according to the induction hypothesis.
- In case $\exists X.\, F$,
  1. $S \models \exists X.\, F[\tilde{o}]$ iff $S \models F[\tilde{o}]\{X \mapsto o\}$ for some $o \in \mathcal{O}$. Correspondingly, following (33) and (27), $\leftarrow p(\tilde{o}, \tau_S)$ has an SLDENF-refutation iff there is some $o \in \mathcal{O}$ such that $\leftarrow q(\tilde{o}, o, \tau_S)$ is refutable, which is equivalent to $S \models F[\tilde{o}]\{X \mapsto o\}$ according to the induction hypothesis.
  2. Analogously, following (41) and (27), $\leftarrow p'(\tilde{o}, \tau_S)$ finitely fails iff $\leftarrow q'(\tilde{o}, \tau_S)$ succeeds iff there is some $o \in \mathcal{O}$ such that $\leftarrow r(\tilde{o}, o, \tau_S)$ succeeds, which is equivalent to $S \models F[\tilde{o}]\{X \mapsto o\}$ according to the induction hypothesis for π.
- In case $\forall X.\, F$,
  1. $S \models \forall X.\, F[\tilde{o}]$ iff $S \models F[\tilde{o}]\{X \mapsto o\}$ for any $o \in \mathcal{O}$. Correspondingly, following (34) and (27), $\leftarrow p(\tilde{o}, \tau_S)$ has an SLDENF-refutation iff $\leftarrow q(\tilde{o}, \tau_S)$ finitely fails, i.e., iff $\leftarrow r(\tilde{o}, o, \tau_S)$ fails for each $o \in \mathcal{O}$. The latter is equivalent to $S \models F[\tilde{o}]\{X \mapsto o\}$ according to the induction hypothesis for $\overline{\pi}$.
  2. Analogously, following (42) and (27), $\leftarrow p'(\tilde{o}, \tau_S)$ finitely fails iff, for each $o \in \mathcal{O}$, $\leftarrow q'(\tilde{o}, o, \tau_S)$ fails. The latter is equivalent to $S \models F[\tilde{o}]\{X \mapsto o\}$ according to the induction hypothesis.

■

6.2 Executing Causal Relationships

Having integrated the concept of fluent formulas into the basic program in order to be able to represent domain constraints, we now formalize the notion of causal relationships and their application to account for indirect effects of actions. Due to the extended notion of fluents employed in this part of the paper, we lift Definition 4 and 5 to exploit this gain of expressiveness on the analogy of supporting a more general concept of action description (c.f. Definition 16):

Definition 20 A *causal relationship* is an expression of the form e `causes` r `if` Φ where Φ is a fluent formula and e and r are (possibly negated) fluent expressions.

Let (S, E) be a pair consisting of a state S and a set of fluent literals E. Furthermore, let $\rho = e$ `causes` r `if` Φ be a causal relationship, and let $\widetilde{X}$ denote a complete sequence of variables occurring in e, r, or Φ. Then an instance $\rho[\widetilde{o}]$ is *applicable* to (S, E) iff $S \models \Phi[\widetilde{o}] \wedge e[\widetilde{o}] \wedge \overline{r[\widetilde{o}]}$ and $e[\widetilde{o}] \in E$. Its application yields the pair $((S \setminus \{\overline{r[\widetilde{o}]}\}) \cup \{r[\widetilde{o}]\}, E \cup \{r[\widetilde{o}]\})$.

Let $\mathcal{A}$ be a set of action descriptions, $\mathcal{D}$ a set of domain constraints, and $\mathcal{R}$ a set of causal relationships. Furthermore, let S be a state satisfying $\mathcal{D}$ and $a[\widetilde{o}]$ an instantiated action. If there exists an action description $\alpha[\widetilde{X}] = \langle C[\widetilde{X}], a[\widetilde{X}], E[\widetilde{X}] \rangle \in \mathcal{A}$ such that $\alpha[\widetilde{o}]$ is applicable in S then a state S' is *successor state* iff

1. $((S \setminus C[\widetilde{o}]) \cup E[\widetilde{o}], E[\widetilde{o}]) \stackrel{*}{\rightsquigarrow}_{\mathcal{R}} (S', E')$ for some E' and
2. S' satisfies $\mathcal{D}$.

■

Example 1 (continued) Based on the formalization of our Electric Circuit domain introduced in this section, we employ the following two causal relationships:

$$on(X) \texttt{ causes } \mathit{light} \texttt{ if } \forall Y.\, on(Y) \qquad \overline{on(X)} \texttt{ causes } \overline{\mathit{light}} \texttt{ if } \top \tag{46}$$

Then the instance $\{X \mapsto s_1\}$ of the former is applicable to $(\{on(s_1), on(s_2), \overline{\mathit{light}}\}, \{on(s_1)\})$ since both $\forall Y.\, on(Y)$ and also $on(s_1)$ and $\overline{\mathit{light}}$ are true in the state at hand. The application results in the pair $(\{on(s_1), on(s_2), \mathit{light}\}, \{on(s_1), \mathit{light}\})$; its first component now satisfies the underlying domain constraint, (43). ■

Similar to action descriptions, each given causal relationship is encoded by a program clause defining predicate $\mathit{causes}(\tau_S, \tau_E, \tau_{S'}, \tau_{E'})$, which is intended to be true if $(S, E) \rightsquigarrow_{\mathcal{R}} (S', E')$. Let $\mathcal{R}$ be a finite set of causal relationships, then each e `causes` r `if` $\Phi \in \mathcal{R}$ is represented by

$$\mathit{causes}(V \circ e \circ \overline{r}, W \circ e, V \circ e \circ r, W \circ e \circ r) \leftarrow p_\Phi(V \circ e \circ \overline{r}) \tag{47}$$

This involves the respective clauses obtained by constructing $\pi(\Phi, p_\Phi)$ to encode fluent formula Φ (including $(27)_{\mathcal{O}}$).

