A general first-order solution to the ramification problem with cycles

Hannes Strass[a,*], Michael Thielscher[b]

[a]*Institute of Computer Science, Leipzig University*
Augustusplatz 10, 04103 Leipzig, Germany
[b]*School of Computer Science and Engineering, The University of New South Wales*
Sydney, NSW 2052, Australia

Abstract

We provide a solution to the ramification problem that integrates findings of different axiomatic approaches to ramification from the last ten to fifteen years. For the first time, we present a solution that: (1) is independent of a particular time structure, (2) is formulated in classical first-order logic, (3) treats cycles – a notoriously difficult aspect – properly, and (4) is assessed against a state-transition semantics via a formal correctness proof.

This is achieved as follows: We introduce indirect effect laws that enable us to specify ramifications that are triggered by activation of a formula rather than just an atomic effect. We characterise the intended models of these indirect effect laws by a state-transition semantics. Afterwards, we show how to compile a class of indirect effect laws into first-order effect axioms that then solve the ramification and frame problems. We finally prove the resulting effect axioms sound and complete with respect to the semantics defined earlier.

Keywords: reasoning about actions, indirect effects, ramification problem, causality, cyclic dependencies

1. Introduction

The ramification problem is concerned with how to represent, and reason about, the indirect effects of actions and events. It has been identified more than 25 years ago by Ginsberg and Smith [16] as a necessary consequence of solving the classical frame problem [30]. The latter is concerned with limiting the update of a state to the direct effects of an action, but Ginsberg and Smith [16] have observed that this may result in states that violate fundamental state constraints. They have tried to reconcile this by inferring additional, implicit effects from the state constraints. But it soon became clear that state constraints alone do not provide a sufficiently accurate representation for deriving the correct indirect effects [24].

Since then, research into the ramification problem has made significant progress. Most of the earlier results concern the *representational* aspect after several authors found that some form of causal information is necessary to be able to correctly infer the ramifications of actions [25, 29, 37]. More recently, the attention has turned to the *inferential* aspect of the ramification problem, which is concerned with how to draw the right conclusions from the causal information. Cyclic

[*]Corresponding author
Email addresses: `strass@informatik.uni-leipzig.de` (Hannes Strass), `mit@cse.unsw.edu.au` (Michael Thielscher)

dependencies between causes and effects are the most difficult issue in this regard, because they can easily lead to inferring phantom effects that would not occur in reality [8].

Existing approaches to the ramification problem can therefore be assessed in two dimensions: according to the expressiveness of their underlying representation technique, and whether or not they can handle cycles.

- The following approaches all use causal laws in which ramifications are triggered by the activation of a formula:
  - Lin's approach [25] is tied to the action formalism of the Situation Calculus and does not allow for cycles.
  - Lin and Soutchanski [27] extend the work of Lin [25] to cope with cycles.
  - Shanahan's solution [35] is tied to the Event Calculus and does not cope with cycles.
  - McIlraith [31] disallows cycles and thinks of ramification as syntactical refinement of state constraints, where the aim is to satisfy these constraints.
- Thielscher [37, 38] uses causal laws with indirect effects being triggered by a literal and checked against an intermediate (possibly unstable) context.
- Denecker et al. [8] show how the principle of inductive definition can treat the ramification problem. In their work, indirect effects are triggered by a formula and checked against an initial context. Due to the constructiveness of inductive definitions, the approach can deal with cycles.

In this paper, we build on much of this past research and combine and extend it to the first general axiomatic first-order approach to the ramification problem that can handle cycles. The key features of our solution can be summarised as follows.

1. It provides a general representation technique that combines trigger formulas with both an initial and a terminal context.
2. It is independent of a specific action formalism.
3. It provides a general first-order solution to the problem of inferring indirect effects in the presence of cycles.

All this is achieved as follows.

We begin by defining a special-purpose action language for representing knowledge of causes and effects that enables us to specify ramifications that are triggered by activation of a formula rather than just an atomic effect. Moreover, each potential ramification can be embedded in both the initial and the terminal context of a transition. We then characterise the intended models of these indirect effect laws by a state-transition semantics.

Turning to the inferential aspect of the ramification problem, we develop a method for compiling indirect effect laws into effect axioms that then solve the ramification and frame problems. These effect axioms are formalised in our unifying action calculus, which is not confined to a particular time structure and can be instantiated with different action formalisms like the Situation Calculus or the Event Calculus [39].

For the compilation, special care needs to be taken to deal with positive cyclic fluent dependencies, that is, self-supporting effects. Using techniques from logic programming, we identify positive loops among indirect effects and build their corresponding loop formulas [28] into the effect axiom. We evaluate our first-order solution to the inferential ramification problem by proving the resulting effect axioms sound and complete with respect to the state-transition semantics defined earlier.

Structure of this paper. The next section provides the necessary background on the unifying action calculus and on the answer set semantics for logic programs. In the section afterwards, we introduce indirect effect laws and their semantics. In Section 4, we provide an axiomatic combined solution to the frame and ramification problems. We start out with how we axiomatise persistence and direct effects, then how we include indirect effects and finally and most importantly how we deal with circular dependencies among effects. After proving the correctness of our axiomatic solution, we show how to express arbitrary trigger formulas in our formalism. We then give a detailed discussion of related work and conclude.

2. Background

In the following, we will employ sorted first-order logic with equality. We assume the language contains the fixed formulas $\top, \bot$ for truth and falsity, respectively. More complex compound formulas may use negation $\neg$, conjunction $\wedge$ and disjunction $\vee$, and universal $\forall$ and existential $\exists$ quantification. For literals of the language, we use a meta-level negation $\bar{\cdot}$ which is defined by $\overline{\neg P} \overset{\text{def}}{=} P$ and $\overline{P} \overset{\text{def}}{=} \neg P$; it carries over to sets L of literals by $\overline{L} \overset{\text{def}}{=} \{\overline{\psi} \mid \psi \in L\}$. Furthermore, each literal has a *sign* defined by $sign(P) \overset{\text{def}}{=} +$ and $sign(\neg P) = -$. First-order structures are denoted by $\mathfrak{M}$, and interpretations $\mathfrak{I}$ consist, as usual, of a structure $\mathfrak{M}$ and a variable valuation $\mathfrak{V}$.

2.1. Unifying Action Calculus

The unifying action calculus (UAC) [39] has the objective to bundle research efforts in action formalisms. It does not confine to a particular time structure and can be instantiated with situation-based action calculi, like the Situation Calculus or the Fluent Calculus, as well as with formalisms using a linear time structure, like the Event Calculus.

The UAC is based on a finite sorted signature with predefined sorts for the basic building blocks of virtually all action formalisms: time itself, world properties that change over time (fluents), and actions, that initiate these changes.

Definition 2.1. A *domain signature* is a signature which includes the sorts TIME, FLUENT and ACTION along with the predicates

- $<$: TIME $\times$ TIME denoting an ordering of time points,
- *Holds* : FLUENT $\times$ TIME stating whether a fluent evaluates to true at a given time point,
- *Poss* : ACTION $\times$ TIME $\times$ TIME indicating whether an action is applicable for particular starting and ending time points.

The following definition introduces the most important types of formulas of the unifying action calculus: they allow to express properties of states and applicability conditions and effects of actions.

Definition 2.2. Let $\vec{s}$ be a sequence of variables of sort TIME.

- A *state formula* $\Phi[\vec{s}]$ *in* $\vec{s}$ is a first-order formula with free variables $\vec{s}$ where
  - for each occurrence of $Holds(f, s)$ in $\Phi[\vec{s}]$ we have $s \in \vec{s}$ and
  - predicate *Poss* does not occur.

Let s, t be variables of sort TIME and A be a function into sort ACTION.

- A *UAC precondition axiom* is of the form

$$Poss(A(\vec{x}), s, t) \equiv \pi_A[s] \tag{1}$$

where $\pi_A[s]$ is a state formula in s with free variables among $s, t, \vec{x}$.

- A *UAC effect axiom* is of the form

$$Poss(A(\vec{x}), s, t) \supset \Upsilon_1[s,t] \vee \ldots \vee \Upsilon_k[s,t] \tag{2}$$

where $k \geq 1$ and each Υ_i $(1 \leq i \leq k)$ is a formula of the form

$$\begin{aligned}(\exists \vec{y}_i)(\Phi_i[s] \wedge (\forall f)(\Gamma_i^+ \supset Holds(f,t)) \\ \wedge (\forall f)(\Gamma_i^- \supset \neg Holds(f,t)))\end{aligned} \tag{3}$$

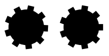
in which $\Phi_i[s]$ is a state formula in s with free variables among $s, \vec{x}, \vec{y}$, and both $\Gamma^+[s,t]$ and $\Gamma^-[s,t]$ are state formulas in s, t with free variables among $f, s, t, \vec{x}, \vec{y}$.

Example 2.3 (Gear Wheels [8])**.** Two gear wheels can be separately turned and stopped. Let the fluents $\mathsf{W}_1, \mathsf{W}_2$ express that the first (respectively second) wheel is turning, then $Holds(\mathsf{W}_1, s)$ and $\neg Holds(\mathsf{W}_2, s)$ are examples of state formulas in s. Let the actions to initiate and stop turning be Turn_i and Stop_i (for $i = 1, 2$) with precondition axioms

$$Poss(\mathsf{Turn}_i, s, t) \equiv \neg Holds(\mathsf{W}_i, s) \qquad Poss(\mathsf{Stop}_i, s, t) \equiv Holds(\mathsf{W}_i, s)$$

The (direct) effects of the actions can be formalised by these UAC effect axioms:

$$\begin{aligned}Poss(\mathsf{Turn}_i, s, t) \supset ((\forall f)(f = \mathsf{W}_i \vee Holds(f,s) \supset Holds(f,t)) \wedge \\ (\forall f)(f \neq \mathsf{W}_i \wedge \neg Holds(f,s) \supset \neg Holds(f,t)))\end{aligned}$$



$$\begin{aligned}Poss(\mathsf{Stop}_i, s, t) \supset ((\forall f)(f \neq \mathsf{W}_i \wedge Holds(f,s) \supset Holds(f,t)) \wedge \\ (\forall f)(f = \mathsf{W}_i \vee \neg Holds(f,s) \supset \neg Holds(f,t)))\end{aligned}$$

The effects are unconditional and deterministic, hence in both cases there is only one sub-formula Υ_1 (cf. (2)), whose condition $\Phi_1[s]$ (cf. (3)) is equivalent to $\top$ and therefore can be omitted. The reader may note that the two effect axioms imply the two wheels to be independent, because it follows, for example,

$$Poss(\mathsf{Turn}_1, s, t) \vee Poss(\mathsf{Stop}_1, s, t) \supset (Holds(\mathsf{W}_2, t) \equiv Holds(\mathsf{W}_2, s))$$

provided that $W_1 \neq W_2$. Later in this paper we will consider the ramification problem that arises from interlocking the wheels.

Next, we formalise the concept of an (action) domain axiomatisation with its notion of time and action laws.

Definition 2.4. Consider a fixed UAC domain signature. A *UAC domain axiomatisation* is a set of axioms $\Sigma = \Omega \cup \Pi \cup \Upsilon \cup \Sigma_{aux}$, where

- Ω is a finite set of foundational axioms that define the underlying time structure,
- Π is a set of precondition axioms (1),

- Υ is a set of effect axioms (2),
- Π and Υ contain exactly one axiom for each function into sort ACTION and
- Σ_{aux} is a set of auxiliary axioms containing unique-names axioms for sorts FLUENT and ACTION.

2.2. Logic Programming

Since the underlying idea of our axiomatic solution to the ramification problem with cycles uses ideas from answer set programming, we first introduce the original concepts from logic programming.

Definition 2.5. For a propositional or first-order signature, a *definite logic program rule* is of the form $H \leftarrow B_1, \ldots, B_m$ where $H, B_1, \ldots, B_m$ are atoms; H is the *head* and the B_i are the *body* atoms. A *definite logic program* Λ is a set of definite logic program rules.

A rule is called *ground* if it does not contain variables. This notion carries over to logic programs. To define the semantics of a ground logic program Λ, one approach is to define an operator T_Λ that maps sets of ground atoms to sets of ground atoms, where

$$T_\Lambda(S) = \{H \mid H \leftarrow B_1, \ldots, B_m \in \Lambda, \{B_1, \ldots, B_m\} \subseteq S\}$$

that is, the operator returns all atoms which can be derived from the atoms in the input set using rules from the program. A set S of atoms is then a *supported model* of the program iff it is a fixpoint of T_Λ, that is, if $T_\Lambda(S) = S$. The axiomatic counterpart to this fixpoint property is given by Clark's completion [6], which syntactically transforms a program into a set of logical equivalences such that the models of these equivalences correspond one-to-one with the supported models of the program.

For definite logic programs Λ, the operator T_Λ is monotone and so possesses a least fixpoint, which can be regarded as the canonical or intended model of the program. Once negation is allowed in rule bodies, however, the operator need not be monotone any more and the existence of fixpoints (i.e. models of the program) cannot be guaranteed. What is more, supported model semantics in general does not check for atoms that cyclicly depend on each other. This may lead to unintuitive and unintended results when logic programs are used for knowledge representation purposes.

Gelfond and Lifschitz [13] introduced the stable model semantics, that checks for cyclic self-support in a declarative way. For a given signature, a *normal logic program rule* is of the form

$$H \leftarrow B_1, \ldots, B_m, \mathit{not}\ C_1, \ldots, \mathit{not}\ C_n$$

where $H, B_1, \ldots, B_m, C_1, \ldots, C_n$ are atoms – $B_1, \ldots, B_m$ are the *positive body atoms* and $C_1, \ldots, C_n$ are the *negative body atoms*. A variable occurs *positively* in a rule if it occurs in a positive body atom. A variable occurs *negatively* in a rule if it occurs in the head or in a negative body atom. A rule is called *safe* if any variable that occurs negatively in the rule also occurs positively. A variable is called *local* if it occurs in the body but not in the head.

