Embracing Occlusion in Specifying the Indirect Effects of Actions

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

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

Accepted for publication: Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR'96).

Abstract

In this paper, we extend PMON, a logic for reasoning about action and change, with causal rules which are used to specify the indirect effects of actions. The extension, called PMON(RCs), has the advantage of using explicit time, includes actions with durations, nondeterministic actions, allows partial specification of the timing and order of actions and has been assessed correct for at least the $\mathcal{K}$-IA class of action scenarios within the Features and Fluents framework. Most importantly, the circumscription policy used is easily shown to be reducible to the first-order case which insures that standard theorem proving techniques and their optimizations may be used to compute entailment. In addition, we show how the *occlusion* concept previously used to deal with duration and nondeterministic actions proves to be equally versatile in representing causal constraints and delayed effects of actions. We also discuss related work and consider the strong correspondence between our work and recent work by Lin, who uses a *Cause* predicate to specify indirect effects similar to our use of *Occlude* in PMON, and a minimization policy related to that used in PMON.

1 Introduction

Sandewall [17] has recently proposed a systematic approach to the representation of knowledge about dynamical systems that includes a framework in which to assess the range and applicability of existing and new logics of action and change. As part of the framework, several logics of action and change are introduced and assessed correct for particular classes of action scenario descriptions. The most general class $\mathcal{K} - IA$ and one of it's associated entailment relations, PMON, permits scenarios with nondeterministic actions, actions with duration, partial specifications at any state in the scenario, context dependency, and incomplete specification of the timing and order of actions. Doherty [2, 1] provides a syntactic characterization of PMON in terms of circumscription and classical logic and shows that for the $\mathcal{K} - IA$ class, the circumscription axiom can be reduced to a 1st-order formula. Although PMON is assessed correct for a broad class of action scenarios, it is restricted to actions that do not permit indirect effects. It deals with the simple frame problem and not ramification.

Inspired by the use of the frame construct in Kartha and Lifschitz [8], Doherty and Peppas [4] have extended PMON with the frame construct to deal with various types of ramification. Lifschitz and Kartha capitalize on the *frame* concept discussed in Lifschitz [10] which is used in this case to specify causal dependencies between direct and indirect effects of actions. In addition, they use both a *release* construct and a *filtering* method, analogous to the *occlusion* construct and filtering method first proposed by Sandewall [16, 17] for dealing with postdiction, actions with duration and nondeterminism. The novelty of Kartha and Lifschitz's approach is a tripartite division of fluents into frame, frame released, and non-frame fluents which is used to deal with certain types of causally directed ramification. In Doherty and Peppas[4], we considered the relation between PMON(R) and $\mathcal{AR}_0$. We show that PMON(R) subsumes $\mathcal{AR}_0$ in several different respects, while $\mathcal{AR}_0$ has more expressibility in other respects. The extension, PMON(R), is also characterized in terms of a circumscription axiom, but in contrast to the case for PMON, it can not in the general case be reduced to a first-order formula.

Recently, causal minimization techniques have again become increasingly more popular. These techniques are based on introducing explicit causal predicates or causal rules to specify the indirect effects of actions([11], [12], [15], [20]). In this paper, we will show how the base logic PMON can be extended with *causal rules* with little change to the existing formalism. In fact, causal rules in this context are really nothing more than macros in a surface language which when translated into the base language of PMON take advantage of the already existing predicate *Occlude*. As stated previously,the *Occlude* predicate has already been proven to be quite versatile in specifying actions with duration and indeterminate effects of actions. One of the benefits of using this approach for specifying indirect effects of actions is that the original circumscription policy for the base logic PMON is virtually left intact. Consequently, the extended version which we will call PMON(RCs), inherits the nice feature of allowing the reduction of any circumscribed action theory to a logically equivalent first-order theory provided the axiomatization of the chosen flow of time is first-order definable.

In Doherty and Peppas [4], we showed how the *release* predicate used by Lifschitz and Kartha had the same function as *Occlude* in regard to simple forms of nondeterministic actions. What is even more striking when investigating PMON(RCs) is the comparison with Lin's recent proposals for dealing with indirect effects of actions and indeterminate actions ([11, 12]). In fact, the *Cause* predicate and minimization policy used by Lin are virtually analogous with the use of the *Occlude* predicate and the minimization policy used in both the original PMON [17, 2], and the minor extension made in PMON(RCs).

There are a number of factors which make a formal comparison between the two approaches difficult. In particular, we use a linear time structure, whereas Lin uses the situation calculus, although we should mention that we initiated a study of the proper formal tools needed to do such comparisons in Doherty and Peppas[4]. In addition, Lin also deals with actions which may fail and the qualification problem. We have not yet extended our formalism to deal with these issues. Finally, much of Lin's work deals with the automatic generation of successor state axioms as a means for computing entailment in the situation calculus framework. Although we are currently working on implementations of PMON and its various extensions, formally we have only gone as far as providing an algorithm for automatic reduction of any circumscribed action theory within our framework to the logically equivalent first-order case.

In the rest of the paper, we will do the following: (1) Briefly introduce the base version of PMON. (2) Extend it with causal and acausal constraints resulting in PMON(RCs). (3) Show that any circumscribed action scenario in PMON(RCs) is reducible to the first-order case. (4) Compare PMON(RCs) with a number of recent proposals in the literature, in particular Lin's, Thielscher's and Sandewall's approaches. (5) Conclude with a discussion.

2 Action Scenario Descriptions

Many reasoning problems involving action and change can be conveniently represented in terms of *(action)scenario descriptions* . Scenario descriptions can be described as partial specifications of an initial and other states of a system, combined with descriptions of some of the actions that have occurred together with their timing. The "Yale Shooting" or "Stanford Murder Mystery" problems are well known examples of scenario descriptions. Scenario descriptions can be described directly in terms of a logical language, or for convenience, described first in a higher level macro language which is then compiled into a logical language. In our framework, we will represent action scenarios in a surface language $\mathcal{L}(\mathcal{SD})$, which will then be translated into a standard logical language $\mathcal{L}(\mathcal{FL})$. All formal reasoning will be done using $\mathcal{L}(\mathcal{FL})$ together with appropriate circumscription policies for modeling various inertia policies. Detailed descriptions of both languages and the translation process may be found in [1, 2]. In the following sections, we will provide sufficient detail to follow the examples in this paper.