The successive application of causal relationships to state-effect pairs is represented by the predicate $\mathit{ramify}(\tau_S, \tau_E, \tau_{S'})$, which is intended to be true if $(S, E) \stackrel{*}{\rightsquigarrow}_{\mathcal{R}} (S', E')$ for some E' such that S' satisfies the underlying domain constraints. The latter notion is encoded by defining a unary predicate called *consistent*:

$$\begin{array}{lcl} \mathit{ramify}(S, E, S) & \leftarrow & \mathit{consistent}(S). \\ \mathit{ramify}(S, E, T) & \leftarrow & \mathit{causes}(S, E, S', E'), \\ & & \mathit{ramify}(S', E', T). \end{array} \tag{48}$$

Let $\{D_1, \ldots, D_n\}$ be the underlying set of domain constraints then

$$\begin{array}{lcl} \mathit{consistent}(S) & \leftarrow & p_1(S), \\ & & \vdots \\ & & p_n(S). \end{array} \tag{49}$$

plus all clauses obtained by constructing $\pi(D_1, p_1), \ldots, \pi(D_n, p_n)$.

Finally, the post-processing step that accommodates possible indirect effects has to be added to the core clause in the basic approach, (26), yielding

$$\begin{array}{rcl} result(S,A,S') & \leftarrow & action(C,A,E), \\ & & S =_{\mathrm{AC1}} C \circ V, \\ & & ramify(V \circ E, E, S'). \end{array} \tag{50}$$

Example 1 (continued) The two causal relationships in (46) are represented by these two clauses:

$$\begin{array}{l} causes(V \circ on(X) \circ \overline{light}, W \circ on(X), V \circ on(X) \circ light, W \circ on(X) \circ light) \\ \qquad\qquad \leftarrow q_1(V \circ on(X) \circ \overline{light}). \\ causes(V \circ \overline{on(X)} \circ light, W \circ \overline{on(X)}, V \circ \overline{on(X)} \circ \overline{light}, W \circ \overline{on(X)} \circ \overline{light}). \end{array} \tag{51}$$

where q_1 encodes formula $\forall Y.\, on(Y)$ as in (45). These clauses along with (50) and the definition of consistency based on domain constraint (43), viz

$$consistent(S) \leftarrow p_1(S). \tag{52}$$

are used in Figure 7, where an SLDENF-derivation is shown that verifies $\{on(s_1), on(s_2), light\}$ be successor of $\{\overline{on(s_1)}, on(s_2), \overline{light}\}$ wrt action $toggle(s_1)$. ■

As the main result of the second part of this paper we prove that our encoding by means of equational logic programs is correct. The proof consists of two steps, the first of which concerns the application of a single causal relationship:

Theorem 21 *Let $\mathcal{D}$ be a set of domain constraints and $\mathcal{R}$ a set of causal relationships. Furthermore, let S, S' be two states and E, E' two sets of fluent literals then $(S,E) \rightsquigarrow_{\mathcal{R}} (S',E')$ iff*

$$\leftarrow causes(\tau_S, \tau_E, \tau_{S'}, \tau_{E'}) \tag{53}$$

has an SLDENF-refutation wrt $(47)_{\mathcal{R}}$; (AC1).

Proof: We have $(S,E) \rightsquigarrow_{\mathcal{R}} (S',E')$ iff there exists some instance $\rho[\tilde{o}]$ of some causal relationship $\rho = e$ `causes` r `if` $\Phi \in \mathcal{R}$ such that

1. $S \models \Phi[\tilde{o}] \wedge e[\tilde{o}] \wedge \overline{r[\tilde{o}]}$
2. $e[\tilde{o}] \in E$
3. $S' = (S \setminus \{\overline{r[\tilde{o}]}\}) \cup \{r[\tilde{o}]\}$
4. $E' = E \cup \{r[\tilde{o}]\}$

Correspondingly, (53) has an SLDENF-refutation iff $(47)_{\mathcal{R}}$ contains a clause

$$causes(V \circ e \circ \overline{r}, W \circ e, V \circ e \circ r, W \circ e \circ r) \leftarrow p_\Phi(V \circ e \circ \overline{r}).$$

such that there exists an AC1-unifier σ for these four equations:

$$\tau_S =_{\mathrm{AC1}} V \circ e \circ \overline{r}, \quad \tau_E =_{\mathrm{AC1}} W \circ e, \quad \tau_{S'} =_{\mathrm{AC1}} V \circ e \circ r, \quad \tau_{E'} =_{\mathrm{AC1}} W \circ e \circ r$$

and $\leftarrow p_\Phi(V \circ e \circ \overline{r})\sigma$ is SLDENF-refutable. According to axioms (AC1) and Theorem 19, this is true iff there exist entities $\tilde{o}$ such that