A *normal logic program* is a set of normal logic program rules. A normal logic program is *ground* if all its rules are ground.

Definition 2.6 (Answer Set)**.** Let Λ be a ground normal logic program and M be a set of ground atoms. M is an answer set for Λ iff M is the least fixpoint of T_{Λ^M}, where the definite program Λ^M is obtained from Λ by (1) eliminating each rule whose body contains a literal *not* C with $C \in M$, and (2) deleting all literals of the form *not* C from the bodies of the remaining rules.

This definition guarantees that – under the assumption that all atoms not in M are false – there is a constructive proof for all atoms *in* M.

Loops and Loop Formulas. Lin and Zhao [28] discovered an interesting relationship between Clark's completion for definite logic programs [6] and the answer set semantics. The relationship focuses on dependencies among program atoms, where P *depends on* Q if there is a program rule with head P such that Q is among the rule's positive body atoms. Answer set semantics implicitly "checks" for positive cyclic dependencies among program atoms and disallows answer sets that contain unjustified self-referential loops. As Clark completion does not check for cycles, it may allow models that do not correspond to answer sets. Lin and Zhao [28] showed that adding so-called loop formulas for each loop of Λ to the Clark completion, there is a one-to-one correspondence between the models of the resulting propositional theory and the answer sets of Λ. So this propositional theory forms the axiomatic counterpart of the answer set semantics.

Chen et al. [5] later generalised the notion of a loop for programs with variables to avoid computing essentially the same loops on different ground instantiations of a program. They provided a theoretical result that guarantees the existence of a finite, complete set of loops for programs over relational signatures.[1] In their work, program clauses with variables are viewed as representatives of their ground instances.

Definition 2.7. Let Λ be a logic program over a relational signature Ξ. The *positive dependency graph* G_Λ for Λ is the (possibly infinite) graph (V, E) where V is the set of atoms over Ξ and for $\mu, \nu \in V$, we have $(\mu, \nu) \in E$ iff there is a program rule $H \leftarrow B_1, \ldots, B_m, \textit{not } C_1, \ldots, \textit{not } C_n \in \Lambda$ and a substitution θ such that $H\theta = \mu$ and $B_i\theta = \nu$ for some $1 \leq i \leq m$. A finite, non-empty subset L of V is called a *first-order loop of* Λ iff for all $\mu, \nu \in L$, there is a directed, non-zero length path from μ to ν in the subgraph of G_Λ induced by L. For two sets of literals L_1, L_2 we say that L_1 *subsumes* L_2 if there is a substitution θ with $L_1\theta = L_2$. A set $\mathfrak{L}$ of loops of a program Λ is *complete* iff for each loop L of Λ there is an $L' \in \mathfrak{L}$ that subsumes L.

The main result of Chen et al. [5] identifies a syntactical class of programs for which the answer sets can be captured by a finite logical theory that is independent of a specific ground instantiation.

Lemma 2.8. *[5, Proposition 9] Let Λ be a normal logic program over a relational signature. Λ has a finite, complete set of loops if no rule in Λ contains local variables.*

3. An Action Language with Indirect Effects

Much like the frame problem, the ramification problem of reasoning about actions has aspects that relate to specification and representation of as well as reasoning about effects. Where the frame problem is concerned with world properties that do *not* change, the ramification problem is about world properties that change *indirectly* – that is, not as a direct result of applying the action but rather as a result of a change induced by the action. The main task here is to efficiently deal with domino effects, where a single change can initiate arbitrarily complex chains of indirect effects. Specifying these effects should be possible in a modular, elaboration-tolerant fashion. For example, it should be sufficient to state indirect effect relations only for immediate neighbours in the effect chain and have the theory figure out what actually happens. After all (direct and indirect) effects have been determined, the persistence assumption can be employed

[1]Although finite, the size of the set of loops may in the worst case be exponential in the number of program atoms for propositional programs already.

to complete the knowledge about the resulting time point with those world properties that have not been affected by direct or indirect effects.

We will now introduce an action language for the purpose of specifying and reasoning about dynamic domains with indirect action effects. Building upon a vocabulary of fluents and actions, it offers a more or less standard way of expressing the direct effects of actions. Below, a *fluent formula* is like an ordinary first-order formula with the difference that FLUENT terms play the role of atomic formulas; consequently a fluent literal is a fluent or its negation.

Definition 3.1. Consider a fixed UAC domain signature. Let A be a function into sort ACTION and $\vec{x}$ be a sequence of variables matching A's arity.

- A *precondition law for* $A(\vec{x})$ is of the form $\underline{\texttt{possible}}$ $A(\vec{x})$ $\underline{\texttt{iff}}$ Φ_A where Φ_A is a fluent formula with free variables in $\vec{x}$.
- A *direct effect law for* $A(\vec{x})$ is of the form $\underline{\texttt{action}}$ $A(\vec{x})$ $\underline{\texttt{causes}}$ ψ $\underline{\texttt{if}}$ Φ where ψ (the *effect*) is a fluent literal and Φ (the *condition*) is a fluent formula.

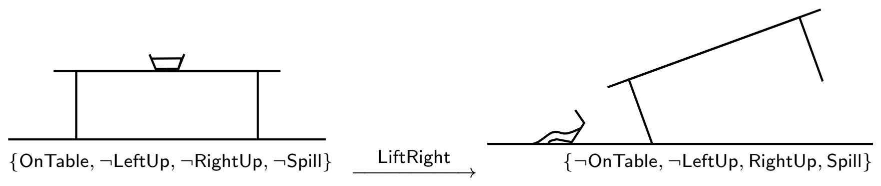
An *action domain specification (ADS)* Θ – for short, domain – is a finite set of precondition and direct effect laws.

The variables $\vec{x}$ matching an action A's arity are placeholders for the arguments of the action. The arguments can be understood as those domain objects which are directly affected by A. The action's precondition may refer to these arguments, but may not contain any further free variables. Effect laws for action A may refer to arbitrary variables, in particular to those among $\vec{x}$. After compilation to an effect axiom, the variables in $\vec{x}$ are universally quantified from the outside. This expresses the intuition that precondition and effect laws hold for any ground instance of the action.

We assume without loss of generality that different effect laws for the same action $A(\vec{x})$ share only variables among $\vec{x}$, and that these constitute the only pairs of elements of action domain specifications that share variables. In view of our later solution to the combined frame and ramification problems, the consequence of an effect law must be a literal, not complex formulas. While this prevents the use of disjunctions to represent actions with indeterminate effects, these actions can be easily modelled within such a setting following Lin's [26] approach of using multiple effect laws, one for each possible effect, as illustrated later in this section with Example 3.7.

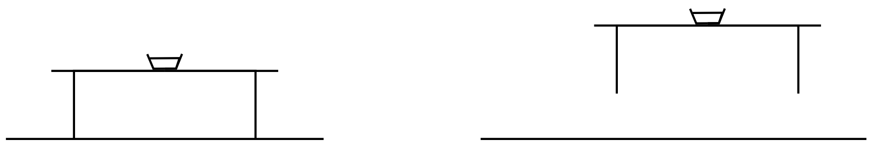
Example 3.2 (Bowl Position)**.** Consider the following simple domain that involves a table and a bowl of soup.[2] Without wishing to anticipate subsequent definitions, we exemplify our desired state transition semantics in the following figures.

If the soup is on the table and just one side of the table is lifted while the other stays on the ground, the bowl falls off and the soup will be spilled.


{OnTable, ¬LeftUp, ¬RightUp, ¬Spill}
LiftRight
{¬OnTable, ¬LeftUp, RightUp, Spill}

[2]For the purpose of this paper, we focus only on the ramification aspects of this domain, and not on its concurrency aspects. In principle, our approach to ramification can be combined with any equally general approach to concurrency to treat both aspects.

If however both sides of the table are lifted at the same time, there will be no spill and the bowl stays on the table.



$$\{\mathsf{OnTable}, \neg\mathsf{LeftUp}, \neg\mathsf{RightUp}, \neg\mathsf{Spill}\} \xrightarrow{\mathsf{LiftBoth}} \{\mathsf{OnTable}, \mathsf{LeftUp}, \mathsf{RightUp}, \neg\mathsf{Spill}\}$$

If the soup is not on the table, there will be no spill whatsoever.

$$\{\neg\mathsf{OnTable}, \neg\mathsf{LeftUp}, \neg\mathsf{RightUp}, \neg\mathsf{Spill}\} \xrightarrow{\mathsf{LiftRight}} \{\neg\mathsf{OnTable}, \neg\mathsf{LeftUp}, \mathsf{RightUp}, \neg\mathsf{Spill}\}$$

With the equipment introduced in Definition 3.1 above, we can model this domain Θ_{Spill} in our action language as follows. The FLUENT sort consists of the functions $\mathsf{OnTable}$ (the soup is on the table) $\mathsf{LeftUp}, \mathsf{RightUp}$ (the left, resp. right side of the table is lifted), Spill (the soup is spilled); available ACTION functions are $\mathsf{LiftLeft}, \mathsf{LiftRight}$ for lifting either side of the table and $\mathsf{LiftBoth}$ for lifting both sides simultaneously. The effects of the last action are clearly specified by $\underline{\texttt{action}}$ $\mathsf{LiftBoth}$ $\underline{\texttt{causes}}$ LeftUp and $\underline{\texttt{action}}$ $\mathsf{LiftBoth}$ $\underline{\texttt{causes}}$ $\mathsf{RightUp}$.

To formalise the semantics of action effects, we begin with how we capture the truth values of all domain-relevant fluents at a specific time point via so-called states. A state represents a snapshot of the properties of the world at a certain point in time. Under the assumption that all objects in the domain are designated by ground terms, the semantics of closed fluent formulas can be defined in a standard way.

Definition 3.3. Consider a fixed UAC domain signature Ξ. A *state* is a maximal consistent set of ground fluent literals. The *satisfaction relation* $\models$ between a state S and ground fluent atoms φ and fluent formulas Φ_1, Φ_2 is recursively defined by

$$S \models \top \text{ and } S \not\models \bot$$
$$S \models \varphi \text{ iff } \varphi \in S$$
$$S \models \neg\Phi_1 \text{ iff } S \not\models \Phi_1$$
$$S \models \Phi_1 \wedge \Phi_2 \text{ iff both } S \models \Phi_1 \text{ and } S \models \Phi_2$$
$$S \models \Phi_1 \vee \Phi_2 \text{ iff one of } S \models \Phi_1 \text{ or } S \models \Phi_2$$
$$S \models (\forall x)\Phi_1 \text{ iff } S \models \Phi_1 \{x \mapsto \mathsf{t}\} \text{ for all ground terms } \mathsf{t} \text{ over } \Xi$$
$$S \models (\exists x)\Phi_1 \text{ iff } S \models \Phi_1 \{x \mapsto \mathsf{t}\} \text{ for some ground term } \mathsf{t} \text{ over } \Xi$$

Whether a fluent formula is satisfied by a given state will later be used to figure out if a conditional action effect occurs. To compute the changes to a state caused by action effects, we use the concept of state update known from the Fluent Calculus. For the purpose of this definition, direct effect laws with variables are viewed as representatives of their ground instances.

Definition 3.4. Consider a fixed UAC domain signature. For a state S and set L of fluent literals, define the *update of S with L* as $S + L \stackrel{\text{def}}{=} (S \setminus \overline{L}) \cup L$. Let Θ be an action domain specification and α be a ground term of sort action. The *resulting state of α in S* is

$$S + \{\psi \mid \underline{\texttt{action}}\ \alpha\ \underline{\texttt{causes}}\ \psi\ \underline{\texttt{if}}\ \Phi \in \Theta, S \models \Phi\} \tag{4}$$

So for a given state S and an action A applied in S, the resulting state is simply the state that contains all positive and negative effects whose conditions were satisfied in S and all literals that are not affected by the action. Note that $S + L$ is a state if and only if L is consistent.

Example 3.2 (Continued)**.** Let us apply the action $\mathsf{LiftBoth}$ to the previously depicted initial state $S = \{\mathsf{OnTable}, \neg\mathsf{LeftUp}, \neg\mathsf{RightUp}, \neg\mathsf{Spill}\}$ where the table is level and on the ground with the soup on top of it. State update determines the resulting state $S + \{\mathsf{LeftUp}, \mathsf{RightUp}\} = \{\mathsf{OnTable}, \mathsf{LeftUp}, \mathsf{RightUp}, \neg\mathsf{Spill}\}$.

For the remaining actions $\mathsf{LiftLeft}$ and $\mathsf{LiftRight}$, we could be tempted to specify their effects by, say,

$$\begin{aligned}
&\underline{\texttt{action}}\ \mathsf{LiftLeft}\ \underline{\texttt{causes}}\ \mathsf{LeftUp}\\
&\underline{\texttt{action}}\ \mathsf{LiftLeft}\ \underline{\texttt{causes}}\ \mathsf{Spill}\ \underline{\texttt{if}}\ \mathsf{OnTable} \wedge \neg\mathsf{RightUp}\\
&\underline{\texttt{action}}\ \mathsf{LiftLeft}\ \underline{\texttt{causes}}\ \neg\mathsf{OnTable}\ \underline{\texttt{if}}\ \mathsf{OnTable} \wedge \neg\mathsf{RightUp}
\end{aligned}$$

for $\mathsf{LiftLeft}$ and symmetrically for $\mathsf{LiftRight}$. But there are several issues with this specification:

Firstly, it makes unstated assumptions. The second and third law implicitly assume that the right side of the table not being up persists through lifting the left. But this cannot be guaranteed. An effect may occur at the same time which causes the left side to be up as well, leading the representation above to falsely predict a spill. What we want to express is that there is a spill if the two sides of the table are at different levels in the *resulting state*! This also relates to elaboration tolerance: if the specification is later changed, these laws might become incorrect and would have to be revisited. This is in contrast to the property of elaboration tolerance that we desire for our formalism, meaning that we incorporate new domain information by adding new laws.

Secondly, the specification above is not a faithful representation of causality. Although the left side of the table being up can be seen as a direct effect of lifting it, this hardly holds for the bowl falling off the table and subsequently producing a spill on the floor.

Thirdly, assume the chain of events possibly initiated by lifting the table does not stop there. What if a paper airplane happens to lie below the table and the spilled soup wets it, thereby impairing the plane's ability to fly? Surely, we cannot take into account all such absurd contingencies when specifying the supposed direct effects of lifting the table. It is particularly mentionable here that the cause of the paper airplane becoming wet is irrelevant to its subsequent not-flying, yet the effect would have to be duplicated for the effect specifications of $\mathsf{LiftLeft}$ and $\mathsf{LiftRight}$.

From these reflections we can see that formulating all possible changes using direct effect laws leads to cumbersome and at last unmanageably large action specifications, if it is not outright impossible. To deal with this representational aspect of the ramification problem, we introduce here a modular specification of indirect action effects. We employ expressions that specify certain conditions under which a change of truth value of one fluent causes a change of truth value of another fluent. These conditions do not hinge on execution of a specific action, but solely depend on change of a world property instead.

Definition 3.5. Consider a fixed UAC domain signature and let χ, ψ be fluent literals and

Φ_1, Φ_2 be fluent formulas. An *indirect effect law* is of the form

$$\underline{\texttt{effect}}\ \chi\ \underline{\texttt{causes}}\ \psi\ \underline{\texttt{if}}\ \Phi_1\ \underline{\texttt{before}}\ \Phi_2 \tag{5}$$

where χ is the *trigger*, ψ is the *effect*, Φ_1 is the *initial* and Φ_2 the *terminal context*. An indirect effect law is *open* if it contains variables, otherwise it is *closed*.

The intended reading of such an indirect effect law is "whenever Φ_1 holds in the starting state, Φ_2 holds in the resulting state and χ has turned from false to true during action execution, then ψ should be an indirect effect." If both contexts are $\top$, we omit them and simply write $\underline{\texttt{effect}}\ \chi\ \underline{\texttt{causes}}\ \psi$.

Example 3.2 (Continued)**.** In the soup-on-the-table domain, the effects propagate as follows. For causing a spill and causing the bowl to fall off the table, we have, respectively:

$$\underline{\texttt{effect}}\ \mathsf{LeftUp}\ \underline{\texttt{causes}}\ \mathsf{Spill}\ \underline{\texttt{if}}\ \mathsf{OnTable}\ \underline{\texttt{before}}\ \neg\mathsf{RightUp} \tag{6}$$

$$\underline{\texttt{effect}}\ \mathsf{RightUp}\ \underline{\texttt{causes}}\ \mathsf{Spill}\ \underline{\texttt{if}}\ \mathsf{OnTable}\ \underline{\texttt{before}}\ \neg\mathsf{LeftUp} \tag{7}$$

$$\underline{\texttt{effect}}\ \mathsf{LeftUp}\ \underline{\texttt{causes}}\ \neg\mathsf{OnTable}\ \underline{\texttt{if}}\ \top\ \underline{\texttt{before}}\ \neg\mathsf{RightUp} \tag{8}$$

$$\underline{\texttt{effect}}\ \mathsf{RightUp}\ \underline{\texttt{causes}}\ \neg\mathsf{OnTable}\ \underline{\texttt{if}}\ \top\ \underline{\texttt{before}}\ \neg\mathsf{LeftUp} \tag{9}$$

For the actions of lifting just one side of the table, it now suffices to specify their immediate, direct effects $\underline{\texttt{action}}\ \mathsf{LiftLeft}\ \underline{\texttt{causes}}\ \mathsf{LeftUp}$ and the symmetric $\underline{\texttt{action}}\ \mathsf{LiftRight}\ \underline{\texttt{causes}}\ \mathsf{RightUp}$, which completes the description of the domain Θ_{Spill}.

Although it might at first glance seem to be an overkill to have two contexts, the expressiveness gained through the distinction is crucial here and in general: the context of the bowl being on the table must be checked in the starting state, since it might become false during action execution (when the bowl falls off the table); the context of the other side of the table not being up must be checked in the resulting state, since it could become true during action execution (when both sides are lifted simultaneously). It is important to note that we do not care exactly *how* the change in the trigger or the terminal context was established. The potential user need not fear to have to specify triggers and terminal contexts manually by themselves – in Section 4.3 we provide a general method for specifying formula triggers, that is, effects that occur whenever some formula becomes true.

Using indirect effect laws, a knowledge engineer need only specify the immediate causal relationships of action domains in which complex causal chains may occur. The decision which causal relations are relevant for a problem at hand is part of the knowledge engineering process and cannot be detailed further in this paper, but we do mention that there have been attempts to automate the generation of causal laws from a given set of state constraints and with the help of information about causal directionality [38].

In an early attempt to overcome the restriction to direct effects from STRIPS-like systems, Wilkins [41] already proposed to use statements like (5), that he called *domain rules*. However, their semantics was only defined operationally by his implemented system. In the reasoning about actions community, on the other hand, researchers started out with simpler causation statements that however had a clear-cut meaning [25, 37, 29]. Here, we in a sense reunite these two lines of research by providing a declarative semantics for Wilkins' domain rules.

Using the notions of states and state update as above, we now formally define the meaning of indirect effect laws. For the following definition, we take open direct and indirect effect laws to represent the respective sets of their ground instances.

Definition 3.6. Let Θ be an action domain specification, α be a ground action and let S, T be states. Define

$$S_0^\alpha \stackrel{\text{def}}{=} \{\psi \mid \underline{\texttt{action}}\ \alpha\ \underline{\texttt{causes}}\ \psi\ \underline{\texttt{if}}\ \Phi \in \Theta, S \models \Phi\}$$

and for $i \geq 0$

$$S_{i+1}^\alpha \stackrel{\text{def}}{=} S_i^\alpha \cup \{\psi \mid \underline{\texttt{effect}}\ \chi\ \underline{\texttt{causes}}\ \psi\ \underline{\texttt{if}}\ \Phi_1\ \underline{\texttt{before}}\ \Phi_2 \in \Theta, S \models \Phi_1 \wedge \neg\chi, S_i^\alpha \models \chi, T \models \Phi_2\}^{3}$$

$$S_\infty^\alpha \stackrel{\text{def}}{=} \bigcup_{i=0}^{\infty} S_i^\alpha$$

T is called a *successor state of S for α* iff $T = S + S_\infty^\alpha$.

So in order to verify that a given state T is indeed a successor state, we must be able to reconstruct it in a well-founded way. First, we figure out all the direct action effects in S_0^α. We then repeatedly apply indirect effect laws to construct a set S_∞^α of literals that contains the direct and indirect atomic effects of the action. It only remains to check whether updating the starting state S with these effects does lead to T.

Example 3.2 (Continued)**.** Let us now apply the action $\mathsf{LiftLeft}$ to the previously depicted initial state $S = \{\mathsf{OnTable}, \neg\mathsf{LeftUp}, \neg\mathsf{RightUp}, \neg\mathsf{Spill}\}$ where the soup is on the table, that is level on the ground. We can by means of Definition 3.6 verify that the state where the bowl fell off the table and the soup has spilled, $T = \{\neg\mathsf{OnTable}, \mathsf{LeftUp}, \neg\mathsf{RightUp}, \mathsf{Spill}\}$, is a successor state of S for $\mathsf{LiftLeft}$: The direct effects are $S_0^{\mathsf{LiftLeft}} = \{\mathsf{LeftUp}\}$, which makes the indirect effect laws (6) and (8) applicable with respect to S. We get $S_1^{\mathsf{LiftLeft}} = S_0^{\mathsf{LiftLeft}} \cup \{\mathsf{Spill}, \neg\mathsf{OnTable}\}$. No more indirect effect laws are triggered through these effects, therefore $S_1^{\mathsf{LiftLeft}} = S_2^{\mathsf{LiftLeft}} = S_\infty^{\mathsf{LiftLeft}}$. Lastly, we verify that $S + S_\infty^{\mathsf{LiftLeft}} = T$: we have that $\{\mathsf{OnTable}, \neg\mathsf{LeftUp}, \neg\mathsf{RightUp}, \neg\mathsf{Spill}\} + \{\mathsf{LeftUp}, \mathsf{Spill}, \neg\mathsf{OnTable}\} = \{\neg\mathsf{OnTable}, \mathsf{LeftUp}, \neg\mathsf{RightUp}, \mathsf{Spill}\}$.

It is clear that the successor state of some S for an action α need not be unique; there may be zero or more successor states in general. This is useful – for example – to model the indeterminate effects of actions. Pinto [33, Section 4.1.3] mentions problems of his approach (that compiles state constraints into successor state axioms) whenever there are multiple minimal models. As our adaptation of his example shows, such a case is easily treated by our state transition semantics.

Example 3.7 ([33])**.** A person can put on a hat and also take it off, but does so only in the bedroom. After taking it off, the hat is either in the wardrobe or on the bed. Fluent OnHead says that the hat is on the head, $\mathsf{InWardrobe}$ and OnBed similarly express the hat's other possible locations. The direct effects of taking off the hat are formalised by $\underline{\texttt{action}}$ $\mathsf{TakeOff}$ $\underline{\texttt{causes}}$ $\neg\mathsf{OnHead}$. Now whenever the hat is caused not to be on the head any more (through whatever means), then either of the two possible new locations of the hat is justified:

$$r_w = \underline{\texttt{effect}}\ \neg\mathsf{OnHead}\ \underline{\texttt{causes}}\ \mathsf{InWardrobe}\ \underline{\texttt{if}}\ \top\ \underline{\texttt{before}}\ \neg\mathsf{OnBed}$$
$$r_b = \underline{\texttt{effect}}\ \neg\mathsf{OnHead}\ \underline{\texttt{causes}}\ \mathsf{OnBed}\ \underline{\texttt{if}}\ \top\ \underline{\texttt{before}}\ \neg\mathsf{InWardrobe}$$

For a state $S = \{\mathsf{OnHead}, \neg\mathsf{InWardrobe}, \neg\mathsf{OnBed}\}$ where the hat is on the head and nowhere else,

[3] Notice the occurrence of T in the definition of the S_i^α.

we can verify that both states

$$\begin{aligned} T_w &= \{\neg\mathsf{OnHead}, \mathsf{InWardrobe}, \neg\mathsf{OnBed}\} \\ T_b &= \{\neg\mathsf{OnHead}, \neg\mathsf{InWardrobe}, \mathsf{OnBed}\} \end{aligned}$$

are successor states of S for $\mathsf{TakeOff}$: We have

$$S_0^{\mathsf{TakeOff}} = \{\neg\mathsf{OnHead}\}$$

and with respect to state T_w, indirect effect law r_w is applicable since we have $S \models \top \wedge \neg\neg\mathsf{OnHead}$, $S_0^{\mathsf{TakeOff}} \models \neg\mathsf{OnHead}$ and $T_w \models \neg\mathsf{OnBed}$. This leads to the accumulated set of effects

$$S_1^{\mathsf{TakeOff}} = S_\infty^{\mathsf{TakeOff}} = \{\neg\mathsf{OnHead}, \mathsf{InWardrobe}\}$$

and $T_w = S + S_\infty^{\mathsf{TakeOff}}$ is easily checked. The verification of state T_b is analogous.

In the same way our approach can easily handle other domains that combine indeterminate with indirect consequences, as for example the Mailboxes scenario introduced by Castilho et al. [4], which features an action that triggers either or both of two possible direct effects along with a further, indirect effect.

4. An Axiomatic Solution to the Ramification Problem

We now present the general, first-order effect axiom that will solve the ramification problem using the UAC introduced in Section 2.1. This axiom appeals to the same axiomatisation technique as the effect axiom from Baumann et al. [2]. It formalises the idea of truth by causation: everything that is true must be caused, and vice versa. In the most simple form of the effect axiom, we allow two causes to determine a fluent's truth value: persistence and direct effects. Before introducing the axiom itself, we formalise the individual causes. For the first cause – persistence – we introduce a pair of macros expressing that a fluent f persists from s to t.

$$\mathit{FrameT}(f, s, t) \stackrel{\text{def}}{=} \mathit{Holds}(f, s) \wedge \mathit{Holds}(f, t) \tag{10}$$
$$\mathit{FrameF}(f, s, t) \stackrel{\text{def}}{=} \neg\mathit{Holds}(f, s) \wedge \neg\mathit{Holds}(f, t) \tag{11}$$

For the second cause – a fluent being a direct effect of an action – we introduce two new macros $\mathit{DirT}_\Theta(f, a, s, t)$ and $\mathit{DirF}_\Theta(f, a, s, t)$ expressing that a fluent f is a direct positive/negative effect of an action a from s to t. In contrast to the domain-independent macros FrameT, FrameF, the effect macros depend on a concrete domain Θ: when the direct (positive and negative) effects of an action are given by the direct effect laws of Θ, they can easily be translated into formulas that determine the truth values of all relevant DirT_Θ and DirF_Θ expressions. For a fluent formula Φ and a term τ : TIME, we denote by $\Phi[\tau]$ the first-order formula where each occurrence of a fluent literal $(\neg)\varphi$ has been replaced by $(\neg)\mathit{Holds}(\varphi, \tau)$.

Definition 4.1. Let Θ be an action domain specification and A be a function into sort ACTION with matching sequence of variables $\vec{x}$. The *direct positive and negative effect macros for* $A(\vec{x})$ are

$$\mathit{DirT}_\Theta(f, A(\vec{x}), s, t) \stackrel{\text{def}}{=} \bigvee_{\underline{\mathtt{action}}\ A(\vec{x})\ \underline{\mathtt{causes}}\ \varphi\ \underline{\mathtt{if}}\ \Phi \in \Theta} (f = \varphi \wedge \Phi[s]) \tag{12}$$
$$\mathit{DirF}_\Theta(f, A(\vec{x}), s, t) \stackrel{\text{def}}{=} \bigvee_{\underline{\mathtt{action}}\ A(\vec{x})\ \underline{\mathtt{causes}}\ \neg\varphi\ \underline{\mathtt{if}}\ \Phi \in \Theta} (f = \varphi \wedge \Phi[s]) \tag{13}$$

Now taking the two causes "persistence" and "direct effect" and putting them together yields the basic version of this section's effect axiom.

Definition 4.2. Let Θ be an action domain specification and A be a function into sort ACTION. An *effect axiom with unconditional effects and the frame assumption* is of the form

$$\begin{aligned} \mathit{Poss}(A(\vec{x}), s, t) \supset (\forall f)(\mathit{Holds}(f, t) \equiv \mathit{CausedT}(f, A(\vec{x}), s, t)) \wedge \\ (\forall f)(\neg \mathit{Holds}(f, t) \equiv \mathit{CausedF}(f, A(\vec{x}), s, t)) \end{aligned} \tag{14}$$

where

$$\mathit{CausedT}(f, A(\vec{x}), s, t) \stackrel{\text{def}}{=} \mathit{FrameT}(f, s, t) \vee \mathit{DirT}_\Theta(f, A(\vec{x}), s, t) \tag{15}$$

$$\mathit{CausedF}(f, A(\vec{x}), s, t) \stackrel{\text{def}}{=} \mathit{FrameF}(f, s, t) \vee \mathit{DirF}_\Theta(f, A(\vec{x}), s, t) \tag{16}$$

The macros *CausedT*, *CausedF* will be re-defined in the rest of this paper. When speaking about effect axiom (14), we will understand it retrofitted with their latest version.[4]

The design principle underlying our axiomatisation technique is that of causation: a fluent holds at a time point that is the end point of an action if and only if there is a cause for that; similarly, a fluent does not hold if and only if there is a cause for that, too.

4.1. Adding Indirect Effects

We now add a third cause, indirect effects, to the basic causes, persistence and direct effects. The idea is to express them as implications and take care that inferences in the contrapositive, non-causal direction are not possible. The macros $\mathit{IndT}_\Theta(f, s, t)$ and $\mathit{IndF}_\Theta(f, s, t)$ express that fluent f is an indirect (positive or negative, respectively) effect of an action occurring from s to t in domain Θ. For an indirect effect law $r = \underline{\texttt{effect}}\ \chi\ \underline{\texttt{causes}}\ \psi\ \underline{\texttt{if}}\ \Phi_1\ \underline{\texttt{before}}\ \Phi_2$, the indirect effect ψ materialises whenever the relationship has been *triggered*, that is, whenever the initial context Φ_1 holds at the starting time point s, the terminal context Φ_2 holds at the resulting time point t and the trigger χ has changed from untrue to true from s to t. As for direct effect laws, by the sign of an indirect effect law we refer to the sign of its effect, $\mathit{sign}(r) \stackrel{\text{def}}{=} \mathit{sign}(\psi)$. To access the effect literal of an indirect effect law r, we use $\mathit{Effect}(r) \stackrel{\text{def}}{=} \psi$ and do so similarly for the trigger: $\mathit{Trigger}(r) \stackrel{\text{def}}{=} \chi$.

Definition 4.3. Let $r(\vec{y}) = \underline{\texttt{effect}}\ \chi\ \underline{\texttt{causes}}\ \psi\ \underline{\texttt{if}}\ \Phi_1\ \underline{\texttt{before}}\ \Phi_2$ be an indirect effect law with free variables among $\vec{y}$, and let s, t : TIME be variables.

$$\mathit{Triggered}_{r(\vec{y})}(s, t) \stackrel{\text{def}}{=} \Phi_1[s] \wedge \Phi_2[t] \wedge \neg\chi[s] \wedge \chi[t] \tag{17}$$

Let Θ be a domain and f be a variable of sort FLUENT.

$$\mathit{IndT}_\Theta(f, s, t) \stackrel{\text{def}}{=} \bigvee_{r(\vec{y}) \in \Theta,\, \mathit{sign}(r) = +} (\exists \vec{y})(f = \mathit{Effect}(r(\vec{y})) \wedge \mathit{Triggered}_{r(\vec{y})}(s, t)) \tag{18}$$

$$\mathit{IndF}_\Theta(f, s, t) \stackrel{\text{def}}{=} \bigvee_{r(\vec{y}) \in \Theta,\, \mathit{sign}(r) = -} (\exists \vec{y})(\neg f = \mathit{Effect}(r(\vec{y})) \wedge \mathit{Triggered}_{r(\vec{y})}(s, t)) \tag{19}$$

[4] The attentive reader will have noticed that the syntax of axiom (14) does not quite correspond to (2). Simple syntactical manipulations can however be conducted to transform the effect axiom into a form that matches (2).

According to these macros, a fluent f is an indirect effect from s to t if there is a corresponding indirect effect law with effect f that triggered from s to t. The macros are straightforwardly integrated into the effect axiom as follows.

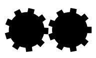
Definition 4.4. Let A be a function into sort ACTION. The *effect axiom* Υ_A *with conditional effects, the frame assumption and ramifications* is of the form (14), where

$$\mathit{CausedT}(f, A(\vec{x}), s, t) \stackrel{\text{def}}{=} \mathit{FrameT}(f, s, t) \lor \mathit{DirT}_\Theta(f, A(\vec{x}), s, t) \lor \mathit{IndT}_\Theta(f, s, t) \tag{20}$$
$$\mathit{CausedF}(f, A(\vec{x}), s, t) \stackrel{\text{def}}{=} \mathit{FrameF}(f, s, t) \lor \mathit{DirF}_\Theta(f, A(\vec{x}), s, t) \lor \mathit{IndF}_\Theta(f, s, t) \tag{21}$$

While the approach presented so far works well for simple ramification domains and easily copes with instantaneous effect propagation, it still harbours a serious flaw: it cannot handle cyclic fluent dependencies.

Example 2.3 (Continued)**.** If two gear wheels that can be separately turned and stopped are interlocked, then the causal relation between them is: whenever the first wheel is turned (resp. stopped), it causes the second one to turn (resp. stop), and vice versa:

$$\begin{array}{ll} \underline{\texttt{effect}}\ \mathsf{W}_1\ \underline{\texttt{causes}}\ \mathsf{W}_2 & \underline{\texttt{effect}}\ \neg\mathsf{W}_1\ \underline{\texttt{causes}}\ \neg\mathsf{W}_2 \\ \underline{\texttt{effect}}\ \mathsf{W}_2\ \underline{\texttt{causes}}\ \mathsf{W}_1 & \underline{\texttt{effect}}\ \neg\mathsf{W}_2\ \underline{\texttt{causes}}\ \neg\mathsf{W}_1 \end{array}$$



Let us compile the (positive half of the) effect axiom for a trivial Wait action that has no direct effects, so that Definitions 4.3 and 4.4 yield the following:

$$\begin{aligned} \mathit{CausedT}(f, \mathsf{Wait}, s, t) &\equiv \mathit{FrameT}(f, s, t) \lor \mathit{IndT}_\Theta(f, s, t) \\ \mathit{IndT}_\Theta(f, s, t) &= (f = \mathsf{W}_2 \land \neg \mathit{Holds}(\mathsf{W}_1, s) \land \mathit{Holds}(\mathsf{W}_1, t)) \lor \\ &\quad\ (f = \mathsf{W}_1 \land \neg \mathit{Holds}(\mathsf{W}_2, s) \land \mathit{Holds}(\mathsf{W}_2, t)) \end{aligned}$$

Consider the first-order structure $\mathfrak{M}$ with $\text{TIME}^{\mathfrak{M}} = \{\sigma, \tau\}$, $\sigma <^{\mathfrak{M}} \tau$, $\mathit{Poss}^{\mathfrak{M}} = \{(\mathsf{Wait}, \sigma, \tau)\}$ and $\mathit{Holds}^{\mathfrak{M}} = \{(\mathsf{W}_1, \tau), (\mathsf{W}_2, \tau)\}$. Together with the variable evaluation $\mathfrak{V} = \{s \mapsto \sigma, t \mapsto \tau\}$, interpretation $\mathfrak{I} = (\mathfrak{M}, \mathfrak{V})$ is a model for effect axiom (14) for Wait where both wheels initially stand still and magically start turning – one being the cause for the other and vice versa. This is undesired as Wait is intended to have no effect at all.

4.2. Loops and Loop Formulas

Much as in the case of Clark's completion of normal logic programs, our compilation of indirect effect laws into effect axioms allows too many models for fluents (that is, reified predicates) that cyclically depend on each other. We propose a solution to this problem in the spirit of loop formulas [28] for normal logic programs. In order for the approach to stay practical, we however have to restrict the syntax of the indirect effect laws in Θ:

1. For each indirect effect law $\underline{\texttt{effect}}\ \chi\ \underline{\texttt{causes}}\ \psi\ \underline{\texttt{if}}\ \Phi_1\ \underline{\texttt{before}}\ \Phi_2 \in \Theta$, we stipulate that $\mathit{Var}(\chi) \subseteq \mathit{Var}(\psi)$, that is, there may not be local variables in triggers.
2. We do not use function symbols with arity greater than zero as arguments of sort FLUENT.

These two constraints guarantee the existence of a finite, complete set of loops [5]. This set can be identified by a simple algorithm operating on open indirect effect laws, which makes our definition of effect axioms entirely constructive and easily automatable.

Throughout the following definitions, we will make explicit use of substitutions, unifiers and most general unifiers (*mgus*). Their domains and ranges are understood to be built from the domain signature used for specifying the indirect effect laws. For unification, negation is treated as a unary function. The definition of loops here follows the one of Chen et al. [5] given in Section 2.

Definition 4.5. Let Θ be an action domain specification over a signature Ξ.

- The *influence graph* G_Θ *of* Θ is the (possibly infinite) directed graph $G_\Theta \stackrel{\text{def}}{=} (V, E)$, where V is the set of all fluent literals over Ξ and for all $\underline{\texttt{effect}}\ \chi\ \underline{\texttt{causes}}\ \psi\ \underline{\texttt{if}}\ \Phi_1\ \underline{\texttt{before}}\ \Phi_2 \in \Theta$ and substitutions θ, there is an edge $(\chi\theta, \psi\theta) \in E$.
- A finite, nonempty set L of literals constitutes a *loop* iff for all $\mu, \nu \in L$, there is a directed, non-zero length path from μ to ν in the subgraph of G_Θ induced by L.
- An indirect effect law $r = \underline{\texttt{effect}}\ \chi\ \underline{\texttt{causes}}\ \psi\ \underline{\texttt{if}}\ \Phi_1\ \underline{\texttt{before}}\ \Phi_2 \in \Theta$ *leads into the loop* L iff
  1. there exists a $\mu \in L$ and a substitution $\theta_r^- = mgu(\psi, \mu)$ and
  2. for all substitutions θ' such that there exists a substitution θ'' with $\theta' = \theta_r^-\theta''$, we have $\chi\theta' \notin L\theta'$.

  Then $\Theta_L^- \stackrel{\text{def}}{=} \{r\theta_r^- \mid r \in \Theta \text{ and } r \text{ leads into the loop } L\}$.

Observe that the literals μ in a loop L share the same variables since none of the effect laws has any local variables. To simplify further constructions, we will assume without loss of generality that $\theta_r^- = mgu(\psi, \mu)$ for rules r leading into a loop L only mentions variables from L.

For example, the indirect effect laws of the gear wheel domain give rise to two loops, $L_1 = \{\mathsf{W}_1, \mathsf{W}_2\}$ and $L_2 = \{\neg\mathsf{W}_1, \neg\mathsf{W}_2\}$.

From the work of Chen et al. [5], we know that although there may be infinitely many loops in general, there is always a finite set $Loops(\Theta)$ that captures all of them.

Theorem 4.6. *Let Θ be a domain that satisfies the restrictions above. There exists a finite set $Loops(\Theta)$ such that every loop L of Θ is subsumed by some $L' \in Loops(\Theta)$.*

Proof. We define a logic program Λ_Θ such that there is a one-to-one correspondence between the loops of Θ and the loops of Λ_Θ. For each function symbol $F : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n \to \textsc{fluent}$, introduce two new predicates $P_F : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n$ and $P_{\neg F} : \mathfrak{s}_1 \times \cdots \times \mathfrak{s}_n$, and define

$$\Lambda_\Theta \stackrel{\text{def}}{=} \{P_\psi(\vec{x_2}) \leftarrow P_\chi(\vec{x_1}) \mid \underline{\texttt{effect}}\ \chi(\vec{x_1})\ \underline{\texttt{causes}}\ \psi(\vec{x_2})\ \underline{\texttt{if}}\ \Phi_1\ \underline{\texttt{before}}\ \Phi_2 \in \Theta\}$$

Correspondence of the loops follows straightforwardly from the definition of loops in Section 2 and Definition 4.5. By Lemma 2.8, there is a finite, complete set of loops for Λ_Θ. Due to the correspondence, its counterpart in Θ is complete for Θ. $\square$

Having defined the loops of a given domain and being sure they can be captured finitely, we can now proceed to define the corresponding loop formulas. The idea of loop formulas is to eliminate the models that arise due to “spontaneous” activation of loops for which no *external support* exists. In the case of logic programs, the external support that counts as “legal” cause for loop activation is a program rule leading into the loop. In our case, the direct effects of an action have to be taken into account as potential reasons for loop activation, too. A loop can also be activated by another loop – but then the union of the two is again a loop, so this case is implicitly catered for. This general form of direct and indirect loop activation is illustrated in Figure 1.