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

The formal syntax for specifying scenario descriptions is defined in terms of the surface language $\mathcal{L}(\mathcal{SD})$, consisting of action occurrence statements, action law schemas, and observation statements, labeled with the symbols "ac", "acs", and "obs", respectively. The well known Stanford Murder Mystery scenario is shown below using the $\mathcal{L}(\mathcal{SD})$ syntax:[1]

Example 1

$obs0 \quad \mathtt{t}_0 = 0 \land \mathtt{t}_1 = 10$

[1] Note that $obs0$ is not strictly necessary. It is used here simply to show that observation statements may contain arbitrary temporal constraints associated with a particular scenario.

*obs*1	$[\mathtt{t}_0]\mathit{alive}$
*obs*2	$[\mathtt{t}_1]\neg \mathit{alive}$
*ac*1	$[2,6]\mathit{Fire}$
*acs*1	$[t_1,t_2]\mathit{Fire} \rightsquigarrow ([t_1]\mathit{loaded} \rightarrow$ $[t_1,t_2](\mathit{alive} := F \wedge \mathit{loaded} := F)$

Given a scenario description Υ, consisting of statements in the surface language $\mathcal{L}(\mathcal{SD})$, these statements are translated into formulas in the many sorted first-order language $\mathcal{L}(\mathcal{FL})$ via a two-step process. In the first step, action schemas in Υ are instantiated with each action occurrence statement of the same name, resulting in what are called *schedule statements* (Each schedule statement is labeled with the symbol "scd"). The resulting schedule statements replace the action schemas and action occurrence statements. The result of the first step is an *expanded (action) scenario description*, Υ', consisting of both schedule and observation statements. The expanded scenario description associated with Example 1 is shown below:

Example 2

*obs*0	$\mathtt{t}_0 = 0 \wedge \mathtt{t}_1 = 10$
*obs*1	$[\mathtt{t}_0]\mathit{alive}$
*obs*2	$[\mathtt{t}_1]\neg \mathit{alive}$
*scd*1	$([2]\mathit{loaded} \rightarrow$ $[2,6](\mathit{alive} := F \wedge \mathit{loaded} := F)$

In the second step of the translation process, macro-translation definitions are used to translate statements in Υ' into formulas in $\mathcal{L}(\mathcal{FL})$. Before applying this step to the example, we must first define $\mathcal{L}(\mathcal{FL})$.

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

$\mathcal{L}(\mathcal{FL})$ is a many-sorted first-order language. For the purposes of this paper, we assume two sorts: a sort $\mathcal{T}$ for time and a sort $\mathcal{F}$ for propositional fluents. In other work [6], an additional sort is used for actions. The language includes the predicate symbols *Holds* and *Occlude* , of type $\mathcal{T} \times \mathcal{F}$, and the predicate symbols $<$ and $\leq$ (interpreted as the usual "less than" and "less than or equal to" relations on natural numbers) of type $\mathcal{T} \times \mathcal{T}$, the equality predicate $=$, and the function symbols $+$, and $-$ (interpreted as the usual "plus" and "minus" functions on natural numbers) of type $\mathcal{T} \times \mathcal{T} \rightarrow \mathcal{T}$.

The numerals $0, 1, 2, \ldots$ and the symbols $\mathtt{t}_0, \mathtt{t}_1, \ldots$, will be used to denote constants of type $\mathcal{T}$ and the symbols $t_0, t_1, \ldots$ will be used to denote variables of type $\mathcal{T}$. We define the *set of temporal terms* to be the closure of the temporal variables and temporal constants of the language under the operators $+$ and $-$. A propositional fluent is a function of time with the boolean truth values as range. The symbols $f_1, f_2, \ldots$ will be used to denote variables of type $\mathcal{F}$. We assume an appropriate set of function symbols of proper arity for fluent names (e.g. *alive*, *at*). If F is a fluent name then a restricted fluent term is defined as a term with form $F(u_1, \ldots, u_n)$, where $u_1, \ldots, u_n$ are terms of sort *object*, where *object* is restricted to be either a variable or a constant term. Restricted fluent terms of arity 0 are called *fluent constants.* We define the *set of fluent terms* to be the union of fluent variables and restricted fluent terms.

An *atomic formula* is defined as any formula of the form $\mathit{Holds}(\mathbf{t}, \mathbf{f})$ or $\mathit{Occlude}(\mathbf{t}, \mathbf{f})$, where $\mathbf{t}$ is a temporal term and $\mathbf{f}$ is a fluent term. The set of formulas of $\mathcal{L}(FL)$ is defined as the closure of the atomic formulas under the boolean connectives (i.e. $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$) and the universal and existential quantifiers (i.e. $\forall$, $\exists$). In what follows, $\mathbf{t} < \mathbf{t}' < \mathbf{t}''$, $\mathbf{t} \leq \mathbf{t}' < \mathbf{t}''$, $\mathbf{t} < \mathbf{t}' \leq \mathbf{t}''$ and $\mathbf{t} \leq \mathbf{t}' \leq \mathbf{t}''$, stand for $\mathbf{t} < \mathbf{t}' \wedge \mathbf{t}' < \mathbf{t}''$, $\mathbf{t} \leq \mathbf{t}' \wedge \mathbf{t}' < \mathbf{t}''$, $\mathbf{t} < \mathbf{t}' \wedge \mathbf{t}' \leq \mathbf{t}''$ and $\mathbf{t} \leq \mathbf{t}' \wedge \mathbf{t}' \leq \mathbf{t}''$, respectively.

The intended interpretation for $\mathcal{T}$ is linear discrete time where $\mathcal{T}$ is considered isomorphic to the natural numbers. Since there is no axiomatization for time interpreted as the natural numbers, we either assume an interpreted language, settle for something less such as "integer-like, discrete flow of time with a first moment" which is axiomatizable [14], or assume a sound, but incomplete axiomatization of the flow of time. In practice, we will normally be using a specialized temporal constraint module for reasoning about time which is normally sound, but incomplete.

2.3 From $\mathcal{L}(SD)$ to $\mathcal{L}(FL)$

As stated in Section 2.1, the second step of the translation process uses macro-translation definitions to translate statements in Υ' into formulas in $\mathcal{L}(FL)$. We need a few preliminary definitions which will prove useful both here and in the definition of causal constraints in a later section. A *fluent formula* is any boolean combination of restricted fluent terms from $\mathcal{L}(FL)$. An *elementary scenario formula* is of the form $[t]\gamma$, where t is a temporal term in $\mathcal{L}(FL)$ and γ is a fluent formula. A *scenario formula* is any boolean combination of elementary scenario formulas. Let γ and δ denote fluent formulas and ϵ denote a restricted fluent term (possibly negated). Let C be any of the logical connectives $\wedge$, $\vee$, or $\rightarrow$. The following list of macro-translation definitions should suffice to provide the general idea:

$$[t](\delta \ C \ \gamma) \overset{def}{=} [t]\delta \ C \ [t]\gamma.$$

Any elementary scenario formula can be reduced to a boolean combination of elementary scenario formulas of the form $[t]\epsilon$.

$$\begin{aligned}
[s,t]\delta &\overset{def}{=} \forall x. s \leq x \leq t \rightarrow [x]\delta \\
[s,t)\delta &\overset{def}{=} \forall x. s \leq x < t \rightarrow [x]\delta \\
[t]\epsilon &\overset{def}{=} Holds(t, \epsilon) \\
[s,t]\epsilon := T &\overset{def}{=} Holds(t, \epsilon) \\
& \quad \wedge \forall t_1 (s < t_1 \leq t \rightarrow Occlude(t_1, \epsilon)) \\
[s,t]\epsilon := F &\overset{def}{=} Holds(t, \neg\epsilon) \\
& \quad \wedge \forall t_1 (s < t_1 \leq t \rightarrow Occlude(t_1, \epsilon)) \\
Holds(t, \neg\epsilon) &\overset{def}{=} \neg Holds(t, \epsilon).
\end{aligned}$$

The translation of Υ' in Example 2 into $\mathcal{L}(FL)$ using the translation rules is shown below:

Example 3

$obs0$ $\quad \mathtt{t}_0 = 0 \wedge \mathtt{t}_1 = 10$

$obs1$ $\quad Holds(\mathtt{t}_0, alive)$

$obs2$ $\quad \neg Holds(\mathtt{t}_1, alive)$

$scd1$ $\quad Holds(2, loaded) \rightarrow$
$[\neg Holds(6, alive) \wedge \neg Holds(6, loaded) \wedge$
$\forall t (2 < t \leq 6 \rightarrow Occlude(t, alive)) \wedge$
$\forall t (2 < t \leq 6 \rightarrow Occlude(t, loaded))]$

Note that although the labels are not part of the language of $\mathcal{L}(FL)$, they are retained. The labels are directly correlated with the partitioning of formulas used in the circumscription policy described in the next section. The following notation

$$\Gamma_C = \Gamma_{OBS} \cup \Gamma_{SCD} \cup \Gamma_{UNA} \cup \Gamma_T,$$

is used for a scenario description in $\mathcal{L}(FL)$, where Γ_{OBS} and Γ_{SCD} contain the observation and schedule statements in the scenario, Γ_T contains the axiomatization for the flow of time (when provided), and Γ_{UNA} contains the appropriate unique name axioms for the sorts $\mathcal{T}$ and $\mathcal{F}$. In the rest of the paper, we will use the convention of suppressing Γ_{UNA} and Γ_T, assuming they are provided with every theory. In addition, we will use the notation Γ_X, where X is an acronym such as OBS, for a finite set of formulas or their conjunction in contexts where this makes sense.

3 PMON Circumscription

In this section, we will describe the intuition behind the use of occlusion, introduce a Nochange Axiom, describe the filtered minimization technique, provide a circumscription policy which uses occlusion, the Nochange Axiom, and filtering, and show that any theory in the $\mathcal{K} - IA$ class of action scenarios is reducible to a first-order theory.

3.1 Occlusion

As we already mentioned in the introduction, the use of the occlusion concept and its representation in terms of the predicate *Occlude* has already proven to be quite versatile in providing solutions to a number of open problems associated with the representation of action and change. Although related to the use of an abnormality predicate together with an inertia assumption, there are some differences. The main difference is perspective. *Occlude* is used to provide a fine-grained means of excluding particular fluents at particular points in time from being subject to what are normally very strong inertia assumptions. In fact, in retrospect much of the progress made in solving a number of problems stemming from the original Yale Shooting Scenario has been the gradual relaxation of strict inertia in dealing with non-determinism, postdiction and in the current case, indirect effects and delayed effects of actions. It is the fine-grained use of *Occlude* together with the filtered minimization technique, where *Occlude* is minimized in only parts of theories, that contributes to the simplicity of the solutions. Filtering minimizes the need for complex minimization strategies. In fact, most of the time, the minimization policy involves little more than applying predicate completion to *Occlude* relative to part of a theory.

Recall the scenario description described in Section 2.3. Associated with each action type in a scenario is a subset of fluents that are potentially influenced by the action (those fluents in the right side of a rule). If the action has duration, then during its execution, it is not known in general what value the influenced fluents have. Since the action performance can potentially change the value of these fluents at any time, all that can generally be asserted is that at the end of the duration the fluent is assigned a specific value. To specify such behavior, the *Occlude* predicate is used in the definition of reassignment expressions which in turn are used as part of the definition of an action schema. *Occlude* serves the purpose of excluding certain features at certain time-points from the global inertia policy which we will soon introduce. In order to specify actions with durations and indeterminate effects of actions properly, it should be clear that fluents directly set by an action should be occluded during the execution of the action. In Lifschitz's recent terminology, occluded fluents are simply frame-released fluents.

The predicate *Occlude* takes a time-point and a fluent as arguments. The definition for a reassignment expression $[t_1, t_2]\delta := T$ used in an action occurrence statement is[2]

$$Holds(t_2, \delta) \wedge \forall t(t_1 < t \leq t_2 \rightarrow Occlude(t, \delta)).$$

Referring to our previous example, it can be observed that the occlusion specification is automatically generated by the translation process from $\mathcal{L}(SD)$ to $\mathcal{L}(FL)$. Occlusion specifies what fluents may change at what points in time. The Nochange Axiom described next specifies when a fluent is *not* permitted to change value.

3.2 The PMON Circumscription Policy

Let Γ_{NCG} denote the following Nochange Axiom,

$$\forall f, t(Holds(t, f) \oplus Holds(t{+}1, f) \rightarrow Occlude(t{+}1, f)), \tag{1}$$

where the connective $\oplus$ is an abbreviation for the exclusive-or connective. The axiom states that if a fluent f is not occluded at $t + 1$ then it can not change value from t to $t + 1$. This axiom, together with the observation axioms will be used to filter potential histories of action scenarios.