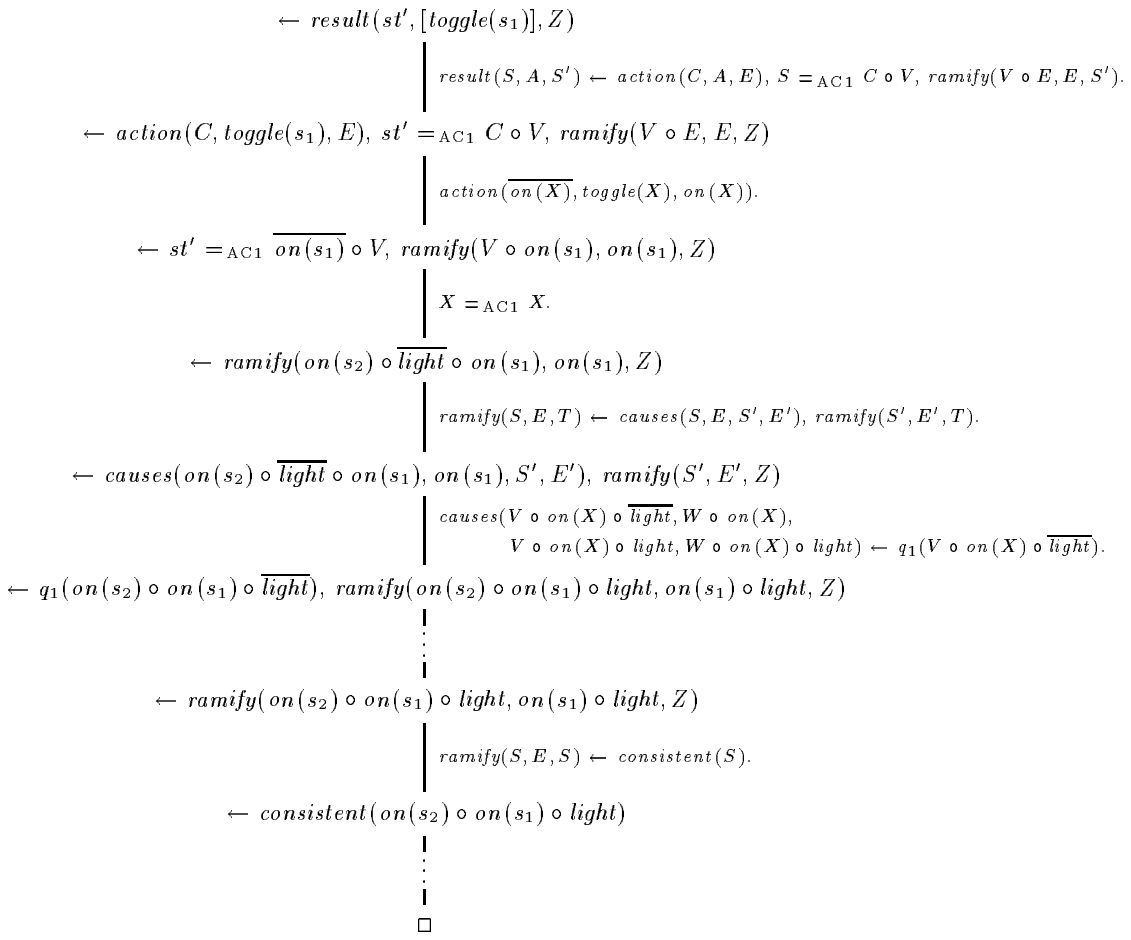

Figure 7: An SLDENF-derivation based on clauses $(25)_{(24)}, (45), (48), (50)$–$(52)$. Let st' abbreviate $\overline{on(s_1)} \circ on(s_2) \circ \overline{light}$ then the derivation shows that executing $toggle(s_1)$ in the state represented by st' admits $\{on(s_1), on(s_2), light\}$ as successor state. For sake of readability, the respective applied program clause is attached to each derivation step. The sub-derivation of $\leftarrow q_1(on(s_2) \circ on(s_1) \circ \overline{light})$ is omitted as it is similar to how $\leftarrow q_1(st)$ is solved in Figure 6, which also shows why $\leftarrow consistent(on(s_2) \circ on(s_1) \circ light)$ can be refuted via (52) and (45).

---

1. $e[\tilde{o}] \in S$ and $\overline{r[\tilde{o}]} \in S$;
2. $e[\tilde{o}] \in E$;
3. $S' = (S \setminus \{e[\tilde{o}], \overline{r[\tilde{o}]}\}) \cup \{e[\tilde{o}], r[\tilde{o}]\}$;
4. $E' = (E \setminus \{e[\tilde{o}]\}) \cup \{e[\tilde{o}], r[\tilde{o}]\}$; and
5. $S \models \Phi[\tilde{o}]$.

Altogether these conditions are equivalent to the ones enumerated above.

■

On the basis of this theorem, we can prove correctness of the entire program:

Theorem 22 *Let $\mathcal{D}$ be a set of domain constraints, $\mathcal{R}$ a set of causal relationships, and $\mathcal{A}$ a set of action descriptions. Furthermore, let S, S' be two states and $a\theta$ ground instance of an*

action then S' *is successor state of* S *wrt* $a\theta$ *iff*

$$\leftarrow \mathit{result}(\tau_S, a\theta, \tau_{S'}) \tag{54}$$

has an SLDENF-refutation wrt $(25)_{\mathcal{A}}, (47)_{\mathcal{R}}, (48), (49)_{\mathcal{D}}, (50); (\mathrm{AC1})$.

Proof: From Theorem 17 and Clause (50) we conclude that there exists some action description $\langle C, a, E\rangle \in \mathcal{A}$ such that $C\theta \subseteq S$ iff Query (54) can be resolved to

$$\leftarrow \mathit{ramify}(\tau_{(S \setminus C\theta) \cup E\theta}, \tau_{E\theta}, \tau_{S'})$$

From Theorem 21 and the clauses depicted in (48) it follows this query is refutable iff there exists some E' such that $(S, E) \overset{*}{\rightsquigarrow}_{\mathcal{R}} (S', E')$ and

$$\leftarrow \mathit{consistent}(\tau_{S'})$$

has an SLDENF-refutation. According to Theorem 19 and Clause $(49)_{\mathcal{D}}$, this is true iff S' satisfies the domain constraints in $\mathcal{D}$. To summarize, (54) has an SLDENF-refutation iff S' is successor of S wrt a according to Definition 20.

■

7 Discussion

We have presented a method to accommodate indirect effects of actions which involves the notion of causality to distinguish intuitively conceivable from unmotivated changes. To this end, we have developed the concept of *causal relationships* connecting two effects with the intended meaning that, under specific circumstances, the occurrence of the former might cause the occurrence of the latter as indirect effect. Causal relationships are serially applied to the intermediate state resulting from computing the direct effects of an action until a state obtains that satisfies all underlying domain constraints. Moreover, we have argued that causal relationships can be generated automatically given additional domain-specific knowledge—called *influence information*—of how fluents may generally affect each other.