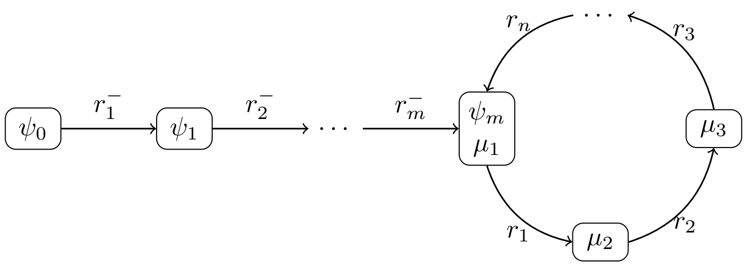


Figure 1: General form of loop activation. The leftmost node ψ_0 represents a direct action effect. The cycle on the right depicts a subset-maximal loop $L = \{\mu_1, \ldots, \mu_n\}$ of length n where $\psi_m = \mu_1$. For $m = 0$, we have the case of direct activation. Otherwise, for any natural m, there is a chain of m effects, and the loop is activated by the indirect effect law r_m^- leading into the loop L. Whenever r_m^- is itself part of a loop L', then $L \cup L'$ is again a loop and L is not subset-maximal.

When translating a logic program into a logical theory, following the approach of Chen et al. [5] the loop formulas are added to the predicate completion of the program.[5] In case of general effect axioms with their standard first-order semantics, loop formulas are "built into" the axioms, by enforcing the frame assumption for all fluent literals that could possibly change their truth value due to spontaneous loop activation. To achieve this for a given literal μ, we specify non-activation of all loops that could change μ as a sufficient cause for persistence of μ's truth value.

We use the notation $L(\vec{y})$ to explicitly refer to the free variables $\vec{y}$ mentioned in the loop L. Macro $\mathit{DirActivated}_{L(\vec{y})}(A(\vec{x}), s, t)$ expresses whether a loop $L(\vec{y})$ has been activated by action $A(\vec{x})$ from s to t. Naturally, this is the case whenever some positive literal $F(\vec{y}) \in L(\vec{y})$ from the loop is a direct positive effect of the action, or symmetrically some negative literal $\neg F(\vec{y}) \in L(\vec{y})$ from the loop is a direct negative effect of the action. To find out if a loop has been activated by an indirect effect, macro $\mathit{IndActivated}_{L(\vec{y})}(s, t)$ checks whether the corresponding instance of an indirect effect law leading into the loop has been triggered from s to t. Clearly, macro $\mathit{Activated}_{L(\vec{y})}(A(\vec{x}), s, t)$ then represents activation of the loop $L(\vec{y})$, directly or indirectly.

Definition 4.7. Let Θ be an action domain specification, $\mathit{Loops}(\Theta)$ be a finite, complete set of loops of Θ, $L(\vec{y}) \in \mathit{Loops}(\Theta)$, A be a function into sort ACTION and s, t be variables of sort TIME.

$$\begin{aligned}\mathit{DirActivated}_{L(\vec{y})}(A(\vec{x}), s, t) &\stackrel{\text{def}}{=} \bigvee_{F(\vec{y}) \in L(\vec{y})} \mathit{DirT}_\Theta(F(\vec{y}), A(\vec{x}), s, t) \vee \\ &\qquad \bigvee_{\neg F(\vec{y}) \in L(\vec{y})} \mathit{DirF}_\Theta(F(\vec{y}), A(\vec{x}), s, t) \qquad (22)\\ \mathit{IndActivated}_{L(\vec{y})}(s, t) &\stackrel{\text{def}}{=} \bigvee_{r(\vec{y}) \in \Theta_L^-} \mathit{Triggered}_{r(\vec{y})}(s, t) \qquad (23)\\ \mathit{Activated}_{L(\vec{y})}(A(\vec{x}), s, t) &\stackrel{\text{def}}{=} \mathit{DirActivated}_{L(\vec{y})}(A(\vec{x}), s, t) \vee \mathit{IndActivated}_{L(\vec{y})}(s, t) \qquad (24)\end{aligned}$$

[5]Meanwhile, there have been more general proposals where the loop formulas are added to the program as is, without forming the completion [10].

Let f : FLUENT be a variable.

$$\begin{aligned}
\mathit{LoopFrameT}(f, A(\vec{x}), s, t) &\overset{\text{def}}{=} \mathit{Holds}(f, s) \wedge \\
&\bigvee_{\substack{L(\vec{y})\in \mathit{Loops}(\Theta),\\ \neg F(\vec{y})\in L(\vec{y})}} (\exists \vec{y})\big(f = F(\vec{y}) \wedge \neg \mathit{Activated}_{L(\vec{y})}(A(\vec{x}), s, t)\big) \qquad (25)\\
\mathit{LoopFrameF}(f, A(\vec{x}), s, t) &\overset{\text{def}}{=} \neg\mathit{Holds}(f, s) \wedge \\
&\bigvee_{\substack{L(\vec{y})\in \mathit{Loops}(\Theta),\\ F(\vec{y})\in L(\vec{y})}} (\exists \vec{y})\big(f = F(\vec{y}) \wedge \neg \mathit{Activated}_{L(\vec{y})}(A(\vec{x}), s, t)\big) \qquad (26)
\end{aligned}$$

Macros (25) and (26) formalise the intuition that the truth value of a fluent should persist whenever there is some loop $L(\vec{y})$ that could change the truth value, but which has not been activated from s to t. If there indeed exists such a loop, then all of the loops containing the literal are activated whenever one of them is activated (cf. Figure 1). However, if for a fluent literal ψ there is *no* loop containing $\neg\psi$, then $\mathit{LoopFrame}(\psi, A(\vec{x}), s, t) = \psi[s] \wedge \bot$ is equivalent to false. This is justified since ψ may never spontaneously change, so the precaution provided by the loop formulas is unnecessary. The new causes are now added to the effect axiom as usual.

Definition 4.8. Let A be a function into sort ACTION. The *effect axiom* Υ_A *with conditional effects, the frame assumption and ramifications* is of the form (14), where

$$\begin{aligned}
\mathit{CausedT}(f, A(\vec{x}), s, t) &\overset{\text{def}}{=} \mathit{FrameT}(f, s, t) \vee \mathit{DirT}_\Theta(f, A(\vec{x}), s, t) \vee \\
&\quad \mathit{IndT}_\Theta(f, s, t) \vee \mathit{LoopFrameT}(f, A(\vec{x}), s, t) \qquad (27)\\
\mathit{CausedF}(f, A(\vec{x}), s, t) &\overset{\text{def}}{=} \mathit{FrameF}(f, s, t) \vee \mathit{DirF}_\Theta(f, A(\vec{x}), s, t) \vee \\
&\quad \mathit{IndF}_\Theta(f, s, t) \vee \mathit{LoopFrameF}(f, A(\vec{x}), s, t) \qquad (28)
\end{aligned}$$

Example 2.3 (Continued)**.** The Wait action has no direct effects and for neither of the loops L_1, L_2 exists an indirect effect law leading into the loop, hence $\mathit{Activated}_{L_1}(\mathsf{Wait}, s, t) = \bot$ and $\mathit{Activated}_{L_2}(\mathsf{Wait}, s, t) = \bot$. The new causes added to the effect axiom say that through Wait, the truth value of the loop literals must persist:

$$\begin{aligned}
\mathit{LoopFrameT}(f, \mathsf{Wait}, s, t) &\equiv \mathit{Holds}(f, s) \wedge (f = \mathsf{W}_1 \vee f = \mathsf{W}_2)\\
\mathit{LoopFrameF}(f, \mathsf{Wait}, s, t) &\equiv \neg\mathit{Holds}(f, s) \wedge (f = \mathsf{W}_1 \vee f = \mathsf{W}_2)
\end{aligned}$$

For the previously seen interpretation $\mathfrak{I}$ where the gear wheels start turning by magic, we now have $\mathfrak{I} \models \mathit{LoopFrameF}(\mathsf{W}_1, \mathsf{Wait}, \sigma, \tau)$ but $\mathfrak{I} \not\models \neg\mathit{Holds}(\mathsf{W}_1, \tau)$, and $\mathfrak{I}$ is no model for the effect axiom any more, just as desired.

This concludes the compilation of direct and indirect effect laws into first-order effect axioms. It remains to give the immediate definition of a domain axiomatisation for a domain with indirect effect laws.

Definition 4.9. Let Θ be an action domain specification. Its corresponding domain axiomatisation is $\Sigma = \Omega \cup \Pi \cup \Upsilon \cup \Sigma_{aux}$, where Ω defines the time structure, for each function into sort ACTION, Π contains a precondition axiom (1) and Υ contains an effect axiom according to Definition 4.8, and Σ_{aux} contains the unique-names axioms for sorts FLUENT and ACTION.

The precise definition of how to turn the precondition laws of a domain into precondition axioms (1) depends on the chosen time structure. We have left out the details since action preconditions are not the primary concern of this paper. The interested reader can consult the various methods presented by Thielscher [39] and Strass [36].

4.3. From Trigger Formulas to Trigger Literals

Up to here, we considered only triggers that are literals. The question arises whether this is a proper limitation, especially in light of the *derived effect rules* $\texttt{initiating}\ X\ \texttt{causes}\ \psi\ \texttt{if}\ \Phi$ of Denecker et al. [8], where ψ is a fluent literal and X and Φ can be general propositional fluent formulas.

It turns out that even more general derived effect rules – namely those where X is a first-order quantifier-free (implicitly universally quantified) formula – can be transformed into indirect effect laws by the procedure below. It takes as input a derived effect rule $\texttt{initiating}\ X\ \texttt{causes}\ \psi\ \texttt{if}\ \Phi$.

1. Transform X into its disjunctive normal form (DNF) $X = X_1 \vee \ldots \vee X_m$.[6] As an example, we look at the derived effect rule

   $$\texttt{initiating}\ \mathsf{LeftUp} \not\equiv \mathsf{RightUp}\ \texttt{causes}\ \mathsf{Spill}\ \texttt{if}\ \mathsf{OnTable}$$

   Transforming the trigger formula $\mathsf{LeftUp} \not\equiv \mathsf{RightUp}$ into disjunctive normal form yields $(\mathsf{LeftUp} \wedge \neg\mathsf{RightUp}) \vee (\neg\mathsf{LeftUp} \wedge \mathsf{RightUp})$.
2. Create the intermediate derived effect rules $\texttt{initiating}\ X_i\ \texttt{causes}\ \psi\ \texttt{if}\ \Phi$ for $1 \leq i \leq m$. Note that all trigger formulas are conjunctions of literals. In the example, this step produces

   $$\begin{array}{l}\texttt{initiating}\ \mathsf{LeftUp} \wedge \neg\mathsf{RightUp}\ \texttt{causes}\ \mathsf{Spill}\ \texttt{if}\ \mathsf{OnTable}\\ \texttt{initiating}\ \neg\mathsf{LeftUp} \wedge \mathsf{RightUp}\ \texttt{causes}\ \mathsf{Spill}\ \texttt{if}\ \mathsf{OnTable}\end{array}$$

3. For each $\texttt{initiating}\ \chi_1 \wedge \ldots \wedge \chi_n\ \texttt{causes}\ \psi\ \texttt{if}\ \Phi$, create for $i = 1, \ldots, n$ the indirect effect laws

   $$\underline{\texttt{effect}}\ \chi_i\ \underline{\texttt{causes}}\ \psi\ \underline{\texttt{if}}\ \Phi\ \underline{\texttt{before}} \bigwedge_{j=1,\ldots,n, j\neq i} \chi_j$$

   For the example, we get the indirect effect laws (6) and (7) we already know and the two new rules

   $$\begin{array}{l}\underline{\texttt{effect}}\ \neg\mathsf{RightUp}\ \underline{\texttt{causes}}\ \mathsf{Spill}\ \underline{\texttt{if}}\ \mathsf{OnTable}\ \underline{\texttt{before}}\ \mathsf{LeftUp}\\ \underline{\texttt{effect}}\ \neg\mathsf{LeftUp}\ \underline{\texttt{causes}}\ \mathsf{Spill}\ \underline{\texttt{if}}\ \mathsf{OnTable}\ \underline{\texttt{before}}\ \mathsf{RightUp}\end{array}$$

   for the case when both sides are up and only one is let down.

The intuition behind the procedure is straightforward. Assume the trigger formula X is activated from s to t, that is $\neg X[s]$ and $X[t]$. Then $\neg X_i[s]$ for all $1 \leq i \leq m$ and $X_j[t]$ for some $1 \leq j \leq m$. In particular $\neg X_j[s] \wedge X_j[t]$. Since $X_j = \chi_{j1} \wedge \ldots \wedge \chi_{jn}$ is a conjunction of literals, this in turn means that there is at least one χ_{jk} for $1 \leq k \leq n$ with $\neg\chi_{jk}[s] \wedge \chi_{jk}[t]$. The procedure now simply creates indirect effect laws for all possible χ_{jk}.

[6]This might involve a (worst-case) exponential blowup, which is however also inherently present for Denecker et al. [8], where all possible "supporting sets" have to be considered.

5. Correctness of the Axiomatic Solution

To show that the axioms just presented indeed capture the state-transition semantics given earlier, we first define how the two semantics link together. Below, we define how two time points of a first-order interpretation that are connected by an action application give rise to two states. This definition will be the basis of the correspondence result of this section. Recall that for a fluent literal $(\neg)\varphi$ and term $\tau : \textsc{time}$, the notation $(\neg)\varphi[\tau]$ abbreviates $(\neg)\mathit{Holds}(\varphi, \tau)$, a notation that generalises to fluent formulas by structural induction.

Definition 5.1. Let Ξ be a domain signature and $\mathfrak{I} = (\mathfrak{M}, \mathfrak{V})$ be an interpretation for Ξ. Let σ and τ be ground terms of sort $\textsc{time}$, α be a ground term of sort $\textsc{action}$ such that $(\alpha^{\mathfrak{M}}, \sigma^{\mathfrak{M}}, \tau^{\mathfrak{M}}) \in \mathit{Poss}^{\mathfrak{M}}$. Define two states S and T as follows: for a ground fluent literal ψ, set $\psi \in S$ if and only if $\mathfrak{I} \models \psi[\sigma]$, and $\psi \in T$ if and only if $\mathfrak{I} \models \psi[\tau]$.

Throughout the proof, we identify α, σ and τ with their interpretations under $\mathfrak{M}$. We then write $\Upsilon_\alpha[\sigma, \tau]$ to refer to effect axiom (14) for α where the $\textsc{time}$ variables s and t have been replaced by σ and τ, respectively. For predicates with an obvious polarity (like $\mathit{DirT}, \mathit{DirF}$), we use a neutral version (like Dir) with fluent literals ψ, where $\mathit{Dir}(\psi, a, s, t)$ denotes $\mathit{DirF}(F, a, s, t)$ if $\psi = \neg F$ for some fluent term F and $\mathit{DirT}(\psi, a, s, t)$ otherwise.

An important first observation concerning the proof is that by the definition of S and T we have for all fluent formulas Φ that $S \models \Phi$ iff $\mathfrak{I} \models \Phi[\sigma]$ and $T \models \Phi$ iff $\mathfrak{I} \models \Phi[\tau]$. An immediate consequence thereof is this:

Lemma 5.2. *For ground action α and direct effect law* `action` α `causes` ψ `if` Φ*, we have* $\psi \in S_0^\alpha$ *iff* $\mathfrak{I} \models \mathit{Dir}(\psi, \alpha, \sigma, \tau)$.

For the rest of the results, assume a fixed domain Θ over the signature Ξ. The next lemma is easy to prove via induction. It says that whenever $\mathfrak{I}$ is a model for α's effect axiom, the action effects S_∞^α have materialised in the resulting state T.

Lemma 5.3. $\mathfrak{I} \models \Upsilon_\alpha[\sigma, \tau]$ *implies* $S_\infty^\alpha \subseteq T$.

Proof. We use induction to show $S_i^\alpha \subseteq T$ for all $i \geq 0$.

$i = 0$. Let $\mu \in S_0^\alpha$. By Lemma 5.2, $\mathfrak{I} \models \mathit{Dir}(\mu, \alpha, \sigma, \tau)$. Together with $\mathfrak{I} \models \Upsilon_\alpha[\sigma, \tau]$, we have $\mathfrak{I} \models \mu[\tau]$. The definition of $\mathfrak{I}$ yields $\mu \in T$.

$i \to i + 1$. Let $\mu \in S_{i+1}^\alpha \setminus S_i^\alpha$. According to Definition 3.6, there is an indirect effect law $r =$ `effect` χ `causes` μ `if` Φ_1 `before` $\Phi_2 \in \Theta$ such that $S \models \Phi_1 \wedge \neg\chi$, $S_i^\alpha \models \chi$ and $T \models \Phi_2$. Applying the induction hypothesis to $\chi \in S_i^\alpha$ yields $\chi \in T$ and thus, by definition of $\mathfrak{I}$, we have $\mathfrak{I} \models \chi[\tau]$, $\mathfrak{I} \models \mathit{Triggered}_r(\sigma, \tau)$ and $\mathfrak{I} \models \mathit{Ind}(\mu, \sigma, \tau)$. By the presumption $\mathfrak{I} \models \Upsilon_\alpha[\sigma, \tau]$ we get $\mathfrak{I} \models \mu[\tau]$ and thus $\mu \in T$ by definition of $\mathfrak{I}$. □

Now any fluent literal of which the effect axiom says it is an indirect effect can be found in the set of action effects.

Lemma 5.4. *Let* $\mathfrak{I} \models \Upsilon_\alpha[\sigma, \tau]$. *For any ground fluent literal* μ, $\mathfrak{I} \models \mathit{Ind}(\mu, \sigma, \tau)$ *implies* $\mu \in S_\infty^\alpha$.

Proof. Let $\mathfrak{I} \models \Upsilon_\alpha[\sigma, \tau]$ and $\mathfrak{I} \models \mathit{Ind}(\mu, \sigma, \tau)$. Then there is an instance of an indirect effect law `effect` χ `causes` μ `if` Φ_1 `before` $\Phi_2 \in \Theta$ such that $\mathfrak{I} \models \Phi_1[\sigma] \wedge \neg\chi[\sigma]$ and $\mathfrak{I} \models \Phi_2[\tau] \wedge \chi[\tau]$. From $\mathfrak{I} \models \chi[\tau]$ by $\mathfrak{I} \models \Upsilon_\alpha[\sigma, \tau]$ we get $\mathfrak{I} \models \mathit{Caused}(\chi, \alpha, \sigma, \tau) \wedge \neg\mathit{Caused}(\neg\chi, \alpha, \sigma, \tau)$. Since additionally $\mathfrak{I} \models \neg\chi[\sigma]$ yields $\mathfrak{I} \models \neg\mathit{Frame}(\chi, \sigma, \tau)$ and $\mathfrak{I} \models \neg\mathit{LoopFrame}(\chi, \alpha, \sigma, \tau)$, we must have one of $\mathfrak{I} \models \mathit{Dir}(\chi, \alpha, \sigma, \tau)$ or $\mathfrak{I} \models \mathit{Ind}(\chi, \alpha, \sigma, \tau)$.

- $\mathfrak{I} \models \mathit{Dir}(\chi, \alpha, \sigma, \tau)$. Then Lemma 5.2 shows $\chi \in S_0^\alpha$ and Definition 3.6 yields $\mu \in S_\infty^\alpha$.
- $\mathfrak{I} \models \mathit{Ind}(\chi, \alpha, \sigma, \tau)$. By $\mathfrak{I} \models \neg \mathit{CausedT}(\neg\chi, \alpha, \sigma, \tau)$, we get $\mathfrak{I} \models \neg \mathit{LoopFrame}(\neg\chi, \alpha, \sigma, \tau)$, whose macro-expansion means $\mathfrak{I} \models \neg(\neg\chi[\sigma] \wedge \bigvee_{L \in \mathit{Loops}(\Theta)_{\chi \in L}} \neg \mathit{Activated}_L(\alpha, \sigma, \tau))$. Together with $\mathfrak{I} \models \neg\chi[\sigma]$, this yields $\mathfrak{I} \models \bigwedge_{L \in \mathit{Loops}(\Theta)_{\chi \in L}} \mathit{Activated}_L(\alpha, \sigma, \tau)$, that is, all loops containing χ have been activated.
  1. χ is not contained in any loop. Due to the syntactic restriction of FLUENTS' arguments, there is a finite sequence $r_1, \ldots, r_m$ of indirect effect laws such that $\mathfrak{I} \models \mathit{Dir}(\mathit{Trigger}(r_1), \alpha, \sigma, \tau)$, $\mathit{Effect}(r_i) = \mathit{Trigger}(r_{i+1})$ for $1 \leq i \leq m-1$ and $\mathit{Effect}(r_m) = \chi$. By Definition 3.6, we get $\chi \in S_\infty^\alpha$ and thus $\mu \in S_\infty^\alpha$.
  2. Let L be a maximal, ground loop containing χ such that $\mathfrak{I} \models \mathit{Activated}_L(\alpha, \sigma, \tau)$. We make a case distinction on the macro expansion of $\mathit{Activated}_L(\alpha, \sigma, \tau)$.
     (a) $\mathfrak{I} \models \mathit{DirActivated}_L(\alpha, \sigma, \tau)$. Then for some $\mu' \in L$ we have $\mathfrak{I} \models \mathit{Dir}(\mu', \alpha, \sigma, \tau)$ and therefore $\mu' \in S_0^\alpha$. By the definition of a loop, there is a sequence $r_1, \ldots, r_n$ of indirect effect laws such that $\mu' = \mathit{Trigger}(r_1)$ and $\mathit{Effect}(r_i) = \mathit{Trigger}(r_{i+1})$ for $1 \leq i \leq n-1$. Now $L = \{\mathit{Effect}(r_i) \mid 1 \leq i \leq n\}$ and Definition 3.6 yield $L \subseteq S_\infty^\alpha$, whence $\chi \in S_\infty^\alpha$ and $\mu \in S_\infty^\alpha$.
     (b) $\mathfrak{I} \models \mathit{IndActivated}_L(\alpha, \sigma, \tau)$. Then there is an indirect effect law $r \in \Theta_L^-$ leading into the loop with $\mathfrak{I} \models \mathit{Triggered}_r(\sigma, \tau)$. Since L is maximal, r is itself not part of another loop. We now have $\mathfrak{I} \models \mathit{Ind}(\mathit{Trigger}(r), \sigma, \tau)$ where $\mathit{Trigger}(r)$ is not part of a loop. Consequently, $\mathit{Effect}(r) \in S_\infty^\alpha$ and $L \subseteq S_\infty^\alpha$ can be argued for as above (item 1). □

With the two lemmata, it becomes straightforward to prove soundness of the axiomatisation. This is actually the "hard" direction of the overall proof, since inability to deal with cyclic dependencies results in unsound prediction of effects.

Theorem 5.5 (Soundness)**.** *$\mathfrak{I} \models \Upsilon_\alpha[\sigma, \tau]$ implies $T = S + S_\infty^\alpha$.*

Proof. Let $\mathfrak{I} \models \Upsilon_\alpha[\sigma, \tau]$.

"$\supseteq$": Let $\psi \in S + S_\infty^\alpha = (S \setminus \overline{S_\infty^\alpha}) \cup S_\infty^\alpha$.

1. $\psi \in S_\infty^\alpha = \bigcup_{i=0}^\infty S_i^\alpha$. Lemma 5.3 makes $\psi \in T$ immediate.
2. $\psi \in S \setminus \overline{S_\infty^\alpha}$, i.e. $\psi \in S$ and $\neg\psi \notin S_\infty^\alpha$. Then $\mathfrak{I} \models \psi[\sigma]$ and $\mathfrak{I} \models \neg \mathit{Frame}(\neg\psi, \sigma, \tau) \wedge \neg \mathit{LoopFrame}(\neg\psi, \sigma, \tau)$. By Lemma 5.2 and the presumption $\neg\psi \notin S_\infty^\alpha$, also $\mathfrak{I} \models \neg \mathit{Dir}(\neg\psi, \alpha, \sigma, \tau)$. By Lemma 5.4, $\neg\psi \notin S_\infty^\alpha$ implies $\mathfrak{I} \not\models \mathit{Ind}(\neg\psi, \sigma, \tau)$. Hence $\mathfrak{I} \models \neg \mathit{Caused}(\neg\psi, \alpha, \sigma, \tau)$, from which – by $\mathfrak{I} \models \Upsilon_\alpha[\sigma, \tau]$ – we can conclude $\mathfrak{I} \models \psi[\tau]$ and thus $\psi \in T$.

"$\subseteq$": Let $\psi \in T$. The definition of $\mathfrak{I}$ tells us $\mathfrak{I} \models \psi[\tau]$; $\mathfrak{I} \models \Upsilon_\alpha[\sigma, \tau]$ tells us $\mathfrak{I} \models \mathit{Caused}(\psi, \alpha, \sigma, \tau) \wedge \neg \mathit{Caused}(\neg\psi, \alpha, \sigma, \tau)$. Macro-expansion of *Caused* yields that $\mathfrak{I} \models \mathit{Dir}(\psi, \alpha, \sigma, \tau)$ or $\mathfrak{I} \models \mathit{Ind}(\psi, \sigma, \tau)$ or $\mathfrak{I} \models \mathit{Frame}(\psi, \sigma, \tau)$ or $\mathfrak{I} \models \mathit{LoopFrame}(\psi, \alpha, \sigma, \tau)$.

1. $\mathfrak{I} \models \mathit{Dir}(\psi, \alpha, \sigma, \tau)$. Lemma 5.2 immediately yields $\psi \in S_0^\alpha \subseteq S + S_\infty^\alpha$.
2. $\mathfrak{I} \models \mathit{Ind}(\psi, \sigma, \tau)$. Using Lemma 5.4, $\psi \in S_\infty^\alpha \subseteq S + S_\infty^\alpha$ is immediate.
3. $\mathfrak{I} \models \mathit{Frame}(\psi, \sigma, \tau)$ or $\mathfrak{I} \models \mathit{LoopFrame}(\psi, \alpha, \sigma, \tau)$. Then $\mathfrak{I} \models \psi[\sigma]$ and $\psi \in S$. Now $\psi \in T$ by T being a state implies $\neg\psi \notin T$, which by Lemma 5.3 yields $\neg\psi \notin S_\infty^\alpha$. Hence $\psi \notin \overline{S_\infty^\alpha}$ and $\psi \in S + S_\infty^\alpha$. □

For the converse direction, the first lemma says that whenever T is a successor state of S for α, then the effect axiom correctly establishes all indirect effects.

Lemma 5.6. *$T = S + S^\alpha_\infty$ and $\mathfrak{I} \models \mathit{Ind}(\mu, \sigma, \tau)$ imply $\mathfrak{I} \models \mu[\tau]$.*

Proof. Let $T = S + S^\alpha_\infty$ and $\mathfrak{I} \models \mathit{Ind}(\mu, \sigma, \tau)$. Then there exists a ground instance of an indirect effect law $r = \underline{\texttt{effect}}\ \chi\ \underline{\texttt{causes}}\ \mu\ \underline{\texttt{if}}\ \Phi_1\ \underline{\texttt{before}}\ \Phi_2$ such that $\mathfrak{I} \models \Phi_1[\sigma] \wedge \Phi_2[\tau] \wedge \neg\chi[\sigma] \wedge \chi[\tau]$. By definition of $\mathfrak{I}$, $S \models \Phi_1 \wedge \neg\chi$ and $T \models \Phi_2 \wedge \chi$. Since $\chi \notin S$, $\chi \in T$ and $T = S + S^\alpha_\infty$, there must be an $i \geq 0$ such that $\chi \in S^\alpha_i$. By Definition 3.6, $\mu \in S^\alpha_{i+1} \subseteq S^\alpha_\infty \subseteq T$; thus $\mathfrak{I} \models \mu[\tau]$. $\square$

Indeed, whenever a fluent literal is found to hold at the resulting time point and it was part of the update, then the effect axiom says it was an effect of the action.

Lemma 5.7. *Let $T = S + S^\alpha_\infty$ and $\mathfrak{I} \models \mu[\tau]$. Then $\mu \in S^\alpha_\infty$ implies $\mathfrak{I} \models \mathit{Dir}(\mu, \alpha, \sigma, \tau) \vee \mathit{Ind}(\mu, \sigma, \tau)$.*

Proof. Let $\mu \in S^\alpha_\infty$ and consider the smallest i such that $\mu \in S^\alpha_i$.

1. $i = 0$, that is, $\mu \in S^\alpha_0$. Lemma 5.2 immediately yields $\mathfrak{I} \models \mathit{Dir}(\mu, \alpha, \sigma, \tau)$.
2. $i > 0$. By Definition 3.6, there is an indirect effect law $r = \underline{\texttt{effect}}\ \chi\ \underline{\texttt{causes}}\ \mu\ \underline{\texttt{if}}\ \Phi_1\ \underline{\texttt{before}}\ \Phi_2$ such that $S \models \Phi_1 \wedge \neg\chi$, $S^\alpha_{i-1} \models \chi$ and $T \models \Phi_2$. Now $S^\alpha_{i-1} \subseteq T$ implies $T \models \chi$, hence $\mathfrak{I} \models \Phi_1[\sigma] \wedge \Phi_2[\tau] \wedge \neg\chi[\sigma] \wedge \chi[\tau]$. That means we have $\mathfrak{I} \models \mathit{Triggered}_r(\sigma, \tau)$ and hence $\mathfrak{I} \models \mathit{Ind}(\mu, \sigma, \tau)$. $\square$

The completeness result below now says that whenever T is a successor state of S for α, the corresponding interpretation $\mathfrak{I}$ is a model for the respective instance of the effect axiom.

Theorem 5.8 (Completeness)**.** *$T = S + S^\alpha_\infty$ implies $\mathfrak{I} \models \Upsilon_\alpha[\sigma, \tau]$.*

Proof. Let $T = S + S^\alpha_\infty$. We show $\mathfrak{I} \models \Upsilon_\alpha[\sigma, \tau]$ by showing $\mathfrak{I} \models \psi[\tau]$ iff $\mathfrak{I} \models \mathit{Caused}(\psi, \alpha, \sigma, \tau)$.

"if": Let $\mathfrak{I} \models \mathit{Caused}(\psi, \alpha, \sigma, \tau)$. We make a case distinction according to the macro-expansion of $\mathit{Caused}(\psi, \alpha, \sigma, \tau)$.

1. $\mathfrak{I} \models \mathit{Dir}(\psi, \alpha, \sigma, \tau)$. Lemma 5.2 yields $\psi \in S^\alpha_0 \subseteq S + S^\alpha_\infty$. The presumption $S + S^\alpha_\infty = T$ now shows $\mathfrak{I} \models \psi[\tau]$.
2. $\mathfrak{I} \models \mathit{Ind}(\psi, \sigma, \tau)$. By Lemma 5.6, we have $\mathfrak{I} \models \psi[\tau]$.
3. $\mathfrak{I} \models \mathit{Frame}(\psi, \sigma, \tau)$. Then $\mathfrak{I} \models \psi[\tau]$ is immediate.
4. $\mathfrak{I} \models \mathit{LoopFrame}(\psi, \alpha, \sigma, \tau)$. This entails $\mathfrak{I} \models \psi[\sigma]$ and thus $\psi \in S$. Furthermore, $\mathfrak{I} \models \bigvee_{L \in \mathit{Loops}(\Theta), \neg\psi \in L} \neg\mathit{Activated}_L(\alpha, \sigma, \tau)$, that is, none of the loops involving $\neg\psi$ has been activated, whereby we can conclude $\neg\psi \notin S^\alpha_\infty$. Now $\psi \in T = S + S^\alpha_\infty$ yields the desired $\mathfrak{I} \models \psi[\tau]$.

"only if": Let $\mathfrak{I} \models \psi[\tau]$. We have to show $\mathfrak{I} \models \mathit{Caused}(\psi, \alpha, \sigma, \tau)$. The definition of $\mathfrak{I}$ yields $\psi \in T$. By the presumption, we know $\psi \in S + S^\alpha_\infty = (S \setminus \overline{S^\alpha_\infty}) \cup S^\alpha_\infty$.

1. $\psi \in S^\alpha_\infty$. By Lemma 5.7, $\mathfrak{I} \models \mathit{Caused}(\psi, \alpha, \sigma, \tau)$.
2. $\psi \in (S \setminus \overline{S^\alpha_\infty})$, that is, $\psi \in S$ and $\neg\psi \notin S^\alpha_\infty$. From $\psi \in S$ and $\psi \in T$ we directly get $\mathfrak{I} \models \mathit{Frame}(\psi, \sigma, \tau)$ and thus $\mathfrak{I} \models \mathit{Caused}(\psi, \alpha, \sigma, \tau)$. $\square$

6. Related Work

Early treatments of ramifications all assumed the behaviour of the world to be specified by state constraints – static laws that must hold in all states of the world – and relied on the principle of minimal change [16, 42]. Brewka and Hertzberg [3] addressed the shortcomings of these approaches and provided an alternative formalisation which incorporated an explicit notion of causation. Lin [25], McCain and Turner [29] and Thielscher [37] independently made the important observation that mere state constraints are insufficient for a proper treatment of ramifications, and likewise introduced explicit representations of causality to deal with indirect effects. Thielscher [38] then showed how information on causal influence among fluents can be used to create a set of causal relationships from a set of domain constraints and provided a solution to the ramification problem, albeit restricted to a specific formalism.

6.1. Situation Calculus

Important pioneering work on ramification in the situation calculus was done by Lin [25]. We will discuss his approach and a recent extension of it in more detail below (Section 6.2).

In another approach basically similar to ours, Pinto [33] compiled ramification constraints into effect axioms by preprocessing. Although he is able to deal with cyclic dependencies, his approach did not incorporate an explicit notion of causality.

A comprehensive treatment of indirect effects was given by McIlraith [31]. There, causality is represented implicitly in the syntax of ramification constraints. The approach resorts to a minimal-model policy and requires stratification of the ramification constraints, hence it cannot deal with cyclic fluent dependencies.

6.2. Lin's Causal Action Theories

Recently, the causal action theories of Lin [25] have been extended to the case of cyclic dependencies among fluents [27]. The approach is based on a version of the Situation Calculus where action domains are characterised by precondition axioms and direct effect statements much like our direct effect laws. Indirect effects are specified by causal rules, formulas $\Phi[t] \supset \mathit{Caused}(\psi, s, t)$ meaning that literal ψ is caused to be true whenever state formula $\Phi[t]$ is true.[7] Lin and Soutchanski have given a minimisation-based approach to compile such domain specifications into successor state axioms which works even if there are cyclic fluent dependencies among the causal rules.

The simplicity of causal rules as implications is appealing from a designer's point of view, but the obvious drawback of this simplicity is an increased modelling complexity whenever additional expressiveness is needed. For example, it is not immediately clear how to treat Example 3.2 about the bowl of soup falling off the table, for the soup spills only when it was on the table before unevenly lifting one side.

Causal rules $X(t) \supset \mathit{Caused}(\psi, s, t)$ where $X(t)$ is quantifier-free can be viewed as derived effect rules `initiating` X `causes` ψ `if` $\top$ of Denecker et al. [8], which allows to apply the procedure of Section 4.3 to produce an equivalent set of indirect effect laws (5). In fact, we can even add an arbitrary first-order formula $\Phi(s)$ as initial context, thus leading to a general causal rule

$$(\Phi(s) \wedge X(t)) \supset \mathit{Caused}(\psi, s, t)$$

[7]Actually, Lin and Soutchanski [27] use a different notation where the truth value is reified, but the two notations are interchangeable.

which can be written as a derived effect rule [8]

$$\texttt{initiating } X \texttt{ causes } \psi \texttt{ if } \Phi$$

To this derived effect rule, we can now apply the procedure of Section 4.3.

On the other hand, the causal rules of Lin and Soutchanski [27] allow for existential quantification in the trigger formula, which is not easily reproduced in our approach.

6.3. Temporal Action Logics

Gustafsson and Doherty [17] present an approach to ramification in temporal action logic (TAL) [9]. They use "causal constraints" (which are directed implications) to represent indirect effects in linear-time domains. For the duration of the action, they use occlusions to exclude fluents from the frame assumption. At the end point of the action, state constraints determine the indirect effects. The semantics of TAL is defined through first-order axiomatisation with circumscription, hence cycles cannot be treated out of the box. Later, Kvarnström and Doherty [23] extend TAL by actions with duration, concurrent actions and nondeterministic actions. The circumscription policy they use is still reducible to the first-order case, so cycles remain a problem.

6.4. Action Language $\mathcal{ME}$

Kakas and Miller [19] use "r-propositions" L **whenever** C where L is a propositional fluent literal and C a set (representing a conjunction) of propositional fluent literals. These r-propositions have both a static aspect – the domain constraint $C \supset L$ – and a dynamic aspect – making all of C true causes L to become true. R-propositions are directed in that the converse $\neg L \supset \neg C$ need not be the case.

In comparison to our work, we note that C mixes context and trigger. In general, r-propositions can be translated into our indirect effect laws, but the converse is not always possible. Additionally, Kakas and Miller [19] present no first-order axiomatisation of the semantics of r-propositions.

Later, Kakas et al. [20][8] (although mainly focusing on the qualification problem) extend r-propositions to "c-propositions" C **causes** L, yet staying within a propositional language. There, C is a set of action and fluent literals denoting action (non-)occurrences and fluents (not) holding at the starting state. L is a fluent literal, the effect of a c-proposition, which notably holds only *provisionally*. That is, the potential effect L is qualified globally by other effects and observations, and competes with other potential effects. To define the semantics of c-propositions, Kakas et al. present a sophisticated fixpoint construction with intermediate states. Their approach can deal with cycles, but they give no logical axiomatisation.

In their semantical solution to the ramification problem they allow that in between two successive observable time points, there may be further, non-observable intermediate states which contribute to the computation of indirect effects. In Example 9 [20], there is a faulty circuit which can be switched on. After doing this, an electric current will flow. This current will then break the fuse in the circuit which in turn terminates the current. The important observation of this example is that "electric current" is neither true at the starting time point (because the switch is off) nor at the ending time point (since the fuse is broken). But the electric current is vital for deriving the indirect effect that the fuse breaks. In our approach, all intermediate effects become full-blown effects and the example is not immediately reproduced.

[8]There is also a journal version of that paper [21].

6.5. *Inductive Definitions*

Denecker et al. [8] and Van Belleghem et al. [40] show how the principle of inductive definition provides a way of modelling indirect effects that is independent of a specific calculus. The approach can treat cycles via the constructive nature of inductive definitions. However, they use a three-valued semantics and view the ramification problem in isolation from its close relative, the frame problem. It is therefore not immediately clear how the approach is to be embedded into general-purpose formalisms (where both frame and ramification problems must be solved) and how classical logical reasoning methods can be used in that approach. Denecker and Ternovska [7] later addressed these issues and extended Reiter's Situation Calculus [34] with inductive definitions. This extension allows to model ramifications (with cyclic dependencies) in a compact, modular and elaboration-tolerant fashion. The price to pay is however an increased complexity of the underlying mathematical machinery: the semantics of the logic FO(ID) they use (first-order logic extended with inductive definitions, where the latter are employed to model ramifications) is an extension of the semantics of classical logic with the well-founded semantics of logic programming [7].

In some sense, the work of Denecker and Ternovska [7] can be seen as dual to ours. The starting point of both is the realisation that classical first-order logic is not particularly well-suited to handle the full extent of the ramification problem due to FOL's inability to express transitive closure. There seem to be two ways to deal with this problem: extending the language and restricting the problem. Denecker and Ternovska [7] looked for a language that extends FOL and can treat the full ramification problem. Their result is that it suffices to extend FOL by inductive definitions. In this paper, we looked for a fragment of the ramification problem that can still be treated in classical first-order logic. Our result is that – in addition to FOL's ability to treat stratified ramification rules, which was previously known [25, 31] – a limited form of cyclic effects can be correctly handled within first-order logic.

6.6. *Event Calculus*

Shanahan [35] showed how to handle a particular class of ramifications in the Event Calculus [22], limited however in that the approach admittedly could not treat self-justifying cycles. In Chapter 6 of his event calculus book, Mueller [32] elaborates on various methods for dealing with the ramification problem. He does not discuss cycles explicitly, but his usage of circumscription suggests that cyclic dependencies pose a problem. Forth and Miller [12] proposed a treatment of indirect effects in the Event Calculus. The approach offers a logical axiomatisation that can deal with cyclic causal dependencies due to its use of nested prioritised circumscription. However, it is bound to linear time and a narrative-based semantics.

6.7. *Default Logic-based Ramifications of Baumann et al. [2]*

Baumann et al. [2] provided a treatment of ramifications using default logic and the unifying action calculus. Since it is close to the approach presented in this paper, we review it here in some more detail. There, indirect effect laws $\underline{\texttt{effect}}\ \chi\ \underline{\texttt{causes}}\ \psi\ \underline{\texttt{if}}\ \Phi_1\ \underline{\texttt{before}}\ \Phi_2$ are translated into justification-free Reiter defaults[9]

$$\frac{\Phi_1[s] \land \Phi_2[t] \land \neg\chi[s] \land \chi[t] :}{\mathit{Indirect}(\psi, s, t)} \tag{29}$$

[9]Actually, Baumann et al. [2] have made no distinction between initial and final context. It is however trivial to add this.

This solution is very elegant in that there are no syntactic restrictions on indirect effect laws, the laws can be modularly translated, and the groundedness of default extensions easily guarantees a proper treatment of causality for indirect effects. On the other hand, the translation of Baumann et al. [2] produces a default theory, where the translation in this paper produces a classical first-order theory. Despite being computationally more complex in the propositional case, the default logic approach is not always strong enough for domains with incompletely known states:

Example 6.1 (Ramification and Casualty)**.** Consider the Yale shooting domain with the single action Shoot, whose effect is that shooting a loaded gun leads to a casualty, a dead turkey: $\underline{\texttt{action}}\ \mathsf{Shoot}\ \underline{\texttt{causes}}\ \neg\mathsf{Alive}\ \underline{\texttt{if}}\ \mathsf{Loaded}$. The indirect effect law $r = \underline{\texttt{effect}}\ \neg\mathsf{Alive}\ \underline{\texttt{causes}}\ \neg\mathsf{Walking}$ states the ramification that whenever the turkey is caused to be not alive, it is caused to be not walking any more. We could decide to represent r by the two seemingly equivalent laws

$$r_1 = \underline{\texttt{effect}}\ \neg\mathsf{Alive}\ \underline{\texttt{causes}}\ \neg\mathsf{Walking}\ \underline{\texttt{if}}\ \mathsf{Walking}\ \underline{\texttt{before}}\ \top$$
$$r_2 = \underline{\texttt{effect}}\ \neg\mathsf{Alive}\ \underline{\texttt{causes}}\ \neg\mathsf{Walking}\ \underline{\texttt{if}}\ \neg\mathsf{Walking}\ \underline{\texttt{before}}\ \top$$

that say that getting killed causes the turkey not to be walking any more if it was walking before (r_1), and that the same happens if it was not walking (r_2). In the approach of this paper, for all pairs of states S, T with $S \models \mathsf{Alive}$ and $T \models \neg\mathsf{Alive}$, we have that r is applicable and exactly one of r_1 or r_2 is applicable. Thus, even if there is uncertainty about $\mathsf{Walking}$ on theory level, the indirect effect $\neg\mathsf{Walking}$ occurs. Translating these laws into the Reiter defaults

$$\delta_r = \frac{\mathit{Holds}(\mathsf{Alive}, s) \wedge \neg\mathit{Holds}(\mathsf{Alive}, t) :}{\mathit{IndirectF}(\mathsf{Walking}, s, t)}$$
$$\delta_1 = \frac{\mathit{Holds}(\mathsf{Alive}, s) \wedge \neg\mathit{Holds}(\mathsf{Alive}, t) \wedge \mathit{Holds}(\mathsf{Walking}, s) :}{\mathit{IndirectF}(\mathsf{Walking}, s, t)}$$
$$\delta_2 = \frac{\mathit{Holds}(\mathsf{Alive}, s) \wedge \neg\mathit{Holds}(\mathsf{Alive}, t) \wedge \neg\mathit{Holds}(\mathsf{Walking}, s) :}{\mathit{IndirectF}(\mathsf{Walking}, s, t)}$$

reveals the difference between the approaches. While loop formulas treat $\{r\}$ and $\{r_1, r_2\}$ equivalently, the ramification defaults behave in a different way when the truth of $\mathsf{Walking}$ at s is unknown: δ_r always yields the indirect effect $\neg\mathit{Holds}(\mathsf{Walking}, t)$ whenever Alive changes to true from s to t; the defaults δ_1 and δ_2 do not yield the indirect effect since they depend on the underlying theory additionally entailing either $\mathit{Holds}(\mathsf{Walking}, s)$ or $\neg\mathit{Holds}(\mathsf{Walking}, s)$.

Baumann et al. [2] also showed that the default logic-based approach to the ramification problem can be straightforwardly combined with action default theories. Alas, since normal and justification-free defaults are mixed, extension existence cannot be guaranteed for that approach. With our approach based on first-order logic, ramifications can be added to the default theories of Baumann et al. [2] without disrupting extension existence.

6.8. Further Approaches

Foo and Zhang [11] were the first to address the ramification problem in propositional dynamic logic, which for this purpose they extended by causal laws [43]. Their solution to the combined frame and ramification problems differs from ours in that they automatically generate disjunctive frame axioms by need when answering a query [44]. More recently, Herzig and Varzinczak [18] dealt with indirect effects through static laws based on their own variant of propositional dynamic logic. Baader et al. [1] use indirect effect laws that can deal with cycles in the restricted language of description logics. Their ramifications are triggered by (sets of) literals and guarded by an initial context only. Gelfond and Lifschitz [15] compare the expressiveness of action languages

$\mathcal{B}$ and $\mathcal{C}$ [14]. While ramification is not their primary concern, their main result is relevant to the study of cyclic dependencies of indirect effects in action theories. They show that there is a sublanguage of $\mathcal{B}$ that can be faithfully and modularly mapped into $\mathcal{C}$ given that its static laws are free of cycles. This is a consequence of $\mathcal{B}$ and $\mathcal{C}$'s differing intuitions regarding causality. For ramification, this means that $\mathcal{B}$ has a built-in treatment of cyclic dependencies while $\mathcal{C}$'s treatment of indirect effects (via static laws) resembles the predicate completion of Lin [25].

7. Conclusion

We presented a solution to the ramification problem that unifies and generalises existing approaches in that it (1) is independent of a particular time structure, (2) is based on first-order effect axioms and (3) deals with cyclic causal dependencies among fluents. Moreover, we have formally proved our axiomatisation correct for the presented state transition semantics.

In the comparison to related work, we have seen that the ramification problem (quite like many of the fundamental problems in reasoning about actions) has numerous aspects. Approaches to ramification can be categorised along the dimensions of the expressivity of their specification language (propositional, first-order, first-order with add-ons, second-order, ...), the definition of the semantics of that language (state-transition semantics, translation into logic or some other previously existing language) and the range of reasoning about actions phenomena it can treat (instantaneous change, cycles, race conditions and conflicts, ...).

In this paper we focused on giving an axiomatic first-order solution that can treat cycles, and so there remains potential for future work. A common problem among axiomatic solutions to the ramification problem is (not) coping with conflicting effects. For example, McIlraith [31] makes the explicit consistency assumption that direct and indirect effects may not make a fluent both true and false at the same time. (This is a generalisation of Reiter's consistency assumption for direct effects.) We do not make such an assumption explicitly, in our case the axiomatisation simply becomes inconsistent in case of a conflict between (in)direct effects. This is reminiscent of logic programming, where odd-length negative cycles among rules (like $a \leftarrow \mathit{not}\ a$) also lead to the absence of (two-valued) models. It could be worthwhile to use ideas from (three-valued) logic programming to generalise the approach from this paper.

Acknowledgements. The authors are grateful to Fangzhen Lin and several anonymous reviewers for constructive criticisms on earlier versions of this paper. The second author is the recipient of an ARC Future Fellowship (project number FT 0991348). He is also affiliated with the University of Western Sydney.

References

[1] Franz Baader, Marcel Lippmann, and Hongkai Liu. Using causal relationships to deal with the ramification problem in action formalisms based on description logics. In C. Fermüller and A. Voronkov, editors, *Proceedings of the International Conference on Logic Programming and Automated Reasoning (LPAR)*, volume 6397 of *LNCS*, pages 82–96, Yogyakarta, Indonesia, October 2010. Springer.

[2] Ringo Baumann, Gerhard Brewka, Hannes Strass, Michael Thielscher, and Vadim Zaslawski. State Defaults and Ramifications in the Unifying Action Calculus. In *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning*, pages 435–444, Toronto, Canada, May 2010.

[3] 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.

[4] Marcos Castilho, Andreas Herzig, and Ivan Varzinczak. It depends on the context! A decidable logic of actions and plans based on a ternary dependence relation. In *Proceedings of the 9th International Workshop on Non-Monotonic Reasoning (NMR)*, pages 343–348, Toulouse, France, April 2002.

[5] Yin Chen, Fangzhen Lin, Yisong Wang, and Mingyi Zhang. First-Order Loop Formulas for Normal Logic Programs. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 298–307. AAAI Press, June 2006.

[6] Keith L. Clark. Negation as Failure. In Hervé Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, 1978.

[7] Marc Denecker and Eugenia Ternovska. Inductive Situation Calculus. *Artificial Intelligence*, 171(5–6): 332–360, 2007.

[8] Marc Denecker, D. Theseider-Dupré, and Kristof Van Belleghem. An Inductive Definition Approach to Ramifications. *Linköping Electronic Articles in Computer and Information Science*, 3(7):1–43, January 1998.

[9] Patrick Doherty and Jonas Kvarnström. Temporal action logics. In *Handbook of Knowledge Representation*, number 3 in Foundations of Artificial Intelligence, pages 709–757. Elsevier, 2008.

[10] Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. A Generalization of the Lin-Zhao Theorem. *Annals of Mathematics and Artificial Intelligence*, 47:79–101, 2006.

[11] Norman Foo and Dongmo Zhang. Dealing with the ramification problem in extended propositional dynamic logic. In F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyaschev, editors, *Advances in Modal Logic*, volume 3, pages 173–191, Leipzig, Germany, 2002. World Scientific.

[12] Jeremy Forth and Rob Miller. Ramifications: An Extension and Correspondence Result for the Event Calculus. *Journal of Logic and Computation*, 17(4):639–685, 2007.

[13] Michael Gelfond and Vladimir Lifschitz. The Stable Model Semantics for Logic Programming. In *Proceedings of the International Conference on Logic Programming (ICLP)*, pages 1070–1080. The MIT Press, 1988.

[14] Michael Gelfond and Vladimir Lifschitz. Action Languages. *Electronic Transactions on Artificial Intelligence*, 3, 1998.

[15] Michael Gelfond and Vladimir Lifschitz. The Common Core of Action Languages B and C. In Riccardo Rosati and Stefan Woltran, editors, *Proceedings of the Fourteenth International Workshop on Non-Monotonic Reasoning (NMR)*, June 2012.

[16] Matthew L. Ginsberg and David E. Smith. Reasoning about Action I: A Possible Worlds Approach. *Artificial Intelligence*, 35:233–258, 1987.

[17] Joakim Gustafsson and Patrick Doherty. Embracing Occlusion in Specifying the Indirect Effects of Actions. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*, pages 87–98. Morgan Kaufmann, November 1996.

[18] Andreas Herzig and Ivan José Varzinczak. Metatheory of actions: Beyond consistency. *Artificial Intelligence*, 171(16–17):951–984, 2007.

[19] Antonios Kakas and Rob Miller. Reasoning about actions, narratives and ramifications. *Electronic Transactions on Artificial Intelligence*, 1(4), 1997.

[20] Antonis Kakas, Loizos Michael, and Rob Miller. Modular-E: An Elaboration Tolerant Approach to the Ramification and Qualification Problems. In *Proceedings of the Eight International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-05)*, pages 211–226. Springer, September 2005.

[21] Antonis Kakas, Loizos Michael, and Rob Miller. Modular-E and the role of elaboration tolerance in solving the qualification problem. *Artificial Intelligence*, 175(1):49–78, 2011. John McCarthy's Legacy.

[22] Robert A. Kowalski and Marek J. Sergot. A logic-based calculus of events. *New Generation Computing*, 4 (1):67–95, 1986.

[23] Jonas Kvarnström and Patrick Doherty. Tackling the Qualification Problem using Fluent Dependency Constraints. *Computational Intelligence*, 16(2):169–209, 2000.

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

[25] Fangzhen Lin. Embracing Causality in Specifying the Indirect Effects of Actions. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1985–1993, Montréal, Canada, August 1995. Morgan Kaufmann.

[26] Fangzhen Lin. Embracing causality in specifying the indeterminate effects of actions. In B. Clancey and D. Weld, editors, *Proceedings of the Thirteenth AAAI Conference on Artificial Intelligence (AAAI-96)*, pages 670–676, Portland, OR, August 1996. MIT Press.

[27] Fangzhen Lin and Mikhail Soutchanski. Causal theories of actions revisited. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI-11)*. AAAI Press, August 2011.

[28] Fangzhen Lin and Yuting Zhao. ASSAT: Computing Answer Sets of a Logic Program by SAT Solvers. *Artificial Intelligence*, 157(1-2):115–137, 2004.

[29] Norman McCain and Hudson Turner. A Causal Theory of Ramifications and Qualifications. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1978–1984, Montréal, Canada, August 1995. Morgan Kaufmann.

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

[31] Sheila McIlraith. Integrating Actions and State Constraints: A Closed-Form Solution to the Ramification Problem (Sometimes). *Artificial Intelligence*, 116(1–2):87–121, January 2000.

[32] Erik Mueller. *Commonsense Reasoning*. Morgan Kaufmann, 2006.

[33] Javier Pinto. Compiling Ramification Constraints into Effect Axioms. *Computational Intelligence*, 15:280–307, 1999.

[34] Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, September 2001. ISBN 0262182181.

[35] Murray Shanahan. The Ramification Problem in the Event Calculus. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 140–146. Morgan Kaufmann, August 1999. ISBN 1-55860-613-0.

[36] Hannes Strass. *Default Reasoning about Actions*. PhD thesis, Leipzig University, June 2012.

[37] Michael Thielscher. Computing Ramifications by Postprocessing. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1994–2000, Montréal, Canada, August 1995. Morgan Kaufmann.

[38] Michael Thielscher. Ramification and Causality. *Artificial Intelligence*, 89(1–2):317–364, 1997.

[39] Michael Thielscher. A Unifying Action Calculus. *Artificial Intelligence*, 175(1):120–141, 2011.

[40] Kristof Van Belleghem, Marc Denecker, and D. Theseider-Dupré. A Constructive Approach to the Ramification Problem. In *Reasoning about Actions; Foundations and Applications*, pages 1–17, 1998.

[41] David E. Wilkins. *Practical Planning: Extending the Classical AI Planning Paradigm*. Morgan Kaufmann, San Mateo, CA, 1988.

[42] Marianne Winslett. Reasoning about Action Using a Possible Models Approach. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 89–93, 1988.

[43] Dongmo Zhang and Norman Foo. EPDL: A logic for causal reasoning. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 131–138, Seattle, 2001. Morgan Kaufmann.

[44] Dongmo Zhang and Norman Foo. Interpolation properties of action logics: Lazy-formalization to the frame problem. In S. Flesca, S. Greco, N. Leone, and G. Ianni, editors, *Logics in Artificial Intelligence, European Conference (JELIA-02)*, volume 2424 of *LNCS*, pages 357–368, Cosenza, Italy, September 2002. Springer.