Filtered preferential entailment is a technique originally introduced by Sandewall [16] for dealing with postdiction. The technique is based on distinguishing between different types of formulas in a scenario description and applying minimization to only part of the scenario, or different minimization policies to different parts of the scenario. In this particular case, we will distinguish between schedule statements Γ_{SCD} and the rest of the scenario. The idea is minimize the *Occlude* predicate relative to the schedule statements and then filter the result with the observation formulas, Γ_{OBS} and the nochange axiom Γ_{NCG}. The minimization policy generates potential histories where the potential for change is minimized. The potential histories are then filtered with the observations, which must hold in any valid history, and with the nochange axiom which filters out any spurious change not explicitly axiomatized by the actions. More formally, instead of using the policy,

$$\begin{aligned} &\Gamma_{NCG} \wedge \Gamma_C \wedge \\ &\quad Circ_{SO}(\Gamma_{NCG} \wedge \Gamma_C(Occlude); Occlude), \end{aligned} \tag{2}$$

[2]For $[t_1, t_2]\delta := F$, simply negate the *Holds* predicate.

where $Circ_{SO}$ denote standard second-order circumscription, PMON circumscription is defined using the policy,

$$\Gamma_{NCG} \wedge \Gamma_C \wedge Circ_{SO}(\Gamma_{SCD}(Occlude); Occlude). \tag{3}$$

Observe that the circumscription policy is surprisingly simple, yet at the same time assessed correct for the broad ontological class $\mathcal{K}$-*IA*. One simply minimizes the *Occlude* (frame-released) predicate while leaving *Holds* fixed in that part of the theory containing the action occurrences and then filters the result with the nochange (inertia) axiom and the observation axioms.

Although $Circ_{SO}(\Gamma_{SCD}(Occlude); Occlude)$ is a second-order formula, it can be shown that it is equivalent to a first-order formula using two results by Lifschitz [9], and the fact that *Occlude*-atoms only occur positively in Γ_{SCD}, or through the use of predicate completion. Details may be found in [2, 1]. In the following sections, we will show that this condition is satisfied even after PMON is extended for ramification.

4 Extending PMON for Ramification

Details regarding the work described so far can be found in previous publications by our group. We will now proceed to the main topic of this paper, that of extending PMON to PMON(RCs) in order to deal with indirect effects of actions. The ramification problem states that it is unreasonable to explicitly specify all the effects of an action in the action specification itself. One would rather prefer to state the direct effects of actions in the action specification and use the deductive machinery to derive the indirect effects of actions using the direct effects of actions together with general knowledge of dependencies among fluents, specified as domain constraints. The dependencies specified using domain constraints do not necessarily have to be based solely on notions of physical causality.

The idea is that there is a certain tiered precedence for change where change for fluents in a certain class are dependent on changes of fluents in another class but not vice-versa. Which fluents have precedence over others is of course a domain dependent call and that information must be provided in some manner, for example, either explicitly in terms of causal rules, or perhaps implicitly in terms of partitioning fluents in classes such as framed, frame-released and non-framed as is the case with [8], and using a particular minimization policy. A particular domain policy might be based on physical causality, where the precedence is for causes to have a stronger inertia quota than their dependencies. On the other hand, when explaining effects, one might reverse the precedence. A good example is the domain constraint $light \equiv switch1 \vee switch2$ where causal flow is in the right-left direction, but one might equally well reverse the precedence for actions which turn a light on instead.

One of the difficulties in dealing with the ramification problem is the fact that a tension exists between solving the standard frame problem which requires minimizing change across the board, and attacking the ramification problem which requires relaxing minimization of change *just enough* to permit change for indirect effects, but only indirect effects that have a justification for changing. As mentioned above, it is still an open issue as to what policies such justifications should be based on. Physical causality is simply one of several reasons one might set up dependencies between fluents. Sandewall [19] and Thiels cher [20] have recently begun analyzing different policies for justifying dependencies between fluents.

The basis of our solution is a straightforward encoding of relaxing minimization of change "just enough" for the indirect effects of actions. This will be done by introducing causal rules in $\mathcal{L}(SD)$ which provide a means of expressing the directionality of dependencies between fluents and defining their translations into $\mathcal{L}(FL)$ in terms of the *Occlude* predicate, which excludes the indirect effects from the nonchange constraint in the nochange axiom Γ_{NCG}. We will begin by distinguishing between two types of domain constraints, *causal constraints* and *acausal constraints*, and then specifying both their representations in $\mathcal{L}(SD)$ and translations into $\mathcal{L}(FL)$, in terms of the *Occlude* predicate.

4.1 Causal and Acausal Constraints

In order to express causal constraints, we begin by defining causal relation expressions in $\mathcal{L}(SD)$.

Definition 1 A *causal relation* is an expression in $\mathcal{L}(SD)$ with the following form,

$$[t]\delta \gg [s]\gamma,$$

where both t and s are temporal terms in $\mathcal{L}(FL)$, $t \leq s$, and δ and γ are fluent formulas. $\Box$

The intended meaning of a causal relation is "if the fluent formula δ is true at time-point t, then the fluent formula γ must be true at time-point s, and a change in γ due to this rule is legal w.r.t the nochange premise".

Definition 2 A *causal constraint* is an expression in $\mathcal{L}(SD)$ with the following form,

$$Q(\alpha \rightarrow [t]\delta \gg [s]\gamma) \text{ or } Q([t]\delta \gg [s]\gamma),$$

where α is a scenario formula in which for any temporal term t' in α,$t' \leq t$, and Q is a sequence of quantifiers binding free variables of sorts $\mathcal{F}, \mathcal{T}$. $\Box$

We call α the *precondition* for the causal constraint. The use of preconditions permits the representation of context dependent dependencies among fluents. Note also, that because the formalism uses explicit time and both t and s may refer to different time-points, it is straightforward to represent delayed effects of actions using causal constraints. We will demonstrate this in the examples which follow in the next section. Causal constraints will be labeled with the prefix *csc* in scenario descriptions in $\mathcal{L}(SD)$. The following expressions provide examples of conditionalized and nonconditionalized causal constraints:

$$\text{csc} \quad \forall t([t]underwater \rightarrow ([t]breathing \gg [t+\mathtt{t}_0]\neg alive))$$
$$\text{csc} \quad \forall t([t]\neg alive \gg [t]\neg walking),$$

Acausal constraints describe relations between fluents without encoding any preference for dependency ordering. In a sense, a special class of acausal constraints is not really necessary. Technically, they could be defined as observations in a scenario which are observed at all time-points, but conceptually there is a difference.