We have illustrated the expressiveness of our approach regarding the problem of implicit qualifications vs. indirect effects (Subsection 3.3), fluents which strive against being categorized (Subsection 5.1), domains involving non-minimal yet still intuitive changes (Subsection 5.2), and regarding domain constraints that require a sophisticated distinction between qualification and ramification (Subsection 5.3). These results form the basis for comparing the method presented in this paper with existing approaches to the ramification problem.

The necessity of additional information to prevent changes that are suggested syntactically by the mere domain constraints but contradict the intuition was first observed in [Ginsberg and Smith, 1988b] in the context of the *possible worlds approach* [Ginsberg and Smith, 1988a]. There, indirect effects of actions are implicitly obtained by searching for successor worlds (i.e., state descriptions) staying as close as possible to the original world and satisfying both the direct effects of the action under consideration and all domain constraints. While the authors argued that this might yield unintended changes (such as a switch magically jumping its position in the circuit depicted in Figure 1), no solution was offered.

In [Lifschitz, 1990], the first and most elementary categorization-based solution to this problem was formally developed by distinguishing between so-called *frame* and *non-frame* fluents.[28] Only the former are subject to the persistence assumption, and their respective values completely determine the states of the latter. Similar ideas have been used in e.g. [del Val and Shoham, 1993] and (in the second part of) [Brewka and Hertzberg, 1993]. More sophisticated categorization methods do not simply restrict persistence to one category by allowing arbitrary changes in the other—rather they exploit different categories to define a partial preference ordering amongst all possible changes (as in our Definition 14), e.g. [Zhang and Foo, 1993; Kartha and Lifschitz, 1994; Sandewall, 1995a]. In [Sandewall, 1995b], a systematic framework based on this concept is introduced with the aim of assessing the range of applicability of different approaches that follow the same principles. However, we have already argued in Subsection 5.1 that even if it is possible to assign an appropriate category to each fluent in a particular domain if only the categorization is suitably fine-grained, the more complex a domain is the more difficult this task becomes as it requires a deep analysis of possible interactions. Besides, despite being the only suitable one, a particular categorization may appear very unnatural even in simple domains, as we have illustrated in the context of Example 3.

Recently, some approaches have been established that take into account specific notions of causality, as does our method, to tackle the problem of unintended changes. The monotonic first-order formalism developed in [Elkan, 1992] supports specifications of indirect effects by means of complete descriptions of how the truth-value of a particular fluent might be caused to change, e.g. (c.f. Example 1)

$$\begin{aligned} \forall A, S\ [\ \mathit{causes}(A, S, \mathit{light}) \ \equiv\ & (\ \mathit{causes}(A, S, sw_1) \wedge \mathit{holds}(sw_2, S)\ ) \\ & \vee\ (\ \mathit{causes}(A, S, sw_2) \wedge \mathit{holds}(sw_1, S)\ )\ ] \\ \forall A, S\ [\ \mathit{cancels}(A, S, \mathit{light}) \ \equiv\ & \mathit{cancels}(A, S, sw_1) \vee \mathit{cancels}(A, S, sw_2)\ ] \end{aligned} \tag{55}$$

where $\mathit{causes}(a, s, f)$ should be read as "executing action a in state s causes fluent f to become true," $\mathit{cancels}(a, s, f)$ as "executing action a in state s causes fluent f to become false," and $\mathit{holds}(f, s)$ as "fluent f is true in state s." Given the specification of how sw_1 can possibly become true (resp. false), viz

$$\begin{aligned} \forall A, S\ [\ \mathit{causes}(A, S, sw_1) \ &\equiv\ A = \mathit{toggle}_1 \wedge \neg \mathit{holds}(sw_1, s)\ ] \\ \forall A, S\ [\ \mathit{cancels}(A, S, sw_1) \ &\equiv\ A = \mathit{toggle}_1 \wedge \mathit{holds}(sw_1, s)\ ] \end{aligned}$$

plus this axiom of persistence:

$$\forall A, S, F\ [\ \mathit{holds}(F, \mathit{do}(A, S)) \ \equiv\ \mathit{causes}(A, S, F) \vee (\ \mathit{holds}(F, S) \wedge \neg \mathit{cancels}(A, S, F)\ )\ ]$$

one obtains $\neg \mathit{holds}(sw_1, s_0) \wedge \mathit{holds}(sw_2, s_0) \wedge \neg \mathit{holds}(\mathit{light}, s_0)$ implies $\mathit{causes}(\mathit{toggle}_1, s_0, \mathit{light})$, say, and, hence, $\mathit{holds}(\mathit{light}, \mathit{do}(\mathit{toggle}_1, s_0))$. No effort has to be made to suppress an unwanted change of sw_2 since no causal relation similar to (55) exists that may support this. On the other hand, the use of if-and-only-if descriptions of causal dependencies, as in (55), is severely restricted to domains where these dependencies are hierarchical. Otherwise, i.e., if fluents depend mutually, unmotivated changes cannot be precluded. To see why, consider the specification

$$\forall A, S\ [\ \mathit{causes}(A, S, f_1) \ \equiv\ \mathit{causes}(A, S, f_2)\ ] \tag{56}$$

[28] Earlier, the author of [Winslett, 1988] raised the idea of introducing some notion of preference as regards changes of specific fluents to changes of other fluents; yet, her discussion was only informal and took place in the context of an example.

