The Ramification Problem in the Event Calculus

Murray Shanahan

Department of Electrical and Electronic Engineering,
Imperial College,
Exhibition Road,
London SW7 2BT,
England.
m.shanahan@ic.ac.uk

Abstract

Finding a solution to the frame problem that is robust in the presence of actions with indirect effects has proven to be a difficult task. Examples that feature the instantaneous propagation of interacting indirect effects are particularly taxing. This article shows that an already widely known predicate calculus formalism, namely the event calculus, can handle such examples with only minor enhancements.

Introduction

The *ramification problem,* that is to say the frame problem in the context of actions with indirect effects, has attracted considerable attention recently [McCain & Turner, 1995], [Lin, 1995], [Gustafsson & Doherty, 1996], [Sandewall, 1996], [Shanahan, 1997], [Thielscher, 1997], [Kakas & Miller, 1997], [Denecker, *et al,* 1998]. The purpose of this paper is to demonstrate that the standard benchmark scenarios for the ramification problem can be handled by the event calculus, as presented in Chapter 16 of [Shanahan, 1997], without introducing any significant new logical machinery.

Following [Shanahan, 1997], this article presents the event calculus in the first-order predicate calculus, augmented with circumscription. In this form, it can be used to represent a variety of phenomena, including concurrent action, actions with non-deterministic effects, and continuous change [Shanahan, 1997].

The event calculus can also be used to represent actions with indirect effects, as shown in [Shanahan, 1997]. However, certain types of domains are problematic. These involve the instantaneous propagation of interacting indirect effects, as exemplified by Thielscher's circuit benchmark [1997]. Staying within the framework of the event calculus, and introducing just two new predicates and two new axioms, this article presents a general technique for representing actions with indirect effects that encompasses such domains.

1 Event Calculus Basics

The event calculus used in this paper is drawn directly from Chapter 16 of [Shanahan, 1997]. Its ontology includes actions (or events), fluents and time points. The formalism's basic predicates are as follows. Initiates(α,β,τ) means fluent β starts to hold after action α at time τ, Terminates(α,β,τ) means fluent β ceases to hold after action α at time τ, Releases(α,β,τ) means fluent β is not subject to inertia after action α at time τ, Initially$_P(\beta)$ means fluent β holds from time 0, InitiallyN(P) means fluent β does not hold from time 0, Happens(a,) means action a occurs at time τ, and HoldsAt(β,τ) means fluent β holds at time τ.

Given a collection of *effect axioms,* expressed as Initiates, Terminates and Releases formulae, and a *narrative* of events, expressed as Happens, Initially N, Initiallyp and temporal ordering formulae, the axioms of the event calculus yields HoldsAt formulae that tell us which fluents hold at what time points. Here are the axioms, whose conjunction will be denoted EC.

$$\mathrm{HoldsAt}(f,t) \leftarrow \mathrm{Initially}_P(f) \wedge \neg\, \mathrm{Clipped}(0,f,t) \qquad \text{(EC1)}$$

$$\begin{aligned}&\mathrm{HoldsAt}(f,t2) \leftarrow \qquad \text{(EC2)}\\ &\quad \mathrm{Happens}(a,t1) \wedge \mathrm{Initiates}(a,f,t1) \wedge\\ &\quad\quad t1 < t2 \wedge \neg\, \mathrm{Clipped}(t1,f,t2)\end{aligned}$$

$$\begin{aligned}&\mathrm{Clipped}(t1,f,t3) \leftrightarrow \qquad \text{(EC3)}\\ &\quad \exists\, a,t2\, [\mathrm{Happens}(a,t2) \wedge t1 < t2 \wedge t2 < t3 \wedge\\ &\quad\quad [\mathrm{Terminates}(a,f,t2) \vee \mathrm{Releases}(a,f,t2)]]\end{aligned}$$

$$\begin{aligned}&\neg\, \mathrm{HoldsAt}(f,t) \leftarrow \qquad \text{(EC4)}\\ &\quad \mathrm{Initially}_N(f) \wedge \neg\, \mathrm{Declipped}(0,f,t)\end{aligned}$$

$$\begin{aligned}&\neg\, \mathrm{HoldsAt}(f,t2) \leftarrow \qquad \text{(EC5)}\\ &\quad \mathrm{Happens}(a,t1) \wedge \mathrm{Terminates}(a,f,t1) \wedge\\ &\quad\quad t1 < t2 \wedge \neg\, \mathrm{Declipped}(t1,f,t2)\end{aligned}$$

$$\begin{aligned}&\mathrm{Declipped}(t1,f,t3) \leftrightarrow \qquad \text{(EC6)}\\ &\quad \exists\, a,t2\, [\mathrm{Happens}(a,t2) \wedge t1 < t2 \wedge t2 < t3 \wedge\\ &\quad\quad [\mathrm{Initiates}(a,f,t2) \vee \mathrm{Releases}(a,f,t2)]]\end{aligned}$$

The frame problem is overcome using circumscription.

Given a conjunction Σ of Initiates, Terminates and Releases formulae, a conjunction Δ of Initiallyp, Initially N, Happens and temporal ordering formulae, and a conjunction Ω of uniqueness-of-names axioms for actions and fluents, we're interested in,

$$\begin{aligned}&\mathrm{CIRC}[\Sigma\ ;\ \mathrm{Initiates, Terminates, Releases}] \wedge\\ &\quad \mathrm{CIRC}[\Delta\ ;\ \mathrm{Happens}] \wedge \mathrm{EC} \wedge \Omega.\end{aligned}$$

In all the cases we're interested in, Σ and Δ are in a form which, according to a theorem of Lifschitz, guarantees that these circumscriptions are equivalent to the predicate completions of Initiates, Terminates, Releases and Happens.

2 State Constraints

The *ramification problem* is the frame problem for actions with indirect effects, that is to say actions with effects beyond those described explicitly by their associated effect axioms. Although it's always possible to encode these indirect effects as direct effects instead, the use of constraints describing indirect effects ensures a modular representation and can dramatically shorten an axiomatisation. One way to represent actions with indirect effects is through *state constraints,* the focus of this section. These express logical relationships that have to hold between fluents at all times.

In the event calculus, state constraints are HoldsAt formulae with a universally quantified time argument. Here's an example, whose intended meaning should be obvious.

$$\text{HoldsAt}(\text{Happy}(x),t) \leftrightarrow \neg \text{HoldsAt}(\text{Hungry}(x),t) \wedge \neg \text{HoldsAt}(\text{Cold}(x),t) \quad \text{(H1.1)}$$

Note that this formula incorporates fluents with arguments. Actions may also be parameterised, as in the following effect axioms.

$$\text{Terminates}(\text{Feed}(x),\text{Hungry}(x),t) \quad \text{(H2.1)}$$

$$\text{Terminates}(\text{Clothe}(x),\text{Cold}(x),t) \quad \text{(H2.2)}$$

Here's a narrative for this example.

$$\text{Initially}_P(\text{Hungry}(\text{Fred})) \quad \text{(H3.1)}$$

$$\text{Initially}_N(\text{Cold}(\text{Fred})) \quad \text{(H3.2)}$$

$$\text{Happens}(\text{Feed}(\text{Fred}), 10) \quad \text{(H3.3)}$$

Finally we need some uniqueness-of-names axioms.

$$\text{UNA}[\text{Feed}, \text{Clothe}] \quad \text{(H4.1)}$$

$$\text{UNA}[\text{Hungry}, \text{Cold}] \quad \text{(H4.2)}$$

The incorporation of state constraints has negligible impact on the solution to the frame problem already presented. However, state constraints must be conjoined to the theory outside the scope of any of the circumscriptions. Given a conjunction Σ of Initiates, Terminates and Releases formulae, a conjunction Δ of Initiallyp, InitiallyN, Happens and temporal ordering formulae, a conjunction Ψ of state constraints, and a conjunction Ω of uniqueness-of-names axioms for actions and fluents, we're interested in,

$$\text{CIRC}[\Sigma\ ; \text{Initiates, Terminates, Releases}] \wedge \text{CIRC}[\Delta\ ; \text{Happens}] \wedge \text{EC} \wedge \Psi \wedge \Omega.$$

For the current example, if we let Σ be the conjunction of (H2.1) and (H2.2), Δ be the conjunction of (H3.1) to (H3.3), Ψ be (H1.1), and Ω be the conjunction of (H4.1) and (H4.2), we have,

$$\text{CIRC}[\Sigma\ ; \text{Initiates, Terminates, Releases}] \wedge \text{CIRC}[\Delta\ ; \text{Happens}] \wedge \text{EC} \wedge \Psi \wedge \Omega \models \text{HoldsAt}(\text{Happy}(\text{Fred}),11).$$

State constraints must be used with caution. As can be seen by inspection, Axioms (EC1) to (EC6) enforce the following principle: *a fluent that has been initiated/terminated directly through an effect axiom cannot then be terminated/initiated indirectly through a state constraint, unless it is released beforehand.* Similarly, a fluent that holds at time 0 because of an Initiallyp formula cannot then be terminated indirectly through a state constraint, unless it's released beforehand, and a fluent that does not hold at time 0 because of an InitiallyN formula cannot then be initiated indirectly through a state constraint, unless it's released beforehand.

Suppose, in the present example, we introduced an Upset(x) event whose effect is to terminate Happy(x). Then the addition of Happens(Upset(Fred),12) would lead to contradiction. Similarly, the addition of InitiallyN(Happy(Fred)) would lead to contradiction.

State constraints are most useful when there is a clear division of fluents into *primitive and derived.* Effect axioms are used to describe the dynamics of the primitive fluents and state constraints are used to describe the derived fluents in terms of the primitive ones.

3 Effect Constraints

State constraints aren't the only way to represent actions with indirect effects, and often they aren't the right way, as emphasised by Lin [1995] and McCain and Turner [1995]. To see this, we'll take a look at the so-called "walking turkey shoot", a variation of the Yale shooting problem in which the Shoot action, as well as directly terminating the Alive fluent, indirectly terminates a fluent Walking.

The effect axioms are inherited from the Yale shooting problem.

$$\text{Initiates}(\text{Load},\text{Loaded},t) \quad \text{(W1.1)}$$

$$\text{Terminates}(\text{Shoot},\text{Alive},t) \leftarrow \text{HoldsAt}(\text{Loaded},t) \quad \text{(W1.2)}$$

The narrative of events is as follows.

$$\text{Initially}_P(\text{Alive}) \quad \text{(W2.1)}$$

$$\text{Initially}_P(\text{Loaded}) \quad \text{(W2.2)}$$

$$\text{Initially}_P(\text{Walking}) \quad \text{(W2.3)}$$

$$\text{Happens}(\text{Shoot},T1) \quad \text{(W2.4)}$$

$$T1 < T2 \quad \text{(W2.5)}$$

We have two uniqueness-of-names axioms.

$$\text{UNA}[\text{Load}, \text{Shoot}] \quad \text{(W3.1)}$$

$$\text{UNA}[\text{Loaded}, \text{Alive}, \text{Walking}] \quad \text{(W3.2)}$$

Now, how do we represent the dependency between the Walking and Alive fluents so as to get the required indirect effect of a Shoot action? The obvious, but incorrect, way is to use a state constraint.

$$\text{HoldsAt}(\text{Alive},t) \leftarrow \text{HoldsAt}(\text{Walking},t)$$

The addition of this state constraint to the above formalisation would yield inconsistency, because it violates the rule that a fluent, in this case Walking, that holds directly through an Initiallyp formula cannot be terminated indirectly through a state constraint. (The same problem would arise if the Walking fluent had been initiated directly by an action.)

A better way to represent the relationship between the Walking fluent and the Alive fluent in the walking turkey shoot is through an *effect constraint.* Effect constraints are Initiates and Terminates formulae with a single universally quantified action variable. The constraint we require for this example is the following.

Terminates(a,Walking,t) ← Terminates(a,Alive,t) (W4.1)

Notice that effect constraints are weaker than state constraints: the possibility of resurrecting a corpse by making it walk, inherent in the faulty state constraint, is not inherent in this formula.

Let Σ be the conjunction of (W1.1), (W1.2) and (W4.1). Let Δ be the conjunction of (W2.1) to (W2.5), and Ω be the conjunction of (W3.1) and (W3.2). We have,

CIRC[Σ ; Initiates, Terminates, Releases] ∧
  CIRC[Δ ; Happens] ∧ EC ∧ Ω ⊨
    ¬ HoldsAt(Walking,T2).

Effect constraints are adequate for the representation of many actions with indirect effects. But there is still a class of examples for which they don't work. Consider the following benchmark problem due to Thielscher [1997]. A circuit comprising a battery, three switches, a relay, and a light bulb is wired up as in Figure 1.

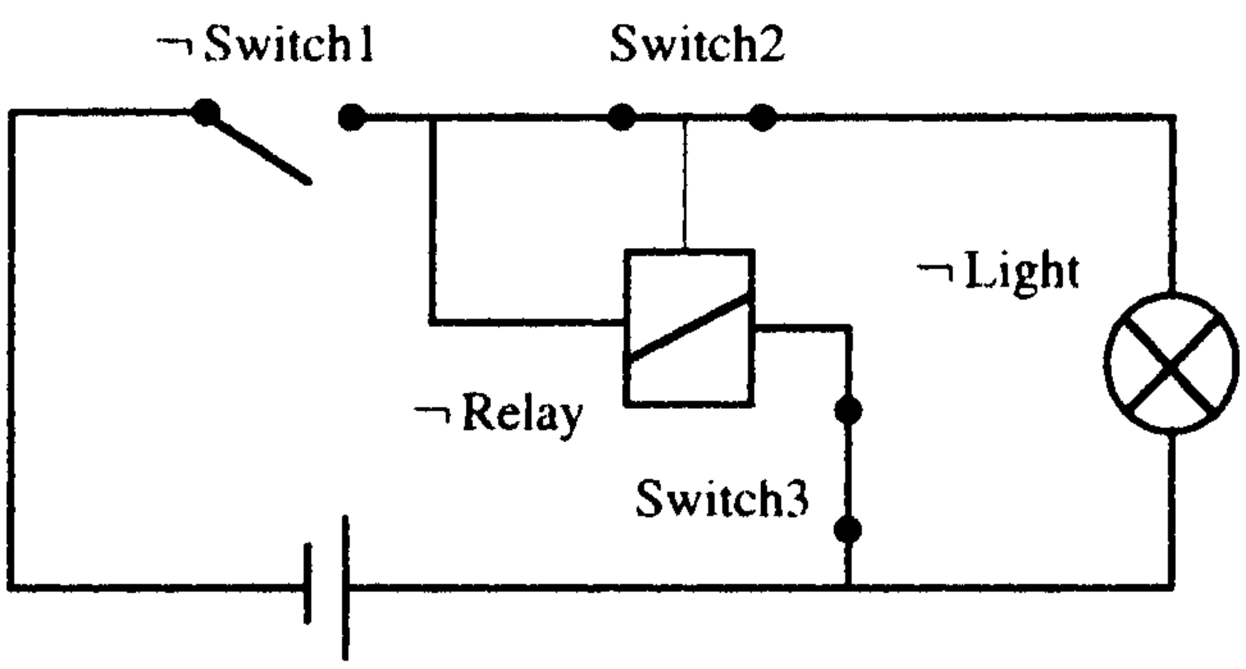

Figure 1: Thielscher's Circuit

Five fluents represent the state of each component in the circuit: Switch 1, Switch2, Switch3, Relay, and Light. Their initial configuration is as in Figure 1. There are various dependencies among the fluents. The light is on if switches one and two are closed. Switch two is open if the relay is on. Finally, the relay is on if switches 1 and 3 are closed. When switch 1 is closed, the relay becomes activated, switch 2 will open, and the light stays off. The awkward nature of this example derives from the fact that closing switch 1 has one indirect effect (closing the relay, which opens switch 2) that disables another indirect effect (the light coming on).

A first, naive attempt to formalise this example might include an effect constraint like the following.

Initiates(a,Light,t) ←
  Initiates(a,S witch 1 ,∧ HoldsAt(S witch2,t)

But this formula is obviously a false start, because in this scenario, initiating Switch1 also indirectly terminates Switch2, and the event calculus axioms entail that Switch2 still holds at the instant of termination. A better attempt would be the following effect constraint.

Initiates(aJLighU) ←
  Initiates(a,S witch l,t) ∧ Holds At(Switch2,t) ∧
    -i Terminates(a,Switch2,t)

This formula is adequate for this particular scenario, but doesn't fully capture the dependency between the fluents. Suppose, for example, that switch 1 is initially closed, while switch 2 and switch 3 are initially open. Then closing switch 2 causes the light to go on, something not captured by this constraint. We need a counterpart to the above formula for this case.

Initiates(a,Light,t) ←
  Initiates(a,Switch2,t) ∧ Holds At(S witch 1 ,t∧
    -i Terminates(a,S witch l,t)

Once again, while this is adequate for the present example, it's not a general solution. In particular, neither of these formulae accounts for the possibility of independent but concurrent switch events.

In the following section, a method for representing the indirect effects of actions is presented whose generality is comparable to that of other recently published solutions to the ramification problem, but which doesn't require the development of significantly more logical machinery than is already present in the event calculus defined above.

4 Causal Constraints

Following a common practise in recent literature on the ramification problem, let's introduce some shorthand notation for expressing dependencies between fluents.

Definition 4.1. A *fluent symbol* is any string of characters starting with an upper-case letter. ☐

Definition 4.2. Any fluent symbol is also *a fluent formula.* If φ and ψ are fluent formulae, then so are ¬ φ, φ ∧ ψ, φ ∨ ψ, φ ← ψ, φ → ψ and φ ↔ ψ. ☐

Definition 4.3. Following the notation of IDenecker, *et* «/., 1998], a *causal constraint* is a formula of the form,

initiating Π causes β

or,

initiating Π causes ¬ β

where Π is a fluent formula and β is a fluent symbol. ☐

Here's a subset of the fluent dependencies in Thielscher's circuit expressed using this notation.

initialing Switch l ∧ switch2 causes Light

initiating Relay causes -> Switch2

Moating Switchl ∧ Switch3 causes Relay

There are other dependencies in the circuit. For example, this set of dependencies neglects to specify the conditions under which the light goes off. But these can be ignored for the example narrative we're interested in here.

Formulae like these are intended to have an intuitive meaning. The translation into the event calculus detailed below could be thought of as one attempt to give them a precise semantics. Alternatively, these formulae can be thought of simply as syntactic sugar for more long-winded event calculus formulae of the particular form defined below.

4.1 Causal Constraints in the Event Calculus

The key to correctly representing causal constraints in the event calculus is first to introduce new events that update each fluent whose value is dependent on other fluents, and second to write formulae ensuring that these events are triggered whenever those influencing fluents attain the appropriate values. (A related proposal is made by Pinto [1998] in the context of the situation calculus.)

To guarantee the instantaneous propagation of the effects of such events, they must be triggered not just when the influencing fluents already have their appropriate values, but also when they are *about to get* those values thanks to other events occurring at the same time. This motivates the introduction of four new predicates, Started, Stopped, Initiated and Terminated. The formula Started(β,τ) means that either β already holds at τ or an event occurs at x that initiates β. Conversely, the formula Stopped(β,τ) means that either β already does not hold at τ or an event occurs at τ that terminates β. The predicates Started and Stopped are defined by the following axioms.

Started(f,t) $\leftrightarrow$ (CC1)
  HoldsAt(f,t) $\vee$
    $\exists$ a [Happens(a,t) $\wedge$ Initiates(a,f,t)]

Stopped(f,t) $\leftrightarrow$ (CC2)
  -i HoldsAt(f,t) $\vee$
    $\exists$ a fHappens(a,t) $\wedge$ Terminates(a,f,t)J

Note that at the instant of a fluent's transition from one value to another, we have both Stopped and Started at the same time.

The formula Initiated(β,τ) means that β has been "started" at τ in the above sense, and furthermore no event occurs at τ that terminates β. Likewise, the formula Terminated(β,τ) means that β has been "stopped" at τ in the above sense, and no event occurs at τ that initiates β. The predicates Initiated and Terminated are defined by the following axioms.

Initiated(f,t) $\leftrightarrow$ (CC3)
  Started(f,t) $\wedge$
    -i 3 a [Happens(a,t) $\wedge$ Terminates(a,f,t)]

Terminated(f,t) $\leftrightarrow$ (CC4)
  Stopped(f,t) $\wedge$
    $\neg \exists$ a [Happens(a,t) $\wedge$ Initiates(a,f,t)]

To represent the causal constraints in Thielscher's circuit example, we introduce three events, LightOn, Open2 and CloseRelay, which are triggered under conditions described by the following formulae.

Happens(LightOn,t) $\leftarrow$ (L 1.1)
  Stopped(LighM) $\wedge$ Initiated(S witch l,t) $\wedge$
    Initiated(Switch2,t)

Happens(Open2,t) $\leftarrow$ (Ll .2)
  Started(Switch2,t) $\wedge$ Initiated(Relay,t)

Happens(CloseRelay ,t) $\leftarrow$ (1.1.3)
  Stopped(Relay ,t) $\wedge$ Initiated(S witch 1 ,t) $\wedge$
    Initiated(Switch3,t)

These triggered events govern the *transition* of fluents from one value to another when certain conditions come about, as prescribed by the corresponding causal constraints. Hence the need for the Stopped and Started conditions in the above formulae. These ensure that an event occurs *only* at the time of the transition in question. The effects of these events are as follows. A Close 1 event is also introduced.

Initiates(LightOn,Light,t) (L2.1)

Terminates(Open2,Switch2,t) (L2.2)

Initiates(CloseRelay,Relay,t) (L2.3)

Initiates(Close 1 ,S witch 1 ,t) *(12 A)*

The circuit's initial configuration, as shown in Figure 1, is as follows.

InitiallyN(Switchl) (L3.1)

Initially p(Switch2) (L3.2)

Initiallyp(Switch3) (L3.3)

InitiallyN(Relay) (L3.4)

Initially N(Light) (L3.5)

The only event that occurs is a Close 1 event, at time 10.

Happens(Close 1,10) (L3.6)

Two uniqueness-of-names axioms are required.

UNA[LightOn, Close 1, Open2, CloseRelay] (L4.1)

UNAfSwitchl, Switch2, Switch3, Relay, Light] (L4.2)

As the following proposition shows, this formalisation of Thielscher's circuit yields the required logical consequences. In particular, the relay is activated when switch 1 is closed, causing switch 2 to open, and the light does not come on.

Proposition 4.4. Let Σ be the conjunction of (L2.1) to (L2.4), Δ be the conjunction of (Ll.l) to (Ll.3) with (L3.1) to (L3.6), Ψ be the conjunction of (CO) to (CC4), and *Q* be the conjunction of (L4.1) and (L4.2). We have,

CIRC[Σ ;; Initiates, Terminates, Releases] $\wedge$
  CIRC[Δ,; Happens] $\wedge$ EC $\wedge$ Ψ $\wedge$ Ω $\models$
    HoldsAt(Relay,20) $\wedge$ -! HoldsAt(Switch2,20) $\wedge$
      -i HoldsAt(Light,20).

Proof. From CIRC[Σ ; Initiates, Terminates, Releases] we get the completions of Initiates, Terminates and Releases. From CIRC[Δ ; Happens] we get the completion of Happens, namely,

Happens(a,t) $\leftrightarrow$ 14.5)
  [a = Close 1 $\wedge$ t = 10 J $\vee$
  [a = LightOn $\wedge$ Stopped(Light.t) $\wedge$
    Initiated(S witch l,t) $\wedge$ Inttiatcd(Switch2,t)J $\vee$
  [a = Open2 $\wedge$ Started(Switch2,t) $\wedge$ Initiated(Relay,t)] $\vee$
  [a = CloseRelay $\wedge$ Stopped(Relay,t) $\wedge$
    Initiated(S witch l,t) $\wedge$ Initiated(Switch3,t)|.

At the time of the first event, the fluents Switch2 and Switch3 hold and the fluents Switch1, Relay and Light don't hold.

First we prove that the Close1 event at time 10 is the first event. Consider any $t <$ 10. There can't be a Closel event at t, since, from [4.5], the only Closel event is at 10. Since we have - HoldsAt(Switchl,t) and only a Closel event can initiate Switch!, we have -i Initiated(Switchl,t), so, from [4.5], there can't be a LightOn or CloseRelay event at t. Since we have -i HoldsAt(Relay,t) and there can't be a CloseRelay event at t, we have -» Initiated(Relay,t), and therefore, from [4.5], there can't be an Open2 event at t. From [4.5], this exhausts all the possible types of event, so there can't be any event occurrence at time t. So the Closel event at time 10 is the first event.

Now we prove that a Closel event, a CloseRelay event and an Open2 event all occur at time 10, but that no LightOn event occurs at time 10. We know directly from [4.5] that a Closel event occurs at 10. Therefore, since there is no type of event that can terminate Switch 1, we have Initiated(Switchl,10), given (L2.4). We know that Stopped(Rclay,10) since we have -i HoldsAt(Relay, 10), and since we have HoldsAt(Switch3,10), we also have Initiated(Switch3,10). So, from [4.5], we know that a CloseRelay event occurs at 10. Since a CloseRelay event occurs at 10 and there is no type of event that can initiate Relay, we have Initiated(Relay,10), given (L2.3). We also know that HoldsAt(Switch2,10) and therefore Started(Switch2,10). So, from [4.5], we know that an Open2 event occurs at time 10. Since there is an Open2 event at 10, which, from (L2.2), terminates Switch2, we have -i Initiated(Switch2,I0), and therefore, from [4.5] there cannot be a LightOn event at 10.

Using a similar argument to the paragraph before last, we can show that no events occur after time 10. Given the events that occur at time 10, it's then straightforward to prove, from Axioms (EC2) and (EC5), that the fluent Relay holds at time 20, but the fluents Switch2 and Light do not.D

Let's briefly consider a couple of minor variations on this example. First, suppose we augment the formalisation with a Close2 action which initiates Switch2. Then the addition of the formula Happens(Close2,15) will give rise to a contradiction, since we would have both a Close2 event at time 15 and, from (LI.2), an Open2 event, enabling us to prove, for any time t after 15, both HoldsAt(Switch2,t) and -i HoldsAt(Switch2,15). In other words, switch 2 cannot be manually closed while switches 1 and 3 are closed, thanks to the relay.

Now consider the original narrative of events, but with a different initial situation, one in which switch 3 is open, then, as desired, we get a different result: the relay isn't activated, switch 2 doesn't open, so the light does come on.

InitiallyN(Switchl) (L5.1)

Initiallyp(Switch2) (L5.2)

Initially_N(Switch3) (L5.3)

Initially_N(Relay) (L5.4)

InitiallyN(Light) (L5.5)

Proposition 4.6. Retaining Σ, Δ, Ψ and Ω as above, let Δ be the conjunction of (L5.1) to (L5.5) with (L3.5). Then we have,

$$\text{CIRC}[\Sigma \text{ ; Initiates, Terminates, Releases}] \wedge \text{CIRC}[\Delta \text{ ; Happens}] \wedge EC \wedge \Psi \wedge \Omega \vDash \text{HoldsAt(Light,20)}.$$

Proof. The proof is similar to that of Proposition 4.4. □

5 From Causal Constraints to Event Calculus

This section presents a general translation from the shorthand notation for causal constraints presented above into the event calculus, along the lines suggested by the preceding example.

Definition 5.1. A *negated fluent symbol* is a fluent formula of the form $\neg \beta$ where β is a fluent symbol. □

First we define the function T_c, which translates a single causal constraint into a pair of event calculus formulae.

Definition 5.2. Let ψ be a causal constraint of the form,

initiating Π causes γ

where Π is a fluent formula and γ is either a fluent symbol or a negated fluent symbol. The *translation* $T_c(\psi)$ *of* ψ *with new action name* α is the pair $\langle\sigma,\delta\rangle$,, where δ and σ are defined as follows. Let Π' be Π with every negated fluent symbol $\neg \beta$ replaced by Terminated(β,t) and every other fluent symbol β replaced by Initiated(β,t). I γ s a negated fluent symbol $\neg \beta$, then σ is,

Terminates(α,β,t)

and 8 is,

$$\text{Happens}(\alpha,t) \leftarrow \text{Started}(\beta,t) \wedge \Pi'.$$

Otherwise σ is,

Initiates(a,Y,t)

and 5 is,

$$\text{Happens}(\alpha,t) \leftarrow \text{Stopped}(\gamma,t) \wedge \Pi'. \quad □$$

Next we define the function T_c^*, which translates a set of causal constraints into a pair of conjunctions of event calculus formulae.

Definition 5.3. Let Φ be a finite set of causal constraints $\{\psi_1, ..., \psi_n\}$. The *translation* $T_c^*(\Phi)$ *of* Φ *with new action names* α_1 to α_n is the pair $\langle\Sigma,\Delta\rangle$, where Σ is $\sigma_1 \wedge ... \wedge \sigma_n$ and Δ is $\delta_1 \wedge ... \wedge \delta_n$, given that for any $1 \leq i \leq n$, $T_c(\psi_i)$ with new action name α_i is $\langle\sigma_i,\delta_i\rangle$. □

5.1 Limitations: The Gear Wheels Example

Although the technique described here can represent the indirect effects of many different types of actions, it does *not* work well in scenarios involving mutually dependent fluents, such as the following example, which is taken from [Denecker, *et al.*$_y$ 1998]. There are two interlocking gear wheels. If one is turning, the other must be turning, and if one is stationary, the other must be stationary. The example is formalised using two fluents, Turning 1 and Turning2.

initiating Turning 1 causes Turning2

initiating Tuming2 pauses Turning 1

initiating -Turning 1 causes -Turning2

initiating - Turning2 causes - Turning1

The proposed event calculus translation of these causal constraints does not yield the desired conclusions, as it cannot rule out phantom self-starting events that cause the wheel to turn. (Note, however, that this example can be correctly formalised using the state constraints of Section 2.) As illustrated in the next section, other examples with cycles of dependencies are handled more satisfactorily.

6 Vicious Cycles

Consider the modification of Thielscher's circuit depicted in Figure 2. This circuit incorporates a potentially vicious cycle of fluent dependencies. If switch 1 is closed, the relay is activated, opening switch 2, which prevents the relay from being activated. Given Axioms (CC1) to *(CCA)* in their present form, the formalisation of this scenario using causal constraints will yield inconsistency.

Here are the causal constraints Φ.

Miiatjj2£ Relay causes -i Switch2

<u>initiating</u> Switch $\wedge$ Switch2 $\wedge$ Switch3 <u>causes</u> Relay

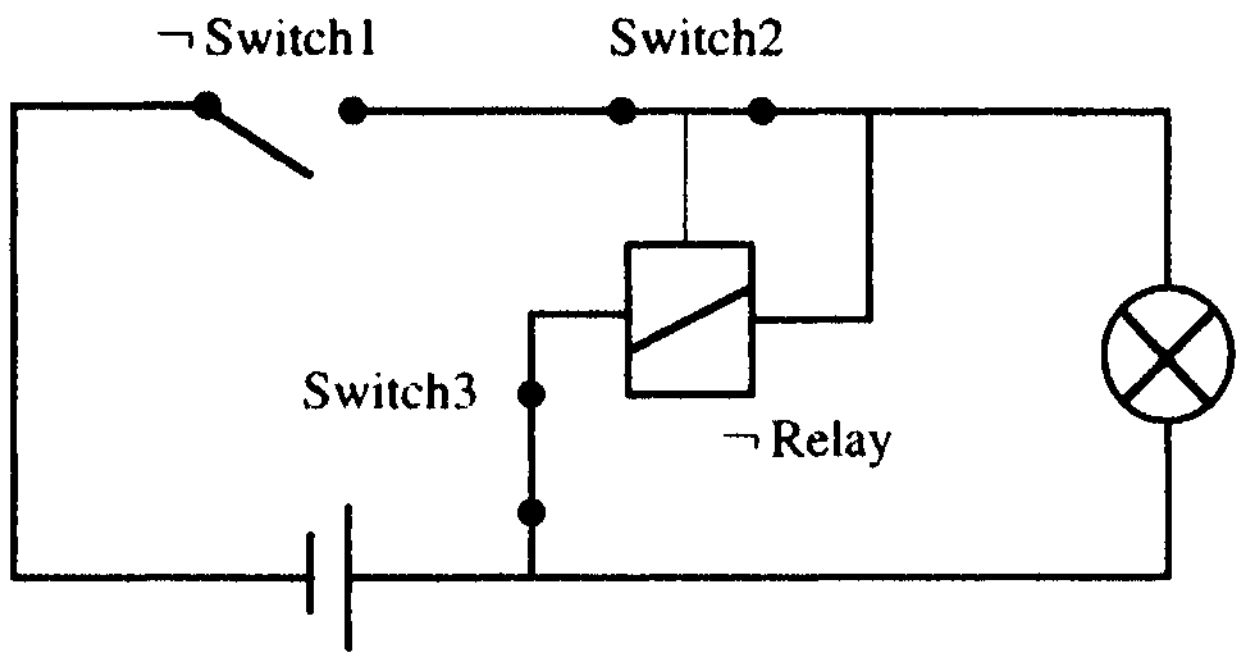

Figure 2: A Modification of Thielscher's Circuit

Let $\langle \Sigma, \Delta \rangle$ be $T_C^*(\Phi)$ with new action names Open2 and CloseRelay. Then A is the conjunction of the following Happens formulae, and Σ is the conjunction of the following Initiates and Terminates formulae.

Happens(Open2,t) $\leftarrow$ (V1.1)
Started(Switch2,t) $\wedge$ Initiated(Relay,t)

Happens(CloseRelay,t) $\leftarrow$ (V1.2)
Stopped(Relay,t) $\wedge$ Initiated(Switch1,t) $\wedge$
Initiated(Switch2,t) $\wedge$ Initiated(Switch3,t)

Terminates(Open2,Switch2,t) (V2.1)

Initiates(CloseRelay,Relay,t) (V2.2)

Initiates(Close 1 ,S witch 1 ,t) (V2.3)

The circuit's initial configuration is as follows.

Initially_N(Switchl) (V3.1)

Initiallyp(Switch2) (V3.2)

Initiallyp(Switch3) (V3.3)

Initially_N(Relay) (V3.4)

The only event that occurs is a Close 1 event, at time 10.

Happens(CloseUO) (V3.5)

Here are the customary uniqueness-of-names axioms.

UNA[Closel, Open2, CloseRelay] (V4.1)

UNA[Switchl, Switch2, Switch3, Relay] (V4.2)

Proposition 6.1. Let Σ be the conjunction of (VI.1) and (V1.2), Δ be the conjunction of (V1.1) and (VI.2) with (V3.1) to (V3.5), Ψ be the conjunction of (CC1) to (CC4), and Ω be the conjunction of (V4.1) and (V4.2). The following formula is inconsistent.

CIRC[L ; Initiates, Terminates, Releases] $\wedge$
CIRC[Δ ; Happens] $\wedge$ EC $\wedge$ Ψ $\wedge$ Ω.

Proof. From CIRC[Δ ; Happens] we get,

Happens(a,t) $\leftrightarrow$ [6.2]
[a = Close $\wedge$ t = 10] $\vee$
[a = Open2 $\wedge$ Started(Switch2,t) $\wedge$ Initiated(Relay,t)] $\vee$
[a = CloseRelay $\wedge$ Stopped(Relay,t) $\wedge$
Initiated(Switch1,t) $\wedge$ Initiated(Switch2,t) $\wedge$
Initiated(Switch3,t)].

Using the techniques of the proof of Proposition 4.4, we can show that the formula entails that the first event occurs at time 10. At time 10, Switch2 and Switch3 hold, but Switch 1 and Relay do not hold. We know that a Closel event occurs at 10. Now suppose no Open2 event occurs at 10. Then, since Open2 is the only event type that can terminate Switch2, we have Initiated(Switch2,10), which, since we have Stopped(Relay,10), Initiated(Switch 1,10) and Initiated(Switch3,10), entails that a CloseRelay event occurs at 10 from [6.2], But if a CloseRelay event occurs at 10, then we have Initiated(Relay,10) and, from 16.2], an Open2 event also occurs at 10, which contradicts out initial assumption.

So an Open2 event must occur at 10. But then, from [6.2], we must have Initiated(Relay,10). From (CC3) and 16.2], this entails that a CloseRelay event must occur at 10. From [6.2], this gives us Initiated(Switch2,10). But since an Open2 event occurs at 10, which terminates Switch2, this contradicts (CC3). Therefore the formula has no models. □

Note that the cycle in this example is only "dangerous*' if switch 3 is initially closed. If switch 3 is initially open, the correspondingly modified theory is consistent, and yields the expected conclusion that the relay remains inactive after the Closel event.

Arguably, inconsistency is not the most desirable response to an example with a vicious cycle. A formalisation that yielded non-determinism instead would at least permit other useful conclusions to be drawn. Moreover, suppose the initial state of switch 3 is unknown, and (V3.3) is omitted. Then, the threat of inconsistency ensures that InitiallyN(Switch3) follows from the theory, even though no Initially^ formula to that effect is included. This seems a little counter-intuitive.

On the other hand, the aim of formalisation should be to avoid inconsistency. The fact that inconsistency can result here simply from selecting an inappropriate initial state for switch 3 indicates that the wrong level of abstraction has been chosen for representing this particular domain. If we want to represent it in earnest (not just for illustrative purposes), a level of abstraction should be chosen in which every possible narrative that is itself consistent results in a consistent theory. (In the present case, this would demand the inclusion of explicit delays in the model.)

Concluding Remarks

The works of Lin [1995], of Gustafsson and Doherty [1996], and of Thielscher [1997] all share an important feature with the present paper. In each case, an existing predicate calculus-based action formalism, respectively the situation calculus, the fluent calculus, and PMON, is extended to handle actions with indirect effects. Moreover, in [Lin, 1995] and [Gustaffson & Doherty, 1996], as in the present article, circumscription policies are deployed which minimise parts of the theory separately.

The solution to the ramification problem offered in the present article is also based on an existing predicate calculus action formalism, namely the event calculus. As such, it doesn't demand the introduction of any new semantic machinery. Moreover, the proposal is conservative in the sense that it only adds to the existing calculus, the extension comprising four new axioms and four new predicates. With these axioms in place, the proposed solution is little more than a novel style of writing certain event calculus formulae.

No formal assessment has yet been undertaken of the range of applicability of the proposed solution to the ramification problem, as recommended by [Sandewall, 1996). This, along with a more formal comparison with other approaches, would be a good topic for future research.

Acknowledgments

Thanks to Patrick Doherty, Rob Miller, Michael Thielscher, Kristof Van Belleghem and Mark Witkowski for discussions related to the topic of this paper. Thanks also to the anonymous referees who spotted flaws in an earlier version of the paper.

References

[Denecker, *et ai,* 1998] M.Denecker, D.Theseider Dupr6 and K.Van Belleghem, An Inductive Definition Approach to Ramifications, *Electronic Transactions on Artificial Intelligence,* to appear.

[Gustafsson & Doherty, 1996] J.Gustafsson and P.Doherty, Embracing Occlusion in Specifying the Indirect Effects of Actions, *Proceedings 1996 Knowledge Representation Conference (KR 96)*, pp. 87-98.

[Kakas & Miller, 1997] A.Kakas and R.S.Miller, Reasoning about Actions, Narrative and Ramification, *Electronic Transactions on Artificial Intelligence,* vol. 1 (1997), pp. 39-72.

[Lin, 1995] F.Lin, Embracing Causality in Specifying the Indirect Effects of Actions, *Proceedings 1JCA1 95,* pp. 1985-1991.

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

[Pinto, 1998] JJPinto, Concurrent Actions and Interacting Effects, *Proceedings 1998 Knowledge Representation Conference (KR 98),* pp. 292-303.

[Sandewall, 1996] E.Sandewall, Assessments of Ramification Methods that Use Static Domain Constraints, *Proceedings 1996 Knowledge Representation Conference (KR 96),* pp. 99-110.

[Shanahan, 1997] M.P.Shanahan, *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia,* MIT Press, 1997.

[Thielscher, 1997] M.Thielscher, Ramification and Causality, *Artificial Intelligence,* vol. 89 (1997), pp. 317-364.