Definition 3 An *acausal constraint* is an expression in $\mathcal{L}(SD)$ which is an arbitrary quantified scenario formula. $\Box$

Acausal constraints will be labeled with the prefix "acc" in $\mathcal{L}(SD)$. We call an acausal constraint where all fluents have the same temporal term, a *static domain constraint*, while those with several terms will be called *transition constraints.* The following expressions provide examples of static and transition acausal constraints, respectively:

$$\text{acc} \quad \forall t([t]\neg black \vee [t]\neg white)$$
$$\text{acc} \quad \forall t([t]\neg alive \rightarrow [t+1]\neg alive).$$

4.2 Translation into $\mathcal{L}(FL)$

In the previous section, we defined a number of different domain constraint types in the surface language . We will now provide a translation into $\mathcal{L}(FL)$ and show that our informal intuitions regarding dependency preferences among constraints are formally encoded into $\mathcal{L}(FL)$. Note that the only new symbol introduced when defining our domain expressions is the symbol $\gg$. The following macro-translation definition provides the proper translation for this relation.

Definition 4

$$[t]\delta \gg [s]\gamma \overset{def}{=} [(([t]\delta \rightarrow [s]\gamma) \wedge \quad (4)$$
$$(([t-1]\neg\delta \wedge [t]\delta) \rightarrow [s]X(\gamma))], \quad (5)$$

where $t \leq s$ and $X(\gamma)$ denotes the occlusion of all restricted fluent terms in γ at s.[3] □

This intermediate formula is then translated into $\mathcal{L}(FL)$ in a manner similar to that described in the example in Section 2.3.

Example 4 The following causal constraint expression specified in $\mathcal{L}(SD)$,

$$\text{csc} \qquad \forall t[t](\neg alive \gg \neg walking)^4,$$

is translated into the following $\mathcal{L}(FL)$ formula,

$$\begin{aligned} \text{csc} \quad & \forall t\{ \\ & (\neg Holds(t, alive) \rightarrow \neg Holds(t, walking)) \wedge \\ & (Holds(t-1, alive) \wedge \neg Holds(t, alive) \rightarrow \\ & \qquad Occlude(t, walking)).\} \end{aligned}$$

□

The intuition behind the translation in Definition 4 is as follows: the first part of Definition 4, (4), represents the actual causal dependency. $[t]\delta \rightarrow [s]\gamma$ forces γ to be true if δ is true. The second part (5) simply justifies the changes caused by the first part, but only in the appropriate causal direction. $([t-1]\neg\delta \wedge [t]\delta) \rightarrow [s]Occlude(\gamma)$, states that a change in γ caused by the rule is legal w.r.t the nochange axiom.

4.3 PMON(RCs) Circumscription

The language of scenario descriptions $\mathcal{L}(SD)$ has been extended for causal and acausal constraints and additional macro-translation definitions have been introduced which permit the translation of scenario descriptions with domain constraints into $\mathcal{L}(FL)$. The final step will be to extend the circumscription policy used in PMON to accommodate these new changes.

Let Γ_{ACC} and Γ_{CSC} denote the translations of the acausal and causal constraints into $\mathcal{L}(FL)$, respectively. An action scenario Γ_C is now defined as $\Gamma_C = \Gamma_{FIL} \cup \Gamma_{CHG}$ where,

$$\Gamma_{FIL} = \Gamma_{NCG} \cup \Gamma_{OBS} \cup \Gamma_{ACC} \cup \Gamma_{UNA} \cup \Gamma_T$$

[3]For example, if γ is $\epsilon_1 C \epsilon_2$, C is a logical connective, then $[s]X(\gamma)$ is $Occlude(s,\epsilon_1) \wedge Occlude(s,\epsilon_2)$.

[4]In the rest of the paper $[t](\delta \gg \gamma)$ will often be used instead of $[t]\delta \gg [t]\gamma$.

$$\Gamma_{CHG} = \Gamma_{SCD} \cup \Gamma_{CSC}.$$

The new circumscription policy is defined as

$$\Gamma_{FIL} \wedge Circ_{SO}(\Gamma_{CHG}(Occlude); Occlude), \qquad (6)$$

Note that since Γ_{CHG} contains no negative occurrences of $Occlude$, and all other predicates are fixed, any action scenario in PMON(RCs) is provably reducible to a logically equivalent first-order theory. The first-order reduction applies to the extended scenarios without change.

5 Examples

In this section, we will consider two examples from the literature, the latter slightly modified from the original. These examples should help in acquiring both a conceptual and technical understanding of PMON(RCs).

Example 5 The walking turkey problem is a well-known ramification scenario and relatively straightforward to encode and solve using causal constraints. We will require one causal constraint stating that dead turkeys do not walk. One ramification of shooting a turkey is that it no longer walks and our theory should entail this indirect effect. The following action scenario description in $\mathcal{L}(SD)$ describes the walking turkey problem:

$obs1$	$[0]walking$
$ac1$	$[2,4]Shoot$
$acs1$	$[s,t]Shoot \rightsquigarrow [s,t]alive := F$
$csc1$	$\forall t[t]\neg alive \gg \neg walking$

The corresponding translation into $\mathcal{L}(FL)$ is,

$$\begin{aligned} obs1 \quad & Holds(0, walking) \\ scd1 \quad & \neg Holds(4, alive) \wedge \forall t(2 < t \leq 4 \rightarrow \\ & \quad Occlude(t, alive)) \\ csc1 \quad & \forall t[(\neg Holds(t, alive) \rightarrow \neg Holds(t, walking)) \wedge \\ & \quad (Holds(t-1, alive) \wedge \neg Holds(t, alive) \rightarrow \\ & \qquad Occlude(t, walking))] \end{aligned}$$

The scenario is partitioned as:

$\Gamma_{FIL} = \Gamma_{NCG} \cup \Gamma_{OBS} \cup \Gamma_{ACC} \cup \Gamma_{UNA} \cup \Gamma_T$, where

$\Gamma_{OBS} = \{obs1\}, \Gamma_{UNA} = \{alive \neq walking\}, \Gamma_{ACC} = \{\}$.

$\Gamma_{CHG} = \Gamma_{SCD} \wedge \Gamma_{CSC}$, where

$$\Gamma_{SCD} = \{scd1\}, \Gamma_{CSC} = \{csc1\}.$$

Circumscribing $Occlude$ in Γ_{CHG} results in the following definition which can be generated using either our algorithm [5], or standard predicate completion:

$$\begin{aligned}\forall t, f(Occlude(t,f) \leftrightarrow \quad & ((f = alive \wedge 2 < t \leq 4) \vee \\ & (f = walking \wedge \\ & \quad Holds(t-1, alive) \wedge \\ & \quad \neg Holds(t, alive))))\end{aligned}$$

For readability, we list the complete translation of the scenario in $\mathcal{L}(FL)$, with the definition of $Occlude$ derived via circumscription:

$$\begin{aligned}& \forall f, t(Holds(t,f) \oplus Holds(t+1,f) \rightarrow \\ & \quad Occlude(t+1,f)) \wedge \\ & Holds(0, walking) \wedge \\ & alive \neq walking \wedge \\ & \neg Holds(4, alive) \wedge \\ & \forall t[(\neg Holds(t, alive) \rightarrow \neg Holds(t, walking)) \wedge \\ & \forall t, f(Occlude(t,f) \leftrightarrow ((f = alive \wedge 2 < t \leq 4) \vee \\ & \qquad (f = walking \wedge Holds(t-1, alive) \wedge \\ & \qquad \neg Holds(t, alive))))\end{aligned}$$

There are two classes of preferred models for this scenario due to the fact that the shoot action has duration and its effects may occur at time point 3 or 4:

$$\begin{aligned}& [0,2] alive \wedge walking \\ & [3,\infty] \neg alive \wedge \neg walking\end{aligned}$$

and

$$\begin{aligned}& [0,3] alive \wedge walking \\ & [4,\infty] \neg alive \wedge \neg walking\end{aligned}$$

Example 6 The extended stuffy room problem is based on the original problem due to Winslett [21]. We extend it by introducing a box which requires the use of chained causal rules, and by encoding a delayed effect of an action by using one of the causal rules.

In this scenario, there are two causal constraints. The first asserts that $duct2$ is blocked if something is put on top of it ($loc2$). The second asserts that the room gets stuffy two time-points after both ducts are blocked. Between time-points 2 and 5 a box is moved to location $loc2$. The action scenario is represented in $\mathcal{L}(SD)$ as:

$$\begin{array}{ll} obs & [0](blocked(duct1) \wedge \neg blocked(duct2) \wedge \\ & at(box, loc1) \wedge \neg stuffy) \\ ac & [2,5] move(box, loc1, loc2) \\ acs & \forall x, l1, l2([s,t] move(x, l1, l2) \leadsto \\ & \quad [s,t] at(x, l1) := F \wedge [s,t] at(x, l2) := T) \\ csc1 & \forall t, x[t](at(x, loc2) \gg blocked(duct2)) \\ csc2 & \forall t([t](blocked(duct1) \wedge blocked(duct2)) \gg \\ & \quad [t+2] stuffy) \end{array}$$

Translating into $\mathcal{L}(FL)$ and circumscribing, results in the following theory:

$$\begin{aligned}& Holds(0, blocked(duct1)) \wedge \\ & \neg Holds(0, blocked(duct2)) \wedge \\ & Holds(0, at(box, loc1)) \wedge \\ & \neg Holds(0, stuffy) \wedge \\ & \neg Holds(5, at(box, loc1)) \wedge \\ & Holds(5, at(box, loc2)) \wedge \\ & \forall t, x(Holds(t, at(x, loc2)) \rightarrow \\ & \quad Holds(t, blocked(duct2))) \wedge \\ & \forall t(Holds(t, blocked(duct1)) \wedge \\ & \quad Holds(t, blocked(duct2)) \rightarrow \\ & \quad Holds(t+2, stuffy)) \wedge \\ & \forall f, t(Holds(t,f) \oplus Holds(t+1,f) \rightarrow \\ & \quad Occlude(t+1, f)) \wedge \\ & \forall t, f(\\ & \quad Occlude(t,f) \leftrightarrow \\ & \quad (2 < t \leq 5 \wedge f = at(box, loc1)) \vee \\ & \quad (2 < t \leq 5 \wedge f = at(box, loc2)) \vee \\ & \quad \forall x(\neg Holds(t-1, at(x, loc2)) \wedge \\ & \qquad Holds(t, at(x, loc2)) \wedge \\ & \qquad f = blocked(duct2)) \vee \\ & \quad (\neg[Holds(t-3, blocked(duct1)) \wedge \\ & \qquad Holds(t-3, blocked(duct2))] \wedge \\ & \quad Holds(t-2, blocked(duct1)) \wedge \\ & \quad Holds(t-2, blocked(duct2)) \wedge \\ & \quad f = stuffy) \\ &)\end{aligned}$$

In addition, the unique name axioms are included, but not listed here. The theory correctly entails that if something is placed on top of $duct2$ while $duct1$ is blocked, then the room will become stuffy two time-points later. In the current axiomatization, the room would remain stuffy forever even after one of the ducts became free. This is because the nochange axiom prevents the room from becoming unstuffy without reason. In the current theory, there is no axiom which states otherwise. A rule stating that the room is stuffy only when both ducts are blocked can be added without difficulty:

$$csc3 \quad \forall t([t]\neg(blocked(duct1) \wedge blocked(duct2)) \gg [t+2]\neg stuffy).$$

This example demonstrates both the use of casual chaining and how causal constraints can be used to specify delayed effects of actions. □

6 Causal constraints and Stratification

Causal cycles without delay cause spontaneous or non-grounded change, as the example below illustrates.

Example 7

$$\begin{array}{ll} \text{obs1} & [0]\neg\alpha \\ \text{csc1} & \forall t[t](\alpha \gg \alpha) \end{array} \tag{7}$$

Unfortunately, this allows a model where α holds at time-point 1, without any outside cause. Where intuitively we would assume that $\neg\alpha$ persists, because there is no reason for it not to, the causal rule introduces unsupported occlusion, blocking this preferred entailment.□

Lin solves this problem by demanding that the causal constraints are stratified, the following definition is from Lin [11], it has been slightly changed for our terminology.

Definition 5 The csc's are stratified if there are no fluents $F_0, F_1, ..., F_n$ such that $F_0 \rightarrow F_1 \rightarrow ... \rightarrow F_n \rightarrow F_0$, where for any fluents F and F', $F' \rightarrow F$ if there is a causal relation such that F' appears in the left hand side of the $\gg$ sign, and F appears in the right hand side of the causal relation. □

Lin requires this definition, not only to rule out non-grounded change, but to avoid cycles or recursion in his causal rules, which prevents the use of Clark's completion when reducing 2nd-order theories. Note that we are never in this position because the translation of causal constraints in our formalism only generates positive occurrences of *Occlude* and the use of restricted fluent terms prevents any recursion in *Occlude* even if negative occurrences were generated.

One way to view stratification is to think in terms of positive and negative recursion. *Positive recursion* occurs when F_0 has the same sign at the beginning and end of a chain like that in Definition 5. A typical positive recursion is shown in Example 7. On the other hand, *Negative recursion* occurs when F_0 has different signs at the beginning and end of a chain. The rules $\alpha \gg \beta, \beta \gg \neg\alpha$ together provide an example of negative recursion. The current stratification definition can be viewed as a means of ruling out both types of recursion.

Although one might have a difficult time finding an example where negative recursion makes sense with causal rules, allowing it in our formalism does not cause any technical problems. For example, the first part of the translation of $\forall t[t](\alpha \gg \neg\alpha)$ is $\forall t([t]\alpha \rightarrow [t]\neg\alpha$ which is equivalent to $\forall t[t]\neg\alpha$ and the lhs of the second part of the translation is always false, so no additional occlusion assertions are generated.

As shown in the example, positive recursion when allowed, may result in unintuitive and unintended models, where a causal constraint in some sense triggers itself. On the other hand, if we restrict ourselves to causal rules which may generate positive recursive cycles, but where the cause occurs strictly before the effect, then the presence of positive cycles is technically unproblematic.

Consequently, PMON(RCs), does not require a stratification definition as restrictive as the one above, but the set of non-delayed causal constraints (constraints in which $t = s$ in Definition 2) must not contain positive recursion.

The requirement that we exclude all types of non-delayed positive recursive causal constraints when using our approach is unfortunate because this would rule out using a number of useful domain constraints in both directions such as $alive \leftrightarrow \neg dead$. If we encode this formula using causal constraints in an attempt to permit precedence in both directions the result would be positive non-delayed cycles, which we have already stated generate unintuitive models. Let's take another example. Suppose we would like to encode the constraint $lamp \leftrightarrow switch_1 \vee switch_2$, so it may be used in the causal direction (right-to-left) and for explanation in the other direction (left-to-right) using causal rules. The following rules would be needed:

$$\begin{array}{ll} csc1 & \forall t[t](la \gg (sw_1 \vee sw_2)) \\ csc2 & \forall t[t](\neg la \gg \neg(sw_1 \vee sw_2)) \\ csc3 & \forall t[t]((sw_1 \vee sw_2) \gg la) \\ csc4 & \forall t[t](\neg(sw_1 \vee sw_2) \gg \neg la), \end{array}$$

Unfortunately, these are positive recursive and would not do the job. In other words, it is very difficult to use the causal rule approach in general when we are dealing with constraints without causal interpretations. Sandewall [19] discusses this in detail. This is obviously a limiting feature of the current approach and approaches like ours.

7 Related work

7.1 Lin

Previously, we mentioned that if one disregards failed actions and qualification when comparing Lin's recent proposals for dealing with indirect effects of actions and indeterminate actions([11]), then there are striking similarities between these proposals and the original and extended versions of PMON. In fact, the *Cause* predicate and minimization policy used by Lin are virtually analogous with the use of the *Occlude* predicate and minimization policy used in both the original PMON and its extension PMON(RCs). Assuming familiarity with [11], let's briefly compare the two frameworks. We'll begin with syntax and then discuss minimization policy.

Compare Lin's frame axiom:

$$\begin{aligned}Poss(a,s) \rightarrow \{ \quad & \neg(\exists v)Caused(p,v,do(a,s)) \rightarrow \\ & [Holds(p,do(a,s)) \leftrightarrow Holds(p,s)]\},\end{aligned}$$

which is equivalent to

$$\begin{aligned}Poss(a,s) \rightarrow \{ \quad & [Holds(p,do(a,s)) \oplus Holds(p,s)] \\ & \rightarrow (\exists v)Caused(p,v,do(a,s))\}\end{aligned}$$

with the nochange axiom in PMON:

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

$Poss(a,s)$ has to do with failed actions so this can be disregarded.

In Lin's example, a typical causal domain constraint is represented as,

$$up(L1,s) \wedge up(L2,s) \rightarrow Caused(open,true,s)$$

An additional rule is used to transfer change in the *Caused* context back to the *Holds* context:

$$Caused(p,true,s) \rightarrow Holds(p,s),$$

which can be used to rewrite the previous causal constraint to the equivalent,

$$\begin{aligned}\forall s \quad & ([up(L1,s) \wedge up(L2,s) \rightarrow Holds(open,s)] \wedge \\ & [up(L1,s) \wedge up(L2,s) \rightarrow Caused(open,true,s)]),\end{aligned}$$

which would be quite similar to our representation of a causal constraint as,

$$\begin{aligned}\forall t \quad & ([[t]up(L1) \wedge [t]up(L2) \rightarrow [t]open] \wedge \\ & [[t-1]\neg(up(L1) \wedge up(L2)) \wedge \\ & [t](up(L1) \wedge up(L2)) \rightarrow \\ & \qquad [t]Occlude(open)].\end{aligned}$$

Lin describes his minimization method as follows:

1. Start with a theory T that includes all the effect axioms and state constraints.
2. Minimize *Caused* in T. Let T' be the resulting theory.
3. Add to T' the frame axiom(10). Let T'' be the resulting theory.
4. Maximize *Poss* in T'' to obtain the final action theory.

Again, we can disregard step 4 in the comparison, which leaves us with steps 1-3. Disregarding what we place in parentheses for the moment, PMON's original minimization policy is,

1. Start with a theory T that includes all the schedule axioms Γ_{SCD}, (and state constraints Γ_{csc}).
2. Minimize *Occlude* in T. Let T' be the resulting theory.
3. Add to T' the nochange axiom Γ_{NCG} and observations Γ_{OBS}. Let T'' be the resulting theory.

To acquire the extended minimization policy for PMON(RCs), simply remove the parentheses in step 1.

One difference between Lin's approach and ours (and we are sure there may be more) is that Lin keeps fluents *Caused* as long as the constraint is active, while we only *Occlude* when the actual change of value takes place. The nochange axiom takes care of the continued behavior.

7.2 Thielscher

Another interesting approach to ramification is suggested by Thielscher [20]. He regards the direct effects obtained after firing an action merely as a preliminary approximation to the resulting situation which is then computed by performing additional post-processing steps that generate indirect effects relative to domain constraints specified as causal rules. For each action, there is a state representing the direct effects of the action followed by a sequence of states representing the progression towards a state that is legal w.r.t the domain constraints[5]. In order to "fire" causal rules in the right order and direction, additional knowledge about the directionality of causation (*influence relationships*) has to be specified. These influence relationships together with the domain constraints are compiled into

[5]In the rest of this section we will call sequences like these ramification sequences

causal relations, which are then used to progress between states in the ramification sequence.

Thielscher does not require stratified theories, this is possible since each application of a causal relation leads to a new state, so non-grounded change is ruled out by definition.

Thielscher's approach, although very promising, has a number of limitations as regards expressiveness. For example, all actions must be deterministic and non-deterministic domain constraints sometimes lead to unintuitive results. For example the causal constraint $light \gg sw_1 \vee sw_2$ if represented in Thielscher's approach using the causal rules, $\neg s_1 : l \rightsquigarrow s_2$ and $\neg s_2 : l \rightsquigarrow s_1$, and these are applied sequentially, then it appears that turning on the *light* would not result in any states where both switches would be on. It also appears that causal rules leading to negative recursion could lead to a situation where no stable state is generated, although this problem may be dealt with using the logic programming technique which implements his formal specification.

7.3 Sandewall

In [19], Sandewall presents the *transition cascade semantics* which is an extension of previous work with the Features and Fluents framework. His approach to providing an underlying semantics for dealing with various types of ramification has some similarities with Thielscher's approach. Like Thielscher he considers the direct effects of an action as an approximation to the actual resulting situation. Actions are specified using an *action invocation relation* $D(E, r, r_1)$, where E is an action, r is the state where the action E is invoked, and r_1 is the new state where the instrumental part of the action has been executed. After the action has been fired, a binary, non-reflexive *causal transition relation* C is used to construct a ramification sequence $r_1, r_2, ..., r_k$ where $C(r_{i,i+1})$ for every i between 1 and $k-1$, and r_k is a stable state. r_k is considered as the result state of the action E in situation r. The set of result states is denoted $N(E, r)$. The idea is that result states contain both direct and indirect effects of an action occurrence.

The concept of *respectful* action systems is also introduced where any fluent in a ramification sequence may only change value once. This constraint, in effect, rules out negative recursion, but only with respect to each action. No action may lead to a ramification sequence where a fluent changes value more than once. Domain constraints may still contain negative recursion, but these cycles cannot be reached.

Sandewall's work is relatively new, but we believe the extended underlying semantics he proposes will be a useful tool for assessing the correctness of PMON(RCs) and determining what classes of action scenarios the logic can be shown correct. His current analysis of existing proposals should also provide a direct means of comparing our work with these other approaches.

8 Conclusion

We feel that PMON and it's extension PMON(RCs) have a number of advantages over other formalisms for specifying action scenarios. They use explicit time in terms of a linear metric time structure which allows one to specify actions with duration, delayed effects of actions, and the incomplete specification of timing and order of actions in a straightforward manner. We've argued that there is a simple and intuitive surface language for describing scenarios and that for this particular class of scenario descriptions, we can reduce the circumscribed scenarios to the first-order case algorithmically.

On the other hand, we have not claimed that this logic is suitable for all classes of problems, nor that it has solved the frame and ramification problems in total, whatever they may be defined as being for the moment. For future work, we would like to assess just what class of scenarios this particular logic functions properly for, using the Features and Fluents framework. Finally, we hope that the comparisons we have made with other formalisms contribute towards progressing in a forward direction by building not only on our own work, but also on the work of others.

References

[1] P. Doherty. Notes on PMON Circumscription. Technical report, Department of Computer and Information Science, Linköping University, Sweden, 1994.

[2] 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.

[3] P. Doherty and W. Łukaszewicz. Circumscribing Features and Fluents. In D. Gabbay and H.J. Ohlbach, editors, *Proceedings of the 1st International Conference on Temporal Logic*, volume 827 of *Lecture Notes in Artificial Intelligence*. Springer, 1994.

[4] P. Doherty and P. Peppas. A comparison between two approaches to ramification: PMON(R) and $\mathcal{AR}_0$. In *Proceedings of the 8th Australian Joint Conference on Artificial Intelligence*, 1995.

[5] P. Doherty, W. Łukaszewicz, and A. Szałas. Computing Circumscription Revisited: Preliminary Report. In *Proceedings of the 14th Int'l Joint Conference on Artificial Intelligence*, volume 2, pages 1502–1508, 1995. Extended version to appear in Journal of Automated Reasoning.

[6] P. Doherty and W. Łukaszewicz and A. Szalas. Explaining Explanation Closure. In *Proc. ISMIS-96* , 1996.

[7] J. Finger. *Exploiting Constraints in Design Synthesis.* PhD thesis, Department of Computer Science, Stanford University, Stanford, CA, 1986.

[8] G.N. Kartha and V. Lifschitz. Actions with Indirect Effects (Preliminary Report). In *International Conference on Principles of Knowledge Representation and Reasoning*, 1994.

[9] V. Lifschitz. Pointwise Circumscription. In M. Ginsberg, editor, *Readings in Non-monotonic Reasoning.* Morgan Kaufmann, 1988.

[10] V. Lifschitz. Frames in the Space of Situations. *Artificial Intelligence*, 46:365–376, 1990.

[11] F. Lin. Embracing Causality in Specifying the Indirect Effects of Actions. *In Proceedings of 14th International Joint Conference on Artificial Intelligence*, 1995.

[12] F. Lin. Embracing Causality in Specifying the Indeterminate Effects of Actions. *Proc. AAAI'96*, 1996.

[13] F. Lin and R. Reiter. State Constraints Revisited. *Journal of Logic and Computation, Special Issue on Actions and Processes*, 1994.

[14] D. Gabbay and I. Hodkinson and M. Reynolds. Temporal Logic, Mathematical Foundations and Computational Aspects, Vol. 1. Oxford University Press, 1994.

[15] N. McCain and H. Turner. A Causal Theory of Ramifications and Qualifications. *In Proceedings of 14th International Joint Conference on Artificial Intelligence*, 1995.

[16] E. Sandewall. Filter Preferential Entailment for the Logic of Action in almost Continuous Worlds. *Proc. of IJCAI'89*, Morgan Kaufmann, 1989.

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

[18] E. Sandewall. Systematic Comparison of Approaches to Ramification using Restricted Minimization of Change. Technical report, Department of Computer and Information Science, Linköping University, Sweden, 1995.

[19] E. Sandewall. Assessments of ramification methods that use static domain constraints In *International Conference on Principles of Knowledge Representation and Reasoning* Morgan Kaufmann, 1996.

[20] M. Thielscher. Computing Ramifications by Postprocessing. In C. S. Mellish, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1994–2000, Montreal, Canada, August 1995. Morgan Kaufmann.

[21] M. Winslett. Reasoning about actions using a possible models approach. In *National Conference on Artificial Intelligence*, 1988.