If a is an action whose execution in s does not influence f_1 nor f_2, Formula (56) in conjunction with the above axiom of persistence is too weak to conclude both f_1 and f_2 keep their truth-values since $\neg causes(a, s, f_1)$ (resp. $\neg causes(a, s, f_2)$) is not entailed. The two mechanically connected switches discussed in Subsection 3.1, for instance, constitute a simple example which falls into this category. A second limitation compared to our approach stems from the fact that the formula in the right hand side of a definition like (55) either refers to the original state (in case of $holds(f, S)$) or to the finally resulting successor state (in case of $causes(A, S, f)$ or $cancels(A, S, f)$, respectively, in conjunction with the persistence axiom). Consequently, no intermediately occurring fact can possibly trigger an indirect effect; hence, this formalization does not allow for deriving the successor state where a flash of the light bulb is recorded by the light detector in our Example 4.

In [Brewka and Hertzberg, 1993] and [McCain and Turner, 1995], the notion of causality is introduced by defining so-called causal rules (c.f. Definition 9), which are formally treated as directed deduction rules in order to avoid using them in a non-causal way (e.g., $sw_1 \wedge sw_2 \Rightarrow light$ has a different meaning than $sw_1 \wedge \overline{light} \Rightarrow \overline{sw_2}$). Aside from a far more simple formalization employed in [McCain and Turner, 1995] compared to [Brewka and Hertzberg, 1993], the latter does not allow for concluding implicit qualifications from domain constraints rather than ramifications (c.f. Subsection 3.3). The reason is that [Brewka and Hertzberg, 1993] always strive for a successor state no matter how many changes are necessary to this end, while [McCain and Turner, 1995] additionally require all changes be explicitly grounded on some causal rule. On the other hand, the two approaches appear closely related; e.g., we strongly presume their equivalence in case of deterministic actions and domain constraints not giving rise to qualifications.[29] In particular, both methods are grounded on the policy of minimal change, which amounts to rejecting any potential successor state whose distance to the original state is strictly larger than the distance of another proper successor state. As argued in Subsection 5.2, however, this paradigm might not always capture the intuition in so far as successor states may exist which have non-minimal distance but are equally plausible. A second difference between these two approaches on the one hand and our concept of causal relationships on the other hand has been elaborated in Subsection 5.3, where a lack of expressiveness of causal rules regarding sophisticated distinctions between qualifications and ramifications triggered by the same domain constraint has been illustrated. Finally, we should also stress both [Brewka and Hertzberg, 1993; McCain and Turner, 1995] assume that causal rules be given as part of a domain specification, which requires more design effort than necessary—as can be seen by our suggestion to generate causal relationships automatically by employing more general information on potential influences.

Similar remarks apply to a recently developed integration of causality into the situation calculus-based framework [Lin and Reiter, 1994], also with the aim of handling indirect effects [Lin, 1995]. There, first-order formulas resembling causal relationships are used to define dependencies between effects and their indirect consequences. These formulas are of the form

$$\forall S\ [\ \Phi(S) \wedge caused(f_1, v_1, S) \wedge \ldots \wedge caused(f_n, v_n, S)\ \supset\ caused(f, v, s)\ ] \qquad (57)$$

where $caused(f, v, s)$ should be read as "fluent f is caused to take truth-value v in state s"

[29] Moreover, a third approach, [Geffner, 1990; Geffner, 1992], which is based on a nonmonotonic theory of "conditional entailment," appears similar to [Brewka and Hertzberg, 1993; McCain and Turner, 1995] in using expressions which resemble causal rules. A thorough and formal comparison between these three frameworks, however, has not yet been accomplished.

and $\Phi(s)$ describes properties of state s, e.g. (c.f. Example 1):

$$\forall S\ [\ holds(sw_2, S) \wedge\ caused(sw_1, true, S)\ \supset\ caused(light, true, S)\ ]$$

along with the action definition[30]

$$\forall S\ [\ \neg holds(sw_1, S)\ \supset\ caused(sw_1, true, do(toggle_1, S))\ ] \tag{58}$$

The general axiom of persistence employed in this context is

$$\forall A, S, F\ [\ \neg\exists V.\, caused(F, V, do(A, S))\ \supset\ (\ holds(F, do(A, S)) \equiv\ holds(F, S)\ )\ ]$$

This axiom is of course useless unless the extension of predicate *caused* is minimized given the direct effects of actions (like in (58)) and the laws of causality (each of which is of the form (57)). This is formally achieved by applying circumscription [McCarthy, 1980]. Hence, aside from also devoting the effort of constructing the various causal relations to the designer, this method is grounded on the paradigm of minimal change as well. In fact, this work too appears closely related to [Brewka and Hertzberg, 1993; McCain and Turner, 1995]. On the other hand, since predicate *caused* might occur in the left hand side of (57), this type of formula is expressive enough, in contrast to causal rules, to allow for modeling our Example 5, namely, by employing $\forall S\,[\,holds(\textit{at-trap}, S) \wedge caused(\textit{trapdoor-open}, true, S)\ \supset\ caused(alive, false, S)\,]$ but not $\forall S\,[\,holds(\textit{trapdoor-open}, S) \wedge caused(\textit{at-trap}, true, S)\ \supset\ caused(alive, false, S)\,]$.

Generally, in order to account for reasonable yet non-minimal changes, more sophisticated means than minimization have to be developed for the approaches discussed above. This requires an elaborated way to carefully distinguish between conceivable changes triggered by actually occurred (direct or indirect) effects and unfounded changes. The approach developed in [Łukaszewicz and Madalińska-Bugaj, 1995], which uses Dijkstra's semantics of programming languages to reason about actions, fails to handle this challenge appropriately: There, the ramification problem is tackled by allowing actions to (temporarily) release fluents from being subject to the assumption of persistence; but in case the domain constraints do not completely determine the new values of all in this way released fluents, unexpected effects may be produced (e.g., a turkey magically starts walking if being shot at with an unloaded gun).[31] Causal relationships account for this problem since they are only applicable if the respective triggering effect either is amongst the direct effects of the action under consideration or has previously been generated as indirect effect.

An approach which is considerably different from all methods discussed so far yet still related, is based on networks representing probabilistic causal theories [Pearl, 1988b]. Such networks describe, in the first place, static dependencies between its components. As argued in [Pearl, 1993; Pearl, 1994], it is however feasible to re-set the truth-value of one or more nodes and, then, to adjust the values of all depending nodes according to standard (Bayesian) rules of probability—which can be interpreted as generating indirect effects. If probability values are restricted to the 0/1-case then such a network of fluents resembles our concept of influence information. For instance, Figure 8(a) depicts a network suitable for Example 4. Although the relation between this approach and the other methods discussed in this section is far from being formally established today, let us point out some restrictions of causal networks compared to our causal relationships. First, recall Figure 8(a). Since the resulting value of a node, after having

[30] For sake of simplicity we neglect the concept of action preconditions.

[31] This was observed by V. Lifschitz during the presentation of [Łukaszewicz and Madalińska-Bugaj, 1995].

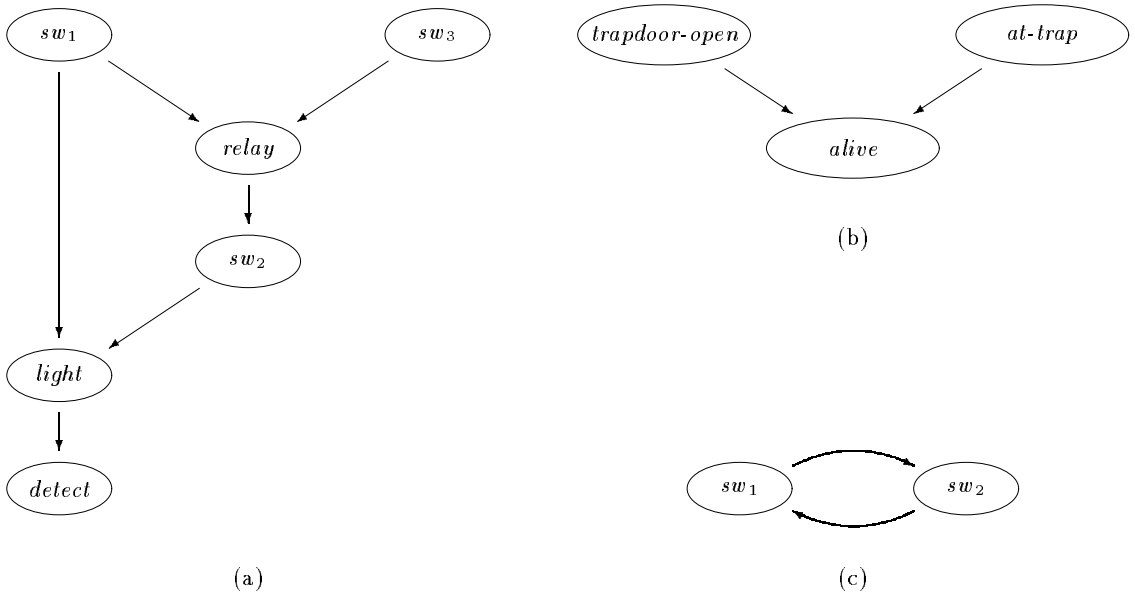

Figure 8: Causal networks representing the structural dependencies between fluents regarding (a) Example 4, (b) Example 5, and (c) domain constraint $\overline{sw_1} \vee \overline{sw_2}$ (inducing a cycle), respectively.

fixed the direct effects, must not be computed until all new values of its predecessors have been determined, the proposition *detect* necessarily remains false regarding Example 4 since *light* stays false, i.e., the non-minimal successor state where a light flash has been detected cannot be obtained. Second, consider Example 5. Since a change of *trapdoor-open* might cause a change of *alive* depending on *at-trap* , the adequate network is the one depicted in Figure 8(b). This, however, does not allow to distinguish between the two situations where either *trapdoor-open* becomes true with *at-trap* being true, or it happens to be the other way round. Hence, the sophisticated distinction whose necessity has been claimed in Subsection 5.3 is not supported by causal networks (similar to causal rules). Finally, networks representing causal theories are based on acyclic graphs, which means that simple examples like the mechanically connected switches (c.f. Subsection 3.1; relationships (3) and (4)), i.e., domain constraint $\overline{sw_1} \vee \overline{sw_2}$ in conjunction with influence information $\mathcal{I} = \{(sw_1, sw_2), (sw_2, sw_1)\}$, cannot be represented (c.f. Figure 8(c)). Aside from these rather specific observations, we would consider it most interesting to have a formal result regarding the range of applicability of approaches based on probabilistic networks. The calculus presented in [Pearl, 1994], for instance, considers only actions without preconditions, and it merely refers to the *temporal projection* problem, which names the task to predict effects of actions. This raises the question whether and how this approach can also be applied in other modes of reasoning like planning or *postdiction* (also called *chronicle completion* in [Sandewall, 1994]); the latter of which deals with finding explanations for observations made during the execution of action sequences.

This discussion leads us to the question of how to exploit the insights gained in this article. Our formalization in the first part of this paper has been embedded in a high-level, abstract description language and semantics for action scenarios, where we have concentrated on aspects of ramifications only and, to this end, employed a most simple form of action specifications

as regards direct effects. Three recent, similarly high-level action semantics focus on sophisticated ways to formalize this aspect, namely, the *Action Description Language* $\mathcal{A}$ [Gelfond and Lifschitz, 1993], the *Ego-World-Semantics* [Sandewall, 1994], and the framework presented in [Thielscher, 1995]. These approaches are considered prime candidates for being enhanced by the concept of causal relationships. In case of $\mathcal{A}$, this should be based on the dialect called $\mathcal{A}_{ND}$ [Thielscher, 1994b], which includes the notion of non-determinism, here needed if more than a single successor state exists. The resulting extension of the Action Description Language would constitute an alternate to the variant presented in [Kartha and Lifschitz, 1994], which handles ramifications on the basis of categorization and minimization, as does the extension of the Ego-World-Semantics presented in [Sandewall, 1995b].

