Linköping Electronic Articles in
Computer and Information Science
Vol. 3(1998): nr 1

Delayed Effects of Actions = Direct Effects + Causal Rules

Patrick Doherty
Department of Computer
and Information Science
Linköping University
S-58183 Linköping, Sweden
patdo@ida.liu.se

Joakim Gustafsson
Department of Computer
and Information Science
Linköping University
S-58183 Linköping, Sweden
joagu@ida.liu.se

Linköping University Electronic Press
Linköping, Sweden

http://www.ep.liu.se/ea/cis/1998/001/

Published on January 26, 1998 by
Linköping University Electronic Press
581 83 Linköping, Sweden

Linköping Electronic Articles in
Computer and Information Science
ISSN 1401-9841
Series editor: Erik Sandewall

Typeset by the author using LaTeX
Formatted using étendu *style*

Recommended citation:
<Author>. <Title>. Linköping Electronic Articles in Computer and Information Science, Vol. 3(1998): nr 1.
http://www.ep.liu.se/ea/cis/1998/001/. *January 26, 1998.*

This URL will also contain a link to the author's home page.

Abstract

We propose an approach to modeling delayed effects of actions which is based on the use of causal constraints and their interaction with the direct effects of actions. The approach extends previous work with a causal approach used to deal with the ramification problem. We show the similarity between solutions to the modeling of indirect effects and delayed effects of actions by example. The base logic PMON$^+$ is a temporal logic for reasoning about action and change and uses circumscription. It is shown that the extension for delayed effects of actions retains the first-order reducibility property shown previously for successfully dealing with the frame and ramification problems for a large class of action scenarios. We also consider the "causal qualification" problem, "natural death" of fluents and causal lag, each of which is closely related to the use of delayed effects.

This report was written in November 1996, submitted to IJCAI'97 on January 21, 1997 and rejected. It is an exact reproduction of the submitted paper, but reformatted. A revised version has been resubmitted to ECAI'98 1998 and may be found in this series via `http://www.ep.liu.se/ea/cis/1997/002/`.

1 Introduction

The use of causal rules in theories of action and change is proving to be quite versatile in successfully dealing with a number of difficult representation issues in the area ([6],[9],[5],[8], [12]). In this paper, we propose an approach to modeling delayed effects of actions that is based on the use of both standard action law descriptions that describe direct effects of actions together with casual rules and the interaction between the two. The basic idea is that the direct effects of an action, under certain conditions, can "trigger" the execution of a causal constraint, where change in certain fluent values in the postcondition of an action rule satisfy the precondition of a causal constraint. The postcondition of the causal constraint will provide the delayed effect.

For instance, Lifschitz[7] uses the following example: "30 seconds after you press the button at the crosswalk, the pedestrian light turns green". Rather than represent the delayed effect in the postcondition of a *Press* action as Lifschitz attempts to do, we would provide an action law,

acs1 $[t_1, t_2]\ Press(button) \leadsto$
$[t_1, t_2]\ pressed(button) := T,$

and a causal constraint,

cc1 $\forall t.[t]\ pressed(button) \gg$
$[t + 30]\ color(light) \hat{=} green.$

The action law $acs1$, states that if there is an occurrence of the $Press$ action during the time interval from t_1 to t_2, the direct effect of the action changes the value of the $pressed(button)$ fluent to true. The causal constraint $cc1$, states that if the $Pressed(button)$ fluent changes value from false to true from $t-1$ to t then the color of the pedestrian light $color(light)$ will change value to green 30 seconds later.

With this approach, one can even represent the "natural death" of a fluent value, where the effects of an action or causal constraint are observed only during a limited period of time after the effect. For instance, in the current example one might want to represent the fact that once the light turns green, it stays green for one minute before changing to red. In our approach, we would use an additional causal constraint which is triggered by a change in the $color(light)$ fluent,

cc2 $\forall t.[t]\ color(light) \hat{=} green \gg$
$[t + 60]\ color(light1) \hat{=} red$

$cc2$ states that if the $color(light)$ fluent changes value from another value to green from $t-1$ to t, then it will turn red 60 seconds later.

In the temporal formalism we use for reasoning about action and change, narratives are represented as action scenario descriptions. Scenario descriptions are partial specifications of initial and other states of a system, combined with descriptions of some of the actions that have occurred together with their timing.

Action scenarios are represented in a surface language $\mathcal{L}(SD)$, which will then be translated into a standard logical language $\mathcal{L}(FL)$. All formal reasoning is done using $\mathcal{L}(FL)$ together with an appropriate circumscription policy for modeling inertia assumptions. Given a scenario description, we are interested in those facts about fluents which may be inferred at different time-points from the set of formulas in $\mathcal{L}(FL)$ representing the scenario.

For instance, the following action scenario in $\mathcal{L}(SD)$ contains a single action occurrence where a button is pressed once, after initially observing that the light is red and the button has not been pressed.

Example 1.1 (Pedestrian Light)

Action Symbols: $Press(button)$.
Feature Symbols: $pressed(button)$, $color(light)$,
Object Symbols: $button1 : button, light1 : light$,

obs1	$[0]\ color(light1) \hat{=} red \wedge \neg pressed(button1)$
occ1	$[2,3]\ Press(button1)$
acs1	$[t_1, t_2]\ Press(button) \rightsquigarrow$
	$[t_1, t_2]\ pressed(button) := T$
cc1	$\forall t.[t]\ pressed(button1) \gg$
	$[t + 30]\ color(light1) \hat{=} green)$
cc2	$\forall t.[t]\ color(light1) \hat{=} green \gg$
	$[t + 60]\ color(light1) \hat{=} red$
cc3	$\forall t.[t]\ pressed(button1) \gg$
	$[t + 1]\ \neg pressed(button) \wedge$

The intended conclusion in this example is that the light is red from 0-32, green from 33-92, and then red from 93 onwards. Of course, for a more realistic model, the behavior of the light would have to be modified appropriately.

Providing a satisfactory language for representing delayed effects of actions is only part of the larger problem involved in providing a useful logic for reasoning about action and change. Our formalism at the very least, should be able to deal with the frame and ramification problems. One of the advantages of using this approach is that it has already been shown in previous papers that it can deal with both the frame and ramification problems in a satisfactory manner for a well defined class of action scenarios ([11], [3], [5]). The class permits use of scenarios with nondeterministic actions, actions with duration, partial specification at any state in the scenario, context dependency, and incomplete specification of the timing and order of actions. The logic has been assessed correct for the $\mathcal{K} - IA$ class of problems using Sandewall's Features and Fluents framework [11]. We are currently working on extensions to the formalism to deal with ramification and concurrency.

In the following sections, we will introduce the logic PMON$^+$, used to reason about scenario descriptions, provide a circumscription policy which can be shown to be reducible to the first-order case, and present a number of example scenarios that show how to reason with both delayed effects and ramification. We conclude with a discussion about a number of difficult open problems related to this particular approach to reasoning about delayed effects of actions.

2 The Language $\mathcal{L}(FL)$

Given an action scenario represented in the language $\mathcal{L}(SD)$, there is a modular mapping into the language $\mathcal{L}(FL)$. $\mathcal{L}(FL)$ is a many-sorted first-order language with equality. We include at least two sorts, $\mathcal{T}$ and $\mathcal{F}$, for temporal and fluent terms. Additional sorts are added as needed for arguments to complex fluent terms such as $color(light1)$. The language includes the predicate symbols $Holds$ and $Occlude$ of sort $\mathcal{T} \times \mathcal{F}$. Informally, $Holds(t, f)$ asserts that the fluent f is true at time t. $Occlude(t, f)$ asserts that the fluent f is not subject to inertia at time t. The intended interpretation for $\mathcal{T}$ is linear discrete time. Additional function and relation symbols are assumed for the particular choice of temporal structure.

3 The Language $\mathcal{L}(SD)$

In the example above, we showed a scenario description represented in the surface language $\mathcal{L}(SD)$. A scenario description consists of action occurrence statements (ac), action law schemas (acs), observation statements (obs), and causal constraint statements (cc). The syntactic transformation from a scenario in $\mathcal{L}(SD)$ to a set of formulas in $\mathcal{L}(FL)$ is straightforward and described in detail in Doherty [2]. We refer the reader to this reference for details and simply provide the translation of example 1.1 below for the necessary intuitions:

Example 3.1 (Pedestrian Light $\mathcal{L}(FL)$)

obs1 $Holds(0, color(light1, red)) \wedge$
$\neg Holds(0, pressed(button1))$

scd1 $Holds(3, pressed(button1)) \wedge$
$\forall t. 2 < t \leq 3 \rightarrow Occlude(t, pressed(button1))$

cc1 $\forall t.(Holds(t, pressed(button1)) \rightarrow$
$Holds(t + 30, color(light1, green))) \wedge$
$\forall t.(0 < t \wedge \neg Holds(t - 1, pressed(button1)) \wedge$
$Holds(t, pressed(button1)) \rightarrow$
$\forall x. Occlude(t + 30, color(light1, x)))$

cc2 $\forall t.(Holds(t, color(light1, green)) \rightarrow$
$Holds(t + 60, color(light1, red))) \wedge$
$\forall t.(0 < t \wedge \neg Holds(t - 1, color(light1, green)) \wedge$
$Holds(t, color(light1, green)) \rightarrow$
$\forall x. Occlude(t + 60, color(light1, x)))$

cc3 $\forall t.(Holds(t, pressed(button1)) \rightarrow$
$\neg Holds(t + 1, pressed(button1))) \wedge$
$\forall t.(0 < t \wedge \neg Holds(t - 1, pressed(button1)) \wedge$
$Holds(t, pressed(button1)) \rightarrow$
$Occlude(t + 1, pressed(button1)))$

For a given action scenario in $\mathcal{L}(FL)$, we use Γ_{obs}, Γ_{scd}, and Γ_{cc}, to denote the sets of observation, schedule, and causal constraint formulas, respectively. Note that in the translation, an action occurrence statement (ac) and its associated action law schema (acs) are replaced with a schedule formula (scd). In addition to these formulas, there are a number of domain independent axioms that are always assumed to be part of any translation of actions scenarios into $\mathcal{L}(FL)$. We denote this set of axioms by Γ_{FA} which includes unique names axioms for the various fluent sorts, value existence axioms for each of the value sorts, unique value axioms for non-boolean propositional fluents such as

$$\forall f, t, d_1, d_2. d_1 \neq d_2 \rightarrow \neg(Holds(t, color(f, d_1)) \wedge Holds(t, color(f, d_2))),$$

and a number of sequentiality and duration axioms to constrain actions not to overlap.

4 Causal Constraints

Causal constraint statements, labeled *cc* in action scenarios are central to our proposal. They not only permit the representation of delayed effects, but are also used to deal with ramification. The definition below in it's full generality is somewhat complex, but the intuition behind it is not.

Definition 4.1
A *causal constraint* statement in $\mathcal{L}(SD)$ has the following form:

$$\forall t_1 \forall \bar{o}.[Q_{\bar{o}_0}.\gamma(\bar{o}_0,\bar{o}) \rightarrow ([t_1]Q_{\bar{o}_1}.\alpha(\bar{o}_1,\bar{o}) \gg [\tau]Q_{\bar{o}_2}.\beta(\bar{o}_2,\bar{o}))]$$

where $\bar{o}_0, \bar{o}_1, \bar{o}_2, \bar{o}$ are disjoint tuples of variables of sorts $\mathcal{O}$, and $Q_{\bar{o}_1}, Q_{\bar{o}_1}, Q_{\bar{o}_2}$ are finite sequences of quantifiers binding the variables $\bar{o}_0, \bar{o}_1, \bar{o}_2$, respectively, α and β are quantifier free fluent formulas , γ is a quantifier free logic formula, and the temporal expression τ is a function g_τ of t_1 where $g_\tau(t_1) \geq t_1$. $\Box$[1]

The translation of a causal constraint into $\mathcal{L}(FL)$ results in a conjunction consisting of two parts, the actual causal dependency and a further constraint which together with the minimization policy constrains the "directionality" of the causal effect. For instance, the causal constraint $cc1$ in example 3.1 has the following translation,

(A) $\forall t.(Holds(t, pressed(button1)) \rightarrow$
$Holds(t+30, color(light1, green)))\wedge$
(B) $\forall t.(0 < t \wedge \neg Holds(t-1, pressed(button1))\wedge$
$Holds(t, pressed(button1)) \rightarrow$
$\forall x.Occlude(t+30, color(light1, x)))$

where (A) represents the logical constraint which asserts that if the button is pressed at t then light1 will turn green at $t+30$. The material implication used in (A) is not strong enough to guarantee the causal directionality that is intended, therefore an additional directional constraint (B) is added which states that if the value of the fluent $pressed(button1)$ changes from false to true from $t-1$ to t then, and *only then*, is a change in value of the fluent $color(light1, green)$ permitted at $t+30$ (unless of course another constraint or action affects the fluent). Note that we occlude all colors with the universal quantifier. This is because fluents with non-boolean value domains can take only one value at a time. When $color(light1, green)$ becomes true, $color(light1, red)$ must become false, so both should be excluded from inertia. The "only then" is guaranteed by the minimization of $Occlude$ which selectively exempts fluent-timepoint pairs from the inertia assumption which we discuss in the next section.

5 PMON Circumscription

Given the $\mathcal{L}(FL)$ translation of an action scenario

$$\Gamma_{FA} \wedge \Gamma_{OBS} \wedge \Gamma_{SCD} \wedge \Gamma_{CC},$$

consisting of the foundational, observation, schedule and causal constraint formulas, we first add an additional *nochange* axiom,

$$\forall t, f.(\neg Occlude(t+1, f) \rightarrow Holds(t, f) \equiv Holds(t+1, f)),$$

denoted Γ_{NCG} to each action scenario. The nochange axiom asserts that if a fluent f is not occluded at time $t+1$ then its value must stay the same from time t to $t+1$. This axiom represents the inertia assumption we use in PMON. The causal constraint and schedule axioms in a scenario assert the sufficient conditions under which a fluent is occluded at a timepoint. The circumscription policy we use will first minimize $Occlude$ relative to just $\Gamma_{SCD} \wedge \Gamma_{CC}$ and then "filter" the resulting models with the nochange axiom which will remove any spurious change not justified by the partitions in the scenario. A beneficial side-effect is that the proper directionality in the causal constraints is encoded in the preferred models.

[1] The distinction between a logic or fluent formula is that a logic formula may contain temporal terms, while a fluent formula is a boolean combination of fluents.

Definition 5.1 (PMON$^+$ Circumscription)

The *PMON$^+$ circumscription* of the action scenario description Υ is

$$\Gamma_{FA} \wedge \Gamma_{NCG} \wedge \Gamma_{OBS} \wedge \Gamma_{AC} \wedge$$
$$Circ_{SO}(\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude); \langle Occlude \rangle),$$

where $Circ_{SO}(\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude); \langle Occlude \rangle)$ is equivalent to

$$\Gamma_{SCD} \wedge \Gamma_{CC} \wedge \forall \Phi. \neg [\Gamma_{SCD}(\Phi) \wedge \Gamma_{CC}(\Phi) \wedge \Phi < Occlude].$$

Definition 5.2 (PMON$^+$ Entailment)
A formula α is said to be *PMON$^+$-entailed by* the action scenario description Υ if

$$\Gamma_{FA} \wedge \Gamma_{NCG} \wedge \Gamma_{OBS} \wedge \Gamma_{AC} \wedge$$
$$Circ_{SO}(\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude); \langle Occlude \rangle) \models \alpha.$$

$Circ_{SO}$ is standard second-order circumscription. The policy minimizes *Occlude* relative to $\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude)$, leaving all other predicates fixed. The result is then filtered with the rest of the scenario formulas together with the nochange axiom.

Due to the simplicity of the circumscription policy, any circumscribed action scenario which is written using the surface language $\mathcal{L}(SD)$ is guaranteed by the following theorem to be reducible to a first-order theory.[2].

Theorem 5.1
Let $\Gamma_{SCD}(Occlude)$ be the result of translating the instantiated action law schemas in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$ and $\Gamma_{CC}(Occlude)$ be the result of translating a conjunction of causal constraint statements in $\mathcal{L}(SD)$ into $\mathcal{L}(FL)$. Then $Circ_{SO}(\Gamma_{SCD}(Occlude) \wedge \Gamma_{CC}(Occlude); \langle Occlude \rangle)$ is reducible to a first-order formula with the following form,

$$\Gamma \wedge \forall t \forall z. (\Delta \leftrightarrow Occlude(t, z)). \Box$$

This theorem together with the nochange axiom provides the basis for reasoning efficiently about inertia conjectures. For example, the definition of *Occlude* for the pedestrian light example which can be automatically generated through syntactic manipulation of the action scenario formulas is:

$$\begin{aligned}
\forall t, f. Occlude(t.f) \leftrightarrow \\
(t = 3 \wedge f = pressed(button1)) \vee \\
(t = 4 \wedge f = pressed(button1)) \vee \\
(\neg Holds(t - 31, pressed(button1)) \wedge \\
Holds(t - 30, pressed(button1)) \wedge t > 30 \\
\wedge \exists x. f = color(light1, x)) \vee \\
(\neg Holds(t - 61, color(light1, green)) \wedge \\
Holds(t - 60, color(light1, green)) \wedge t > 60 \\
\wedge \exists x. f = color(light1, x))
\end{aligned}$$

This formula represents those and only those fluent-timepoint pairs where the fluent is allowed to change value. At all other timepoints the nochange axiom forbids fluents to change value due to the inertia assumption. It is easy to see that the intended conclusions for our scenario are derivable using standard deductive techniques.

[2] Proofs and translation details may be found in [2].

6 Examples

In this section, we will show that modeling delayed effects of actions is closely related to the ramification problem. We do this by taking an example normally used to discuss ramification, but modify it, to include both a delayed effect and the "natural death" of a fluent.

6.1 Drying in the Sun Example

6.1.1 Jump in a Lake Scenario

This is an example concerning the persistence of derived effects of actions [10] In some cases, such as the first example, we expect the indirect effects of actions to persist after the action terminates. In others, such as the second example, we do not. In still other examples, we do expect indirect effects to persist, but only for awhile. The next three examples demonstrate each of these cases. In the last example, we will use a causal rule with delayed effects.

Example 6.1 (Jump in Lake Scenario)

This example is first mentioned in Crawford [1]. The intended conclusion in this example is that a person is still wet after jumping into a lake and then getting out.

Action Symbols: *JumpIn*, *GetOut*.
Feature Symbols: *wet*, *inlake*,

obs1	$[0]\ \neg inlake \wedge \neg wet$
occ1	[2,3] $JumpIn$
occ2	[5,6] $GetOut$
acs1	$[t_1, t_2]\ JumpIn \rightsquigarrow [t_1, t_2]\ inlake := T$
acs2	$[t_1, t_2]\ GetOut \rightsquigarrow [t_1, t_2]\ inlake := F$
cc1	$\forall t.[t]\ inlake \gg [t]\ wet$

Example 6.2 The corresponding set of labeled wffs in $\mathcal{L}(FL)$ for the action scenario is

obs1	$\neg Holds(0, \mathsf{inlake}) \wedge \neg Holds(0, \mathsf{wet})$
scd1	$Holds(3, \mathsf{inlake}) \wedge Occlude(3, \mathsf{inlake})$
scd2	$\neg Holds(6, \mathsf{inlake}) \wedge Occlude(6, \mathsf{inlake})$
cc1	$(\forall t_1.Holds(t_1, \mathsf{inlake}) \rightarrow Holds(t_1, \mathsf{wet})) \wedge$
	$\forall t_1.(\neg Holds(t_1 - 1, \mathsf{inlake}) \wedge Holds(t_1, \mathsf{inlake}))$
	$\rightarrow Occlude(t_1, \mathsf{wet}))$

Example 6.3 (Jump in Lake with Hat Scenario)

This example is discussed in Giunchiglia, et. al. [4]. The intended conclusion in this example is that if a person jumps into a lake with a hat on then the person is still wet after jumping into the lake and then getting out, but the person may not have a hat on any longer.

Action Symbols: *JumpIn*, *GetOut*.
Feature Symbols: *wet*, *inlake*, *hat*

obs1	$[0]\ \neg inlake \wedge \neg wet \wedge hat$
occ1	[2,3] $JumpIn$
occ2	[5,6] $GetOut$
acs1	$[t_1, t_2]\ JumpIn \rightsquigarrow [t_1, t_2]\ inlake := T$
acs2	$[t_1, t_2]\ GetOut \rightsquigarrow [t_1, t_2]\ inlake := F$
cc1	$\forall t.[t]\ inlake \gg [t]\ wet \wedge (hat \vee \neg hat)$

The translation is the same as in the previous example with the exception of $cc1$:

cc1 $(\forall t_1.Holds(t_1, \mathsf{inlake}) \rightarrow Holds(t_1, \mathsf{wet})) \wedge$
$\forall t_1.(\neg Holds(t_1 - 1, \mathsf{inlake}) \wedge Holds(t_1, \mathsf{inlake}))$
$\rightarrow Occlude(t_1, \mathsf{wet}) \wedge Occlude(t_1, \mathsf{hat})$

Note the effect of using $(hat \vee \neg hat)$ in the action scenario. In the translation, it simply has the effect of occluding hat without any additional logical constraint. This implies that whenever there is a change in the value of $inlake$ from false to true, that the status of hat can be either true or false. There are some arguments against using "tautologies" like this because it makes the formalism syntactically sensitive. To neutralize any criticism from this direction, we could always modify the language of the surface language to include a special disjunction macro for this type of use.

Example 6.4 (Drying in the Sun)

A more realistic representation of the problem would be that a person is still wet after getting out of the lake, but only for a few minutes because the sun will dry the person. In our first attempt to represent this, we add an additional causal constraint $cc2$ to that effect.

Action Symbols: $JumpIn$, $GetOut$.
Feature Symbols: wet, $inlake$, hat

obs1 $[0]\ \neg inlake \wedge \neg wet \wedge hat$
occ1 [2,3] $JumpIn$
occ2 [5,6] $GetOut$
acs1 $[t_1, t_2]\ JumpIn \rightsquigarrow [t_1, t_2]\ inlake := T$
acs2 $[t_1, t_2]\ GetOut \rightsquigarrow [t_1, t_2]\ inlake := F$
cc1 $\forall t.[t]\ inlake \gg [t]\ wet \wedge (hat \vee \neg hat)$
cc2 $\forall t.[t]\ \neg inlake \gg [t+5]\ \neg wet$

Although this addition appears to do the job, careful analysis shows that the action scenario is inconsistent. The reason for this is that with the current approach to modeling delayed effects, there is a possibility of intervening events or observations together with inertia which would qualify the causal effect of being dry at a later time-point. In the current example, the logical constraint of $cc2$,

$$\forall t.\neg Holds(t, inlake) \rightarrow \neg Holds(t + 5, wet)$$

interferes with the value of wet at timepoint 5 which is true due to the action of jumping in the lake at time 3 and inertia. Since $inlake$ is observed false at time 0, the logical constraint states that wet should be false at time 5. This implies that casual constraints should have "memory" about previous points in a scenario.

One possibility for modeling this would be to use the context part of a causal constraint which contextualizes when a casual constraint should apply. Here is an alternative representation of $cc2$:

cc2 $\forall t.(\forall t_1.t < t_1 < t + 5 \rightarrow \neg Holds(t_1, inlake))$
$\rightarrow ([t]\ \neg inlake \gg [t + 5]\ \neg wet)$

Roughly, this causal constraint states that "If I get out of the lake and stay out of the lake for 5 minutes, then I will not be wet."

Using this context-dependent causal constraint avoids the possibility of inconsistency. In fact context dependency is a very important feature of this approach as we will see in a later example.

One can view this last problem in at least two ways. In the first, causal constraints behave very much like action effect rules and one might try and deal with *causal qualification* in a manner similar to action qualification. One other way to view the matter is that a delayed causal effect should be interpreted as an action which may occur concurrently with others. Consequently, one

would have to deal with *keep* conditions and *cancelation* conditions among concurrent actions. Dealing with the complexities of this and similar types of interaction between causal rules, action effects, observations, etc. opens up a number of challenging research issues which are not the topic of this paper, but are certainly on the research agenda.

6.2 The Delayed Circuit Scenario

The following is a modification of an example due to Thielscher [12], who uses his example to show that certain formalisms which use minimal change approaches inadequately model indirect effects of actions. The idea is that we are given a circuit with a certain amount of indeterminacy built into it. In our modification we explicitly model the indeterminacy by introducing temporal constants representing the delays in time of different paths through the circuit.

Example 6.5 (Delayed Circuit)

Action Symbols: $SwitchOn(\texttt{switch})$.
Feature Symbols: $on(\texttt{switch})$, $light$, $relay$, $detect$
Object Symbols: $sw1 : \texttt{switch}$, $sw2 : \texttt{switch}$, $sw3 : \texttt{switch}$

obs1	$[0]\ \neg on(sw1) \wedge on(sw2) \wedge on(sw3)$
obs2	$[0]\ \neg light \wedge \neg relay \wedge \neg detect$
occ1	$[2,3]\ SwitchOn(sw1)$
acs1	$[t_1, t_2]\ SwitchOn(sw) \rightsquigarrow [t_1, t_2]\ on(sw) := T$
cc1	$\forall t.\mathtt{t}_1 < \mathtt{t}_2 \rightarrow ([t]\ (on(sw1) \wedge on(sw2)) \gg [t + \mathtt{t}_1]\ light)$
cc2	$\forall t.\mathtt{t}_1 < \mathtt{t}_2 \rightarrow ([t]\ \neg(on(sw1) \wedge on(sw2)) \gg [t + \mathtt{t}_1]\ \neg light)$
cc3	$\forall t.[t]\ (on(sw1) \wedge on(sw3)) \gg [t + \mathtt{t}_2]\ relay$
cc4	$\forall t.[t]\ \neg(on(sw1) \wedge on(sw3)) \gg [t + \mathtt{t}_2]\ \neg relay$
cc5	$\forall t.[t]\ relay \gg [t]\ \neg on(sw2)$
cc6	$\forall t.[t]\ light \gg [t]\ detect$

This a good example where the use of delayed effects can be used to model *causal lag*. If the value of the temporal constant $\mathtt{t}_1$ is less than $\mathtt{t}_2$ then this implies that when $sw1$ and $sw2$ are on that the $light$ will be on for the period $(\mathtt{t}_2 - \mathtt{t}_1)$. At this point the $relay$ kicks in and turns $sw2$ off. When this happens the $light$ will go off $\mathtt{t}_1$ time-points later. This means that the *detector* will go on and stay on even after the $light$ goes off. If the value of the temporal constant $\mathtt{t}_1$ is greater than $\mathtt{t}_2$ then this implies that the $light$ will never go on because the $relay$ kicks in before and turns $sw2$ off. If the value of the temporal constant $\mathtt{t}_1$ is equal to $\mathtt{t}_2$ then the results will be the same.

Without placing additional constraints on $\mathtt{t}_1$ and $\mathtt{t}_2$, the preferred models would provide both types of conclusion, so, the nondeterminism is made explicit in the axioms and the minimization policy in PMON$^+$ does not interfere with the results. This example shows that the combination of explicit time and causal constraints provides a powerful basis for modeling the subtleties of causal lag and transitive change with physical systems.

7 Conclusions

We have proposed a method for representing delayed effects of actions which capitalizes on the use of causal constraints and their interaction with the direct effects of action laws. In the process, we have introduced a versatile logic for reasoning about action and change whose circumscription

is shown to be reducible to the first order case. The logic deals with the frame and ramification problems in a robust manner for a large class of action scenarios. We have also shown how the approach may be used for modeling causal lag and "natural death". It is surprising that not more work has been done with delayed effects of actions. Causal lag would seem to be the rule rather than the exception when modeling physical systems. The fact that the current formalism is based on the use of explicit time contributes to the ease with which delay can be represented for a number of uses. We are currently investigating extensions to the formalism for dealing with concurrent actions and qualification. Results here should be applicable to the problem of "causal qualification" we demonstrated in one of the examples.

References

[1] J. Crawford. Three issues in action. Unpublished note for the 5th Int. Workshop on Non-monotonic Reasoning, 1994.

[2] P. Doherty. PMON+: A Fluent Logic for Action and Change: Formal Specification, Version 1.0. Technical Report LITH-IDA-96-33, Department of Computer and Information Science, Linköping University, Linköping, Sweden, December 1994.

[3] P. Doherty. Reasoning about action and change using occlusion. In *Proceedings of the 11th European Conference on Artificial Intelligence, Aug. 8-12, Amsterdam*, pages 401–405, 1994.

[4] E Giunchiglia and V. Lifschitz. Dependent fluents. In *Proc. of the 14th Int'l Conf. on Artificial Intelligence*, 1995.

[5] J. Gustafsson and P. Doherty. Embracing occlusion in specifying the indirect effects of actions. In *5th International Conference on Principles of Knowledge Representation and Reasoning*, 1996.

[6] V. Lifschitz. Formal Theories of Action. In F. Brown, editor, Proceedings of the 1987 Workshop on the Frame Problem in Artificial Intelligence, Morgan Kaufmann, 1987.

[7] M. Gelfond and V. Lifschitz and A. Rabinov. What are the Limitations to the Situation Calculus? In V. Lifschitz, editor, Artificial Intelligence and the Theory of Computation, Academic Press, 1991.

[8] F. Lin. Embracing Causality in Specifying the Indirect Effects of Actions. Proceedings IJCAI-95, 1995.

[9] N. McCain and H. Turner. A Causal Theory of Ramifications and Qualifications. Proceedings IJCAI-95, pp. 1978-1984, 1995.

[10] K. L. Myers and D. E. Smith. The persistence of derived information. In *Proc. AAAI-93*, pages 496–500, 1988.

[11] E. Sandewall. *Features and Fluents: A Systematic Approach to the Representation of Knowledge about Dynamical Systems.* Oxford University Press, 1994.

[12] M. Thielscher. Ramification and causality. Technical Report TR-96-003, International Computer Science Institute, 1996. Accepted for publication: AI Journal.