Linköping Electronic Articles in
Computer and Information Science
Vol. 4(1999): nr 16

The Event Calculus in Classical Logic – Alternative Axiomatisations

Rob Miller

Murray Shanahan

Linköping University Electronic Press
Linköping, Sweden

http://www.ep.liu.se/ea/cis/1999/016/

Published on September 15, 1999 by
Linköping University Electronic Press
581 83 Linköping, Sweden

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

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

Recommended citation:
<Authors>. <Title>. Linköping Electronic Articles in Computer and Information Science, Vol. 4(1999): nr 16. http://www.ep.liu.se/ea/cis/1999/016/. *September 15, 1999.*

This URL will also contain a link to the authors' home pages.

Abstract

We present several alternative classical logic axiomatisations of the Event Calculus, a narrative based formalism for reasoning about actions and change. We indicate the range of applicability and key characteristics of each alternative formulation.

1 Introduction

The "Event Calculus" was originally introduced by Kowalski and Sergot [KoSe 86] as a logic program framework for representing and reasoning about actions (or events) and their effects, especially in database applications. However, the term is now used to refer to a number of different but related languages or formalisms for reasoning about the effects of actions, some of which build upon the logic program tradition, whereas others are described in terms of classical, modal or specialised logics (see Section 4 for citations). This article does not attempt to provide a comprehensive survey of each and every such variant of the Event Calculus (but see Section 4). Instead we focus on describing how the Event Calculus may be axiomatised using classical logic, and using circumscription as a method for default reasoning to solve the frame and related problems. What follows is largely developed from [Shan 97a] and [MiSh 96].

Even if attention is restricted to classical logic formulations of the Event Calculus, there are a number of different choices or variations for the core set of axioms. The various alternatives are each geared to classes of domains with particular restrictions or features; for example to describe systems most naturally viewed as deterministic, as involving both continuous and discrete change, or which require reasoning about the future but not the past. The primary aim of this article is to list and motivate these alternatives. To facilitate this, we first present and illustrate the use of one particular (basic) form of the Event Calculus (in Section 2), with six domain independent axioms labeled (EC1) to (EC6). In subsequent sections possible substitutes for (for example) axiom (EC1) are labeled (EC1a), (EC1b), etc. Inevitably there are incompatibilities and subtleties involved in "mixing and matching" such substitutes, and these are described in the text.

A central feature of the Event Calculus is that it is *narrative-based*, i.e. a time structure which is independent of any action occurrences is established or assumed, and then statements about when various actions occur within this structure are incorporated in the description of the domain under consideration. The time structure is usually assumed or stated to be linear – typically the real or integer number line – although the underlying ideas can equally be applied to other (possibly branching) temporal structures. For the purposes of simplicity, we will assume in this article that time is represented either by the real numbers, the integers, the non-negative reals or the non-negative integers, and that appropriate axioms are included in the theory which establish one of these time structures.

2 An Example Event Calculus Axiomatisation

Informally, the basic idea of the Event Calculus is to state that *fluents* (time-varying properties of the world) are *true* at particular time-points if they have been *initiated* by an action occurrence at some earlier time-point, and not *terminated* by another action occurrence in the meantime. Similarly, a fluent is *false* at a particular time-point if it has been previously terminated and not initiated in the meantime. Domain dependent axioms are provided to describe which actions initiate and terminate which fluents under various circumstances, and to state which actions occur when. In the context of the Event Calculus, individual action occurrences are often referred to as "events", so that "actions" are "event types".

The Event Calculus given here is written in a sorted predicate calculus with equality, with a sort $\mathcal{A}$ for *actions* (variables $a, a_1, a_2, \ldots$), a sort $\mathcal{F}$ for *fluents* (variables $f, f_1, f_2, \ldots$), a sort $\mathcal{T}$ for *timepoints* (here either real numbers or integers, variables $t, t_1, t_2, \ldots$) and a sort $\mathcal{X}$ for *domain objects* (variables $x, x_1, x_2, \ldots$). To describe a very basic calculus we need five predicates (other than equality); $\mathit{Happens} \subseteq \mathcal{A} \times \mathcal{T}$, $\mathit{HoldsAt} \subseteq \mathcal{F} \times \mathcal{T}$, $\mathit{Initiates} \subseteq \mathcal{A} \times \mathcal{F} \times \mathcal{T}$, $\mathit{Terminates} \subseteq \mathcal{A} \times \mathcal{F} \times \mathcal{T}$ and $< \; \subseteq \mathcal{T} \times \mathcal{T}$. $\mathit{Happens}(A,T)$ indicates that action A occurs at time T, $\mathit{HoldsAt}(F,T)$ means that fluent F is true at time T, and $\mathit{Initiates}(A,F,T)$ (respectively $\mathit{Terminates}(A,F,T)$) expresses that if A occurs at T it will initiate (respectively terminate) the fluent F. "$<$" is the standard order relation for time.