The main purpose of these three formal frameworks is to provide a uniform semantics for calculi designed to reason about actions and change. Given our formal proposal to handle indirect effects, existing approaches can be extended in such a way that their solution to the ramification problem is provably correct wrt our semantics. As an exemplary formalism, in the second part of this paper we have adapted a method which is based on reification of entire state descriptions and uses equational logic programming [Hölldobler and Schneeberger, 1990; Hölldobler and Thielscher, 1995]. Our Theorem 22 demonstrates that the extension of this approach developed here is sound and complete wrt the formal semantics described in the first part. While the work reported in this article has been concentrated solely on the ramification problem, the particular approach—aside from being closely related, in its basic form, to the *Linear Connection Method* [Bibel, 1986] and reasoning about actions based on *Linear Logic* [Girard, 1987; Masseron *et al.*, 1993]—has shown a wide range of applicability, e.g. regarding postdiction problems and non-deterministic actions [Thielscher, 1994b], reasoning about counterfactual action sequences [Thielscher, 1994a], or concurrent actions in conjunction with (locally) inconsistent specifications [Bornscheuer and Thielscher, 1994]. Thus, a main goal of future research consists in combining all these results, each of which focuses on a single ontological aspect, into a uniform and expressive calculus.

Acknowledgments. The author wants to thank Wolfgang Bibel, Jürgen Giesl, Christoph Herrmann, Jana Köhler and Stuart Russell for helpful comments and suggestions.

References

[Baker, 1991] Andrew B. Baker. Nonmonotonic reasoning in the framework of situation calculus. *Artificial Intelligence Journal*, 49:5–23, 1991.

[Bibel, 1986] Wolfgang Bibel. A deductive solution for plan generation. *New Generation Computing*, 4:115–132, 1986.

[Bornscheuer and Thielscher, 1994] Sven-Erik Bornscheuer and Michael Thielscher. Representing concurrent actions and solving conflicts. In B. Nebel and L. Dreschler-Fischer, editors, *Proceedings of the German Annual Conference on Artificial Intelligence (KI)*, volume 861 of *LNAI*, pages 16–27, Saarbrücken, Germany, September 1994. Springer.

[Brewka and Hertzberg, 1993] Gerhard Brewka and Joachim Hertzberg. How to do things with worlds: on formalizing actions and plans. *Journal of Logic and Computation*, 3(5):517–532, 1993.

[Bürckert *et al.*, 1988] Hans-Jürgen Bürckert, Alexander Herold, Deepak Kapur, Jörg H. Siekmann, Mark E. Stickel, M. Tepp, and H. Zhang. Opening the AC-unification race. *Journal of Automated Reasoning*, 4:465–474, 1988.

[Clark, 1978] Keith L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Workshop Logic and Data Bases*, pages 293–322. Plenum Press, 1978.

[del Val and Shoham, 1993] Alvaro del Val and Yoav Shoham. Deriving properties of belief update from theories of action (II). In R. Bajcsy, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 732–737, Chambéry, France, August 1993. Morgan Kaufmann.

[Elkan, 1992] Charles Elkan. Reasoning about action in first-order logic. In *Proceedings of the Conference of the Canadian Society for Computational Studies of Intelligence*, Vancouver, Canada, May 1992.

[Fikes and Nilsson, 1971] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence Journal*, 2:189–208, 1971.

[Finger, 1987] Joseph J. Finger. *Exploiting Constraints in Design Synthesis.* PhD thesis, Stanford University, CA, 1987.

[Gallier and Raatz, 1989] Jean H. Gallier and Stan Raatz. Extending SLD-resolution to equational horn clauses using E-unification. *Journal of Logic Programming*, 6:3–44, 1989.

[Geffner, 1990] Hector Geffner. Causal theories for nonmonotonic reasoning. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 524–530, Boston, MA, 1990.

[Geffner, 1992] Hector Geffner. *Default Reasoning: Causal and Conditional Theories.* MIT Press, 1992.

[Gelfond and Lifschitz, 1993] Michael Gelfond and Vladimir Lifschitz. Representing Action and Change by Logic Programs. *Journal of Logic Programming*, 17:301–321, 1993.

[Ginsberg and Smith, 1988a] Matthew L. Ginsberg and David E. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence Journal*, 35:165–195, 1988.

[Ginsberg and Smith, 1988b] Matthew L. Ginsberg and David E. Smith. Reasoning about action II: The qualification problem. *Artificial Intelligence Journal*, 35:311–342, 1988.

[Girard, 1987] Jean-Yves Girard. Linear Logic. *Journal of Theoretical Computer Science*, 50(1):1–102, 1987.

[Green, 1969] Cordell Green. Application of theorem proving to problem solving. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 219–239, Los Altos, CA, 1969. Morgan Kaufmann.

[Große *et al.*, 1992] Gerd Große, Steffen Hölldobler, Josef Schneeberger, Ute Sigmund, and Michael Thielscher. Equational logic programming, actions, and change. In K. Apt, editor, *Proceedings of the International Joint Conference and Symposium on Logic Programming (IJCSLP)*, pages 177–191, Washington, 1992. MIT Press.

[Hanks and McDermott, 1987] Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence Journal*, 33(3):379–412, 1987.

[Hölldobler and Schneeberger, 1990] Steffen Hölldobler and Josef Schneeberger. A new deductive approach to planning. *New Generation Computing*, 8:225–244, 1990.

[Hölldobler and Thielscher, 1995] Steffen Hölldobler and Michael Thielscher. Computing change and specificity with equational logic programs. *Annals of Mathematics and Artificial Intelligence*, 14(1):99–133, 1995.

[Hölldobler, 1989] Steffen Hölldobler. *Foundations of Equational Logic Programming*, volume 353 of *LNAI*. Springer, 1989.

[Jaffar *et al.*, 1984] Joxan Jaffar, Jean-Louis Lassez, and Michael J. Maher. A theory of complete logic programs with equality. *Journal of Logic Programming*, 1(3):211–223, 1984.

[Kartha and Lifschitz, 1994] G. Neelakantan Kartha and Vladimir Lifschitz. Actions with indirect effects. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 341–350, Bonn, Germany, May 1994. Morgan Kaufmann.

[Lifschitz, 1986] Vladimir Lifschitz. On the semantics of STRIPS. In M. P. Georgeff and A. L. Lansky, editors, *Proceedings of the Workshop on Reasoning about Actions & Plans*. Morgan Kaufmann, 1986.

[Lifschitz, 1990] Vladimir Lifschitz. Frames in the space of situations. *Artificial Intelligence Journal*, 46:365–376, 1990.

[Lin and Reiter, 1994] Fangzhen Lin and Raymond Reiter. State constraints revisited. *Journal of Logic and Computation*, 4(5):655–678, 1994.

[Lin, 1995] Fangzhen Lin. Embracing causality in specifying the indirect effects of actions. In C. S. Mellish, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1985–1991, Montreal, Canada, August 1995. Morgan Kaufmann.

[Lincoln and Christian, 1990] Patrick D. Lincoln and J. Christian. Adventures in associative-commutative unification. In C. Kirchner, editor, *Unification*, pages 393–416. Academic Press Limited, 1990.

[Lloyd, 1987] John W. Lloyd. *Foundations of Logic Programming*. Series Symbolic Computation. Springer, second, extended edition, 1987.

[Łukaszewicz and Madalińska-Bugaj, 1995] Witold Łukaszewicz and Ewa Madalińska-Bugaj. Reasoning about action and change using Dijkstra's semantics for programming languages: Preliminary report. In C. S. Mellish, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1950–1955, Montreal, Canada, August 1995. Morgan Kaufmann.

[Masseron *et al.*, 1993] M. Masseron, Christophe Tollu, and Jacqueline Vauzielles. Generating plans in linear logic I. Actions as proofs. *Journal of Theoretical Computer Science*, 113:349–370, 1993.

[McCain and Turner, 1995] Norman McCain and Hudson Turner. A causal theory of ramifications and qalifications. In C. S. Mellish, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1978–1984, Montreal, Canada, August 1995. Morgan Kaufmann.

[McCarthy and Hayes, 1969] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[McCarthy, 1959] John McCarthy. Programs with Common Sense. In *Proceedings of the Teddington Conference on the Mechanization of Thought Processes*, London, 1959. (Reprinted in: J. McCarthy, *Formalizing Common Sense*, Ablex, Norwood, New Jersey, 1990).

[McCarthy, 1980] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence Journal*, 13:27–39, 1980.

[Pearl, 1988a] Judea Pearl. Embracing causality in default reasoning. *Artificial Intelligence Journal*, 35(2):259–271, 1988.

[Pearl, 1988b] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[Pearl, 1993] Judea Pearl. Graphical models, causality, and intervention. *Statistical Science*, 8(3):266–273, 1993.

[Pearl, 1994] Judea Pearl. A probabilistic calculus of actions. In R. Lopez de Mantaras and D. Poole, editors, *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 454–462, San Mateo, CA, 1994. Morgan Kaufmann.

[Plotkin, 1972] Gordon D. Plotkin. Building in equational theories. *Machine Intelligence*, 7:73–90, 1972.

[Sandewall, 1994] Erik Sandewall. *Features and Fluents*. Oxford University Press, 1994.

[Sandewall, 1995a] Erik Sandewall. Reasoning about actions and change with ramification. In *Computer Science Today*, volume 1000 of *LNCS*. Springer, 1995.

[Sandewall, 1995b] Erik Sandewall. Systematic comparison of approaches to ramification using restricted minimzation of change. Technical Report LiTH-IDA-R-95-15, Department of Computer Science, Linköping University, Sweden, 1995.

[Shepherdson, 1992] John C. Shepherdson. SLDNF-resolution with equality. *Journal of Automated Reasoning*, 8:297–306, 1992.

[Stickel, 1981] Mark E. Stickel. A unification algorithm for associative commutative functions. *Journal of the ACM*, 28(3):207–274, 1981.

[Thielscher, 1994a] Michael Thielscher. An analysis of systematic approaches to reasoning about actions and change. In P. Jorrand and V. Sgurev, editors, *International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA)*, pages 195–204, Sofia, Bulgaria, September 1994. World Scientific.

[Thielscher, 1994b] Michael Thielscher. Representing actions in equational logic programming. In P. Van Hentenryck, editor, *Proceedings of the International Conference on Logic Programming (ICLP)*, pages 207–224, Santa Margherita Ligure, Italy, June 1994. MIT Press.

[Thielscher, 1995] Michael Thielscher. The logic of dynamic systems. In C. S. Mellish, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1956–1962, Montreal, Canada, August 1995. Morgan Kaufmann.

[Thielscher, 1996] Michael Thielscher. On the completeness of SLDENF-resolution. *Journal of Automated Reasoning*, 1996. (To appear Fall '96).

[Winslett, 1988] Marianne Winslett. Reasoning about action using a possible models approach. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 89–93, Saint Paul, MN, August 1988.

[Zhang and Foo, 1993] Yan Zhang and Norman Y. Foo. Reasoning about persistence: A theory of actions. In R. Bajcsy, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 718–723, Chambéry, France, August 1993. Morgan Kaufmann.