It is convenient to also define auxiliary predicates $\mathit{Clipped} \subseteq \mathcal{T} \times \mathcal{F} \times \mathcal{T}$ and $\mathit{Declipped} \subseteq \mathcal{T} \times \mathcal{F} \times \mathcal{T}$ in terms of *Happens*, *Initiates*, *Terminates*, and $<$. $\mathit{Clipped}(T_1, F, T_2)$ (respectively $\mathit{Declipped}(T_1, F, T_2)$) means "the fluent F is terminated (respectively initiated) between times T_1 and T_2." The corresponding definitional axioms[1] are

$$\mathit{Clipped}(t_1, f, t_2) \stackrel{\text{def}}{\equiv} \exists a, t[\mathit{Happens}(a,t) \wedge t_1 \leq t < t_2 \wedge \mathit{Terminates}(a, f, t)] \qquad \text{(EC1)}$$

$$\mathit{Declipped}(t_1, f, t_2) \stackrel{\text{def}}{\equiv} \exists a, t[\mathit{Happens}(a,t) \wedge t_1 \leq t < t_2 \wedge \mathit{Initiates}(a, f, t)] \qquad \text{(EC2)}$$

We can now axiomatise the two principles stated in the introduction to this section. Fluents which have been initiated by an occurrence of an action continue to hold until an occurrence of an action which terminates them

$$\mathit{HoldsAt}(f, t_2) \leftarrow [\mathit{Happens}(a, t_1) \wedge \mathit{Initiates}(a, f, t_1) \wedge t_1 < t_2 \wedge \neg \mathit{Clipped}(t_1, f, t_2)] \qquad \text{(EC3)}$$

and fluents which have been terminated by an occurrence of an action continue not to hold until an occurrence of an action which initiates them:

$$\neg \mathit{HoldsAt}(f, t_2) \leftarrow [\mathit{Happens}(a, t_1) \wedge \mathit{Terminates}(a, f, t_1) \wedge t_1 < t_2 \wedge \neg \mathit{Declipped}(t_1, f, t_2)] \qquad \text{(EC4)}$$

The four axioms above capture the behaviour of fluents once initiated or terminated by an action. But we need also to describe fluents' behaviour before the occurrence of any actions which affect them. We therefore axiomatise a general principle of persistence for fluents; fluents change their truth values only via the occurrence of initiating and terminating actions:

$$\mathit{HoldsAt}(f, t_2) \leftarrow [\mathit{HoldsAt}(f, t_1) \wedge t_1 < t_2 \wedge \neg \mathit{Clipped}(t_1, f, t_2)] \qquad \text{(EC5)}$$

$$\neg \mathit{HoldsAt}(f, t_2) \leftarrow [\neg \mathit{HoldsAt}(f, t_1) \wedge t_1 < t_2 \wedge \neg \mathit{Declipped}(t_1, f, t_2)] \qquad \text{(EC6)}$$

Definitions of the predicates *Happens*, *Initiates* and *Terminates* are given in the domain-dependent part of the theory, as illustrated in the following

[1] By $E_1 \stackrel{\text{def}}{\equiv} E_2$ we mean that expression E_1 is notational shorthand for expression E_2.

example.

2.1 An Example Domain Dependent Axiomatisation

As an example domain dependent theory, we axiomatise a simple scenario of a robot going outside a room by moving through a door, which can be locked and unlocked using an electronic key. For this example we will assume a real number time-line. We will use three fluents, *Inside* (the robot is inside the room), *HasKey* (the robot is holding the electronic key), and *Locked* (the door is locked), and three actions, *Insert* (insert the key in the door), *GoThrough* (move through the door), and *Pickup* (pick up the key). We assume uniqueness-of-names axioms[2] which confirm that all of these constant symbols refer to distinct fluents or actions. Inserting the key alternately locks and unlocks the door. Picking up the key causes the robot to be holding the key, and, going through the unlocked door causes the robot to swap from being inside to outside or vice-versa. We use the predicates *Initiates* and *Terminates* to express these effects:

$$
\begin{aligned}
\mathit{Initiates}(a,f,t) \equiv\ & [[a = \mathit{Pickup}\ \wedge\ f = \mathit{HasKey}] \\
& \vee [a = \mathit{Insert}\ \wedge\ f = \mathit{Locked} \\
& \qquad \wedge \neg \mathit{HoldsAt}(\mathit{Locked}, t) \\
& \qquad \wedge \mathit{HoldsAt}(\mathit{HasKey}, t)] \\
& \vee [a = \mathit{GoThrough}\ \wedge\ f = \mathit{Inside} \\
& \qquad \wedge \neg \mathit{HoldsAt}(\mathit{Locked}, t) \\
& \qquad \wedge \neg \mathit{HoldsAt}(\mathit{Inside}, t)]]
\end{aligned} \tag{R1}
$$

$$
\begin{aligned}
\mathit{Terminates}(a,f,t) \equiv\ & [[a = \mathit{GoThrough}\ \wedge\ f = \mathit{Inside} \\
& \qquad \wedge \neg \mathit{HoldsAt}(\mathit{Locked}, t) \\
& \qquad \wedge \mathit{HoldsAt}(\mathit{Inside}, t)] \\
& \vee [a = \mathit{Insert}\ \wedge\ f = \mathit{Locked} \\
& \qquad \wedge \mathit{HoldsAt}(\mathit{Locked}, t) \\
& \qquad \wedge \mathit{HoldsAt}(\mathit{HasKey}, t)]]
\end{aligned} \tag{R2}
$$

Let us suppose that the door is locked and the robot inside at time 0, and that the robot picks up the key, unlocks the door and goes through the door at times 2, 4 and 6 respectively:

$$
\mathit{HoldsAt}(\mathit{Locked}, 0)\ \wedge\ \mathit{HoldsAt}(\mathit{Inside}, 0) \tag{R3}
$$

$$
\begin{aligned}
\mathit{Happens}(a,t) \equiv\ & [[a = \mathit{Pickup}\ \wedge\ t = 2]\ \vee \\
& [a = \mathit{Insert}\ \wedge\ t = 4]\ \vee \\
& [a = \mathit{GoThrough}\ \wedge\ t = 6]]
\end{aligned} \tag{R4}
$$

The reader is invited to check that from (EC1)-(EC6) and (R1)-(R4), together with uniqueness-of-names axioms for fluents and actions and an appropriate axiomatisation of the real numbers, it is for example possi-

[2]In this case the collection of uniqueness-of-names axioms will consist of a sentence such as $\mathit{Inside} \neq \mathit{HasKey}$ for each pair of fluent names and action names. In domains where parameterised fluents or actions are used, e.g. a $\mathit{Lower}(x)$ action to represent the act of lowering an object x meters, it might also typically include sentences such as $\mathit{Lower}(x_1) = \mathit{Lower}(x_2) \rightarrow x_1 = x_2$. The inclusion of such uniqueness-of-names axioms is not obligatory (we might for example wish to deliberately use two names to refer to the same action), but their omission will generally lead to unexpected results.

ble to deduce that the robot is no longer inside the room at time 8, i.e. $\neg HoldsAt(Inside, 8)$.

2.2 Circumscription and the Frame Problem

In Event Calculus terms, the frame problem is the problem of expressing in a succinct and elaboration tolerant way that in most cases a given action will not initiate or terminate a given fluent. The description of which actions initiate and terminate which fluents via single biconditionals (as in axioms (R1) and (R2) above), although succinct, is rather unsatisfactory from the point of view of elaboration tolerance. For example, if new information about the initiating effects of a new action needs to be included in the robot domain (e.g. $Initiates(PressDoorBell, RingingNoise, t)$) this cannot be simply added to the theory, since it would be inconsistent with axiom (R1) (from which it is possible to infer $\neg Initiates(PressDoorBell, RingingNoise, t)$).

Hence most versions of the Event Calculus describe each fact or rule about initiation and termination in a separate axiom or clause, and provide an extra transformation or non-monotonic reasoning method to infer negative information about *Initiates* and *Terminates* from the collection of such rules. In the context of our classical logic Event Calculus and robot example, the individual rules would be

$$Initiates(Pickup, HasKey, t) \tag{R5}$$

$$Initiates(Insert, Locked, t) \leftarrow [\neg HoldsAt(Locked, t) \wedge HoldsAt(HasKey, t)] \tag{R6}$$

$$Initiates(GoThrough, Inside, t) \leftarrow [\neg HoldsAt(Locked, t) \wedge \neg HoldsAt(Inside, t)] \tag{R7}$$

$$Terminates(GoThrough, Inside, t) \leftarrow [\neg HoldsAt(Locked, t) \wedge HoldsAt(Inside, t)] \tag{R8}$$

$$Terminates(Insert, Locked, t) \leftarrow [HoldsAt(Locked, t) \wedge HoldsAt(HasKey, t)] \tag{R9}$$

Predicate completion or circumscription [McCa 80] can then be used to transform this collection of axioms into expressions such as (R1) and (R2). In this article we use the notation described in [Lifs 94] to indicate circumscriptions of particular conjunctions of sentences. In particular, the circumscription

$$CIRC[(R5) \wedge (R6) \wedge (R7) \wedge (R8) \wedge (R9)\ ;\ Initiates, Terminates]$$

yields exactly (R1) and (R2). For simple domains such as the above, this type of transformation (whether described in terms of circumscription or predicate completion) is effectively the Event Calculus equivalent of the solution to the frame problem developed by Reiter for the Situation Calculus [Reit 91].

To make useful deductions using axioms (EC1)-(EC6), it is also necessary to be able to infer both positive and negative information about *Happens* from the domain dependent part of the theory. Again the issue of elaboration tolerance arises, so that, as for *Initiates* and *Terminates*, most versions of the Event Calculus encapsulate each individual action occurrence in a separate *Happens* assertion (rather than using a biconditional such as (R4)), and then use some form of non-monotonic reasoning to infer negative information about this predicate. For example, in the case of our robot example the assertions would be

$$Happens(Pickup, 2) \tag{R10}$$

$$Happens(Insert, 4) \tag{R11}$$

$$Happens(GoThrough, 6) \tag{R12}$$

The circumscription $CIRC$[(R10) $\wedge$ (R11) $\wedge$ (R12) ; *Happens*] then gives (R4).

If we now wish to add more information about *Happens*, we can do so without altering axioms (R10)-(R12) and then reapply the circumscription operator. This information need not just be in the form of ground literals – we may have less precise information about the order or timing of action occurrences. For example, we might know that the robot pressed the door bell either just before, just after or at the same time as inserting the key, in which case we could add

$$\exists t_1.[Happens(PressDoorBell, t_1) \ \wedge \ 2 < t_1 < 6] \tag{R13}$$

The circumscription $CIRC$[(R10) $\wedge$ (R11) $\wedge$ (R12) $\wedge$ (R13) ; *Happens*] then gives

$$\begin{aligned} \exists t_1.[2 < t_1 < 6 \ \wedge \ [Happens(a,t) \ \equiv \ & [[a = Pickup \ \wedge \ t = 2] \vee \\ & [a = Insert \ \wedge \ t = 4] \vee \\ & [a = GoThrough \ \wedge \ t = 6] \vee \\ & [a = PressDoorBell \ \wedge \ t = t_1]]]] \end{aligned} \tag{R14}$$

enabling us to deduce facts such as $\neg HoldsAt(Inside, 8)$ as before.

More generally, complete Event Calculus domain descriptions of this basic type are of the form

$$CIRC[\Sigma \ ; \ Initiates, Terminates] \ \wedge \ CIRC[\Delta \ ; \ Happens] \ \wedge \ \Omega \ \wedge \ \mathrm{EC}$$

where Σ is a conjunction of *Initiates* and *Terminates* formulae, Δ is a conjunction of *Happens* and temporal ordering formulae, Ω is a conjunction of fluent-specific *HoldsAt* formulae such as (R3) and time-independent formulae (such as uniqueness-of-names axioms for actions and fluents), and EC is the conjunction of axioms (EC1) to (EC6) together an appropriate axiomatisation of the sort $\mathcal{T}$. The minimisation of *Initiates* and *Terminates* corresponds to the default assumption that actions have no unexpected effects, and the minimisation of *Happens* corresponds to the default assumption that there are no unexpected event occurrences. The key to this solution to the frame problem is thus the splitting of the theory into different parts, which are circumscribed separately. This technique, sometimes referred to as *forced circumscription*, is also employed in [CrEt 92], [Dohe 94] and [KaLi 95], and is akin to what Sandewall calls *filtering* [Sand 94].

2.3 Narrative Information and Planning

In some circumstances it is convenient to define *Happens* in terms of other predicates representing different categories of action occurrence. For example, in the context of planning we may wish to distinguish between actions that have (definitely) happened in the past and actions that the agent will (possibly) perform in the future[3]. In this case we may include a domain independent axiom such as

$$Happens(a,t) \equiv [Occurred(a,t) \vee Perform(a,t)] \tag{EC7}$$

We can now maintain a complete definition for *Occurred* (in the same way that we previously had a complete definition for *Happens*) based on our knowledge of actions that have already taken place, whilst keeping *Perform* undefined within the theory. In this way we can formulate a deductive specification of the planning task in terms of *Perform*. For example, in the context of the robot example suppose that at time 3 we know that a *Pickup* action has already taken place (at time 2), and wish to plan for the goal $\neg HoldsAt(Inside, 8)$. We would include the axiom

$$Occurred(a,t) \equiv [a = Pickup \wedge t = 2] \tag{R15}$$

(or the equivalent expression $CIRC[Occurred(Pickup, 2) \; ; \; Occurred]$) in the domain description, and then show that the sentence

$$Perform(a,t) \equiv [[a = Insert \wedge t = 4] \vee [a = GoThrough \wedge t = 6]] \tag{P1}$$

is a plan for $\neg HoldsAt(Inside, 8)$ in the sense that

$$[(\mathrm{EC1}) \wedge \ldots \wedge (\mathrm{EC7}) \wedge (\mathrm{R1}) \wedge (\mathrm{R2}) \wedge (\mathrm{R3}) \wedge (\mathrm{R15})] \models [(\mathrm{P1}) \rightarrow \neg HoldsAt(Inside, 8)]$$

More generally, planning can be viewed as the deduction[4] of sentences of the form $[Plan \rightarrow Goal]$ from an Event Calculus domain description, where *Plan* is a sentence such as (P1) defining the predicate *Perform*, and *Goal* is a sentence containing just the predicates *HoldsAt* and $<$ (we need also to establish via general theorems or a specific check that *Plan* is consistent with the Event Calculus theory). By the Deduction Theorem $Theory \models [Plan \rightarrow Goal]$ is equivalent to $[Theory \wedge Plan] \models Goal$ so that planning in the context of the Event Calculus can also be understood in terms of abduction (i.e. finding plans to add to the theory so that the goal is entailed). Indeed, it is this abductive view which is taken in the majority of work on Event Calculus planning, e.g. in [Eshg 88], [Chle 96], [JuFi 96], [Shan 97b], [Shan 98] and [Jung 98].

2.4 Non-determinism

In contrast to many versions of the Event Calculus, the axiomatisation described in (EC1)–(EC6) is non-deterministic, in the sense that simultaneously initiating and terminating a fluent simply gives rise to two sets of models (one in which the fluent is true immediately afterwards and one in which it is false), rather than resulting in an inconsistent theory. This is because of the requirement in axioms (EC1) and (EC2) that $t_1 \leq t$, rather than $t_1 < t$.

[3] Or we may wish to distinguish between actions performed by the agent and events occurring in the environment and outside the agent's control.

[4] That is to say, planning can be specified as a deductive task. We do not wish to claim that general purpose classical theorem provers are practical as planning systems.

For example, let us suppose that the action of tossing a coin is represented as *TossCoin*, and that each occurrence of this action results in the fluent *HeadsUp* being either true or false. We can represent this with an *Initiates* and a *Terminates* literal:

$$Initiates(TossCoin, HeadsUp, t) \tag{C1}$$

$$Terminates(TossCoin, HeadsUp, t) \tag{C2}$$

Suppose the time is represented as the reals, and that a single *TossCoin* action happens at time 2:

$$Happens(TossCoin, 2) \tag{C3}$$

The theory which consists of axioms (EC1)-(EC6), $CIRC[(\mathrm{C1}) \wedge (\mathrm{C2}) \; ; \; Initiates, Terminates]$ and $CIRC[(\mathrm{C3}) \; ; \; Happens]$ has four classes of models with respect to the fluent *HeadsUp* – one in which *HeadsUp* holds for all timepoints, one in which *HeadsUp* holds for no timepoints, one in which *HeadsUp* changes from true to false for all timepoints greater than 2, and one in which *HeadsUp* changes from false to true for all timepoints greater than 2. This is because using axioms (EC1) and (EC2) we can show $Clipped(2, HeadsUp, T)$ and $Declipped(2, HeadsUp, T)$ for all $T > 2$, so that (EC3) and (EC4) are trivially satisfied and the truth value of *HeadsUp* at different timepoints is constrained only by axioms (EC5) and (EC6).

The narrative-based nature of the Event Calculus (i.e. the fact that action occurrences are explicitly represented) facilitates a simple alternative to representing non-determinism. We can for example regard the action *TossCoin* as representing a choice of two deterministic actions *TossHead* and *TossTail*:

$$\begin{aligned} &Happens(TossCoin, t) \rightarrow \\ &\qquad [Happens(TossHead, t) \vee Happens(TossTail, t)] \end{aligned} \tag{C4}$$

We can then rewrite axioms (C1) and (C2) as

$$Initiates(TossHead, HeadsUp, t) \tag{C1a}$$

$$Terminates(TossTail, HeadsUp, t) \tag{C2a}$$

The theory consisting of (EC1)-(EC6), (C5), $CIRC[(\mathrm{C3}) \wedge (\mathrm{C4}) \; ; \; Happens]$ and $CIRC[(\mathrm{C1a}) \wedge (\mathrm{C2a}) \; ; \; Initiates, Terminates]$ now also gives rise to the desired classes of models described above. (The circumscription of *Happens* eliminates models where both a *TossHead* and a *TossTail* action occur at time 2.) For tasks such as planning, it is straightforward to specify that the agent in question can attempt some actions (such as *TossCoin*) but not others (such as *TossHead* or *TossTail*).

2.5 Concurrent Actions

The syntax of the Event Calculus makes it straightforward to express that two or more actions have occurred or will occur simultaneously, since different *Happens* literals in the domain description may refer to the same timepoint.

In some domains, concurrently performed actions may cancel each others' effects, and may combine to cause effects which none of the actions performed in isolation would achieve. A standard example is that if a bowl is filled with water, lifting just the left side of the bowl or just the right side will cause the water to spill. Lifting both sides simultaneously will not cause the water to spill but will cause the bowl to be raised.

In the Event Calculus, we can describe cancellations and combinations of effects with *Happens* preconditions in the domain dependent axioms defining *Initiates* and *Terminates*. For example:

$$Initiates(LiftLeft, Spilt, t) \leftarrow \neg Happens(LiftRight, t) \tag{B1}$$

$$Initiates(LiftRight, Spilt, t) \leftarrow \neg Happens(LiftLeft, t) \tag{B2}$$

$$Initiates(LiftRight, Raised, t) \leftarrow Happens(LiftLeft, t) \tag{B3}$$

To illustrate the effect of such statements, suppose that our domain description also includes the following narrative information:

$$\neg HoldsAt(Spilt, 0) \tag{B4}$$

$$Happens(LiftLeft, 2) \tag{B5}$$

$$Happens(LiftRight, 2) \tag{B6}$$

The theory consisting of (EC1)-(EC6), (B4), $CIRC[(B1) \wedge (B2) \wedge (B3)\ ;\ Initiates, Terminates]$ and $CIRC[(B5) \wedge (B6)\ ;\ Happens]$ entails, for example, both $\neg HoldsAt(Spilt, 4)$ and $HoldsAt(Raised, 4)$.

3 Alternative and Extended Event Calculus Axiomatisations

The version of the Event Calculus described in Section 2 has a number of characteristics; it is geared to time-lines extending infinitely backwards as well as forwards, it is "non-deterministic" (in the sense described in Section 2.4), it regards all actions as possible under all circumstances, it regards all fluents' truth values as persisting between all relevant action occurrences, and it regards all action occurrences as instantaneous. However, the choice of which of these characteristics to include in a given Event Calculus axiomatisation is to a large extent arbitrary, and in this section we describe alternative axiomatisations which each negate one or more of these properties.

For ease of presentation, sub-sections 3.1 to 3.7 below each alter the axiomatisation (EC1)–(EC6) in as minimal a way as possible to illustrate the particular point under discussion. But unless otherwise stated these alterations can be combined in a straightforward and obvious manner. For example we can combine the modifications described in sub-sections 3.2, 3.3 and 3.6 below to produce a "deterministic" Event Calculus with facilities to describe when it is impossible for particular actions to occur, and including actions of a non-zero duration.

Where in a particular sub-section no alternative to one of the axioms (EC1)–(EC6) is given, it should be assumed that the axiom in question remains unchanged.

3.1 An Alternative Axiomatisation for Non-Negative Time

Where time is modeled as the non-negative reals or integers, it is often convenient to introduce two new predicates[5] $InitiallyP \subseteq \mathcal{F}$ and $InitiallyN \subseteq \mathcal{F}$ ("P" for "positive" and "N" for "negative"), and to replace axioms (EC5) and (EC6) with the following three axioms:

$$HoldsAt(f,t) \leftarrow [InitiallyP(f) \wedge \neg Clipped(0,f,t)] \qquad \text{(EC5a)}$$

$$\neg HoldsAt(f,t) \leftarrow [InitiallyN(f) \wedge \neg Declipped(0,f,t)] \qquad \text{(EC6a)}$$

$$InitiallyP(f) \vee InitiallyN(f) \qquad \text{(EC8a)}$$

Indeed, for non-negative time (EC5a), (EC6a) and (EC8a) may be deduced from (EC5) and (EC6) together with the assertion

$$[HoldsAt(f,0) \equiv InitiallyP(f)] \wedge [\neg HoldsAt(f,0) \equiv InitiallyN(f)]$$

Axioms (EC5a) and (EC6a) have an advantage over (EC5) and (EC6) in that they can readily be converted to logic program clauses without causing obvious looping problems.

However, this alternative axiomatisation is slightly weaker. For example, in the non-deterministic domain described in Sub-section 2.4 by axioms (C1), (C2) and (C3), axioms (EC5a), (EC6a) and (EC8) would lisense models where *HeadsUp* fluctuated arbitrarily between true and false at times after 2. Although this characteristic is problematic for this particular example, it can be an advantage for representing other types of domain where it is convenient to "dynamically manage the frame", i.e. to regard some fluents as having an inherent persistence during some intervals of time but not during others. Indeed, for such domains axiom (EC8a) may not be appropriate. These issues are discussed in more detail in Section 3.7.

Since this particular axiomatisation does not include the general principle of persistence encapsulated in (EC5) and (EC6) (which describes how fluents persist independently of initiating and terminating action occurrences), adding individual *HoldsAt* literals to a given domain description (see for example axiom (R3) in Section 2.1) no longer necessarily has the same effect, particularly in axiomatisations where (EC8a) is omitted. Instead, individual observations of the form $HoldsAt(F,T)$ and $\neg HoldsAt(F,T)$ can be assimilated indirectly into the theory (perhaps automatically by a process of abduction) by appropriate addition of *Happens*, *InitiallyP* and *InitiallyN* literals. Axiom (R3), for example, can be replaced by

$$InitiallyP(Locked) \wedge InitiallyP(Inside) \qquad \text{(R3a)}$$

so that (R3) is now entailed by the theory consisting of (R3a), (R1), (R2), (R4), (EC1)–(EC4), (EC5a) and (EC6a).

3.2 Deterministic Event Calculus

A strictly deterministic Event Calculus (in the sense that simultaneously initiating and terminating a fluent results in inconsistency) may be

[5] These predicates are referred to as *InitiallyTrue* and *InitiallyFalse* in [MiSh 96].

formulated by replacing (EC3) and (EC4) by the following two axioms:

$$HoldsAt(f,t_2) \leftarrow [Happens(a,t_1) \wedge Initiates(a,f,t_1) \wedge t_1 < t_2 \wedge \neg StoppedIn(t_1,f,t_2)] \quad \text{(EC3b)}$$

$$\neg HoldsAt(f,t_2) \leftarrow [Happens(a,t_1) \wedge Terminates(a,f,t_1) \wedge t_1 < t_2 \wedge \neg StartedIn(t_1,f,t_2)] \quad \text{(EC4b)}$$

where the predicates *StoppedIn* and *StartedIn* are defined as follows:

$$StoppedIn(t_1,f,t_2) \stackrel{\text{def}}{\equiv} \exists a,t[Happens(a,t) \wedge t_1 < t < t_2 \wedge Terminates(a,f,t)] \quad \text{(EC9b)}$$

$$StartedIn(t_1,f,t_2) \stackrel{\text{def}}{\equiv} \exists a,t[Happens(a,t) \wedge t_1 < t < t_2 \wedge Initiates(a,f,t)] \quad \text{(EC10b)}$$

Note that *StoppedIn* and *StartedIn* are identical to *Clipped* and *Declipped* except for the inequality relations between the time-point variables. Strictly speaking (EC1) and (EC2) defining *Clipped* and *Declipped* are still required since these predicates are used in (EC5) and (EC6), or in their substitutes (EC5a) and (EC6a). But for domains using non-negative time and axioms (EC5a) and (EC6a), the definitions of *Clipped* and *Declipped* may be straightforwardly replaced by those for *StoppedIn* and *StartedIn*, provided no actions occur at time 0 which either terminate initially-positive fluents or initiate initially-negative fluents.

The effective meanings of *Initiates* and *Terminates* are slightly different in the deterministic Event Calculus (i.e. in axiomatisations including (EC3b) and (EC4b)) from their meanings in non-deterministic Event Calculus. (EC3b) and (EC4b) ensure that $Initiates(A,F,T)$ can be read as "F holds immediately after an occurrence of A at time T", whereas with axioms (EC3) and (EC4) $Initiates(A,F,T)$ corresponds to the slightly weaker assertion that "an occurrence of A at time T has an initiating influence on F" (which may or may not be overridden by a simultaneously occurring terminating influence).

There are several ways in which non-determinism may be reintroduced into what we have described here as deterministic Event Calculus. For example, the technique exemplified in axioms (C1a), (C2a) and (C4) (see Section 2.4) is still applicable. Other methods include the use of *determining fluents* or a *Releases* predicate (see [Shan 97a] and Section 3.7).

3.3 Action Preconditions and the Qualification Problem in the Event Calculus

We have already illustrated with axioms such as (R6)-(R9) (See Section 2.2) how preconditions for particular effects of actions may be expressed within the Event Calculus. These types of precondition are often referred to as *fluent preconditions.* There are also various ways in which *action preconditions* (i.e. conditions necessary for actions to be possible at all) can be expressed. One method is to introduce a new predicate $Impossible \subseteq \mathcal{A} \times \mathcal{T}$ and write an appropriate definition for *Impossible* with respect to each action in the domain in question. For instance, in our example domain we may wish to express that it is impossible for the robot to pickup the key if it is not fitted with a grabber, and it is impossible for the robot to go through a locked door:

$$Impossible(Pickup, t) \leftarrow \neg HoldsAt(HasGrabber, t) \tag{R16}$$

$$Impossible(GoThrough, t) \leftarrow HoldsAt(Locked, t) \tag{R17}$$

We can regard the qualification problem (at least in part) as the problem of expressing, in a succinct and elaboration tolerant way, that under most circumstances most actions are possible. To achieve this in the Event Calculus, we can minimise the predicate *Impossible*. $CIRC[Impossible\ ;\ (\text{R16}) \wedge (\text{R17})]$ gives

$$\begin{aligned} Impossible(a, t) \equiv\ & [[a = Pickup \wedge \neg HoldsAt(HasGrabber, t)] \\ & \vee [a = GoThrough \wedge HoldsAt(Locked, t)]] \end{aligned} \tag{R18}$$

For narrative formalisms such as the Event Calculus, the way in which this type of knowledge is to be interpreted, and thus the way in which the domain independent axioms need to be adapted, depends to some extent on the individual domain and mode of reasoning under consideration. For tasks such as planning, which involves (hypothetical) reasoning about future events, it makes sense to regard the assertion $Impossible(A, T)$ as stating "it is impossible to predict the effects of attempting to perform action A at time T" (so that $\neg Impossible(A, T)$ can be regarded as analogous to $Poss(A, S)$ in Reiter's Situation Calculus [Reit 91]). In this case it is necessary only to block any inferences about what holds or does not hold at any time after an (attempt at an) 'impossible' action occurrence. This can be done by appropriately modifying the definitions of *Clipped* and *Declipped*:

$$\begin{aligned} & Clipped(t_1, f, t_2) \stackrel{\text{def}}{\equiv} \\ & \quad [\exists a, t[Happens(a, t) \wedge t_1 \leq t < t_2 \wedge Terminates(a, f, t)] \\ & \quad \vee \exists a, t[Happens(a, t) \wedge t < t_2 \wedge Impossible(a, t)]] \end{aligned} \tag{EC1c}$$

$$\begin{aligned} & Declipped(t_1, f, t_2) \stackrel{\text{def}}{\equiv} \\ & \quad [\exists a, t[Happens(a, t) \wedge t_1 \leq t < t_2 \wedge Initiates(a, f, t)] \\ & \quad \vee \exists a, t[Happens(a, t) \wedge t < t_2 \wedge Impossible(a, t)]] \end{aligned} \tag{EC2c}$$

On the other hand, if for example the domain includes certain knowledge about actions or events that have actually occurred in the past, it makes sense to regard the assertion $Impossible(A, T)$ as stating "action A could not have occurred at time T". Hence where the definition of *Happens* is split as in axiom (EC7) (see Section 2.3), we can include additional constraints such as

$$\neg Occurred(a, t) \leftarrow Impossible(a, t) \tag{EC11c}$$

Notice for example that from (R15), (R16) and (EC11c) we can infer the action precondition $HoldsAt(HasGrabber, 2)$ for the known occurrence of *Pickup* at time 2. This illustrates why we would not want to include constraints analogous to (EC11c) for hypothetical future performances of actions – at time 0 we would not for example want $Perform(Pickup, 1)$ to constitute a plan for the goal $HoldsAt(HasGrabber, 1)$. We could however safely state that nothing happens when an agent attempts to perform an impossible action, by replacing (EC7) with

$$Happens(a,t) \equiv [[Perform(a,t) \wedge \neg Impossible(a,t)] \vee Occurred(a,t)] \quad \text{(EC7c)}$$

Finally, note that rules such as (R16) and (R17) partially defining *Impossible* can have *Happens* (and *Perform* and *Occurred*) preconditions as well as *HoldsAt* preconditions. This can be useful, for example, for expressing that it is impossible to perform certain combinations of actions simultaneously. For instance, the sentence

$$Impossible(a_1,t) \leftarrow [Perform(a_2,t) \wedge a_1 \neq a_2]$$

states that it is in general impossible to perform more than one action at a time. Like (R16) and (R17), such sentences must be placed within the scope of the circumscription of *Impossible*.

3.4 Categorisation of Fluents in the Event Calculus

For some domains, it is appropriate to categorise fluents into *frame* fluents and *non-frame* fluents (or *primitive* and *derived* fluents), and then to restrict the application of the principles of persistence encapsulated in axioms (EC3)-(EC6) to frame fluents only. To do this it is necessary to introduce a new predicate $Frame \subseteq \mathcal{F}$, and alter (EC3)-(EC6) as follows:

$$HoldsAt(f,t_2) \leftarrow [Happens(a,t_1) \wedge Initiates(a,f,t_1) \wedge \neg Frame(f) \wedge t_1 < t_2 \wedge \neg Clipped(t_1,f,t_2)] \quad \text{(EC3d)}$$

$$\neg HoldsAt(f,t_2) \leftarrow [Happens(a,t_1) \wedge Terminates(a,f,t_1) \wedge \neg Frame(f) \wedge t_1 < t_2 \wedge \neg Declipped(t_1,f,t_2)] \quad \text{(EC4d)}$$

$$HoldsAt(f,t_2) \leftarrow [HoldsAt(f,t_1) \wedge t_1 < t_2 \wedge \neg Frame(f) \wedge \neg Clipped(t_1,f,t_2)] \quad \text{(EC5d)}$$

$$\neg HoldsAt(f,t_2) \leftarrow [\neg HoldsAt(f,t_1) \wedge t_1 < t_2 \wedge \neg Frame(f) \wedge \neg Declipped(t_1,f,t_2)] \quad \text{(EC6d)}$$

This axiom set can be useful when we want to include simple types of indirect effects in domain descriptions, since we are now free to write definitions or partial definitions of non-frame fluents (i.e. *state constraints*) in terms of *HoldsAt* and frame fluents. For example, as regards the robot we may wish to introduce a non-frame fluent *Happy* and state that, although the robot is not happy at time 0, it is in general happy if it is holding the key:

$$Frame(f) \equiv [f = Inside \vee f = HasKey \vee f = Locked] \quad \text{(R19)}$$

$$\neg HoldsAt(Happy, 0) \quad \text{(R20)}$$

$$HoldsAt(Happy,t) \leftarrow HoldsAt(HasKey,t) \quad \text{(R21)}$$

Using (EC1), (EC2), (EC3d)-(EC6d), (R1)-(R4) and (R19)-(R21) we can now, for example, infer $HoldsAt(Happy, 5)$. Indeed, we can also infer $\neg HoldsAt(HasKey, 0)$ and therefore $\neg HoldsAt(HasKey, 1)$. But we can neither infer $HoldsAt(Happy, 1)$ nor $\neg HoldsAt(Happy, 1)$, since the non-frame fluent *Happy* has no intrinsic persistence of its own.

3.5 Trajectories, Delayed Actions and Gradual Change

Several techniques are available within the context of the Event Calculus for describing delayed effects. The simplest approach is to write rules in terms of *Happens*. For example, if setting an alarm clock causes it to ring 8 hours later, we can write

$$\mathit{Happens}(\mathit{StartRing}, t+8) \leftarrow \mathit{Happens}(\mathit{Set}, t) \tag{A1}$$

$$\mathit{Initiates}(\mathit{StartRing}, \mathit{Ringing}, t) \tag{A2}$$

A disadvantage of rules such as (A1) is that it is difficult to express that the occurrence of the later action might be prevented by some intervening action (e.g. somebody might switch off the alarm during the night).

A more flexible approach involves the use of *trajectories* [Shan 90]. It is convenient to illustrate this technique here by introducing a new sort $\mathcal{P}$ of *parameters* into the language. Like fluents, parameters are time-varying properties, but unlike (frame) fluents they have no associated default persistence. More precisely, parameters are names for arbitrarily-valued functions of time, and accordingly we introduce a new function $\mathit{ValueAt} : \mathcal{P} \times \mathcal{T} \mapsto \mathcal{X}$. For example, we might write $\mathit{ValueAt}(\mathit{Countdown}, 5) = 2$ to indicate that at time 5 the parameter *Countdown*, representing the time remaining before the alarm clock rings, has a value of 2. To represent delayed and triggered effects, as well as simple forms of gradual or continuous change, specific parameters are associated with specific fluents via the predicate $\mathit{Trajectory} \subseteq \mathcal{F} \times \mathcal{T} \times \mathcal{P} \times \mathcal{T} \times \mathcal{X}$. The intended meaning of $\mathit{Trajectory}(F, T_1, P, T_2, X)$ is that if fluent F is initiated at time T_1 and continues to hold until time T_1+T_2, this results in parameter P having a value of X at time T_1+T_2. For example, in the case of the alarm clock we might write

$$\mathit{Trajectory}(\mathit{SwitchedOn}, t_1, \mathit{Countdown}, t_2, 8-t_2) \tag{A3}$$

We can translate this intended meaning into Event Calculus terms with the addition of a single extra domain independent axiom

$$\begin{aligned} \mathit{ValueAt}(p, t_1+t_2) = x \leftarrow \\ & [\mathit{Happens}(a, t_1) \wedge \mathit{Initiates}(a, f, t_1) \\ & \wedge\ 0 < t_2 \wedge \mathit{Trajectory}(f, t_1, p, t_2, x) \\ & \wedge \neg \mathit{Clipped}(t_1, f, t_1+t_2)] \end{aligned} \tag{EC11}$$

Continuing with our example, it is straightforward to express that when *Countdown* reaches 0 the alarm goes off:

$$\mathit{Happens}(\mathit{StartRing}, t) \leftarrow \mathit{ValueAt}(\mathit{Countdown}, t) = 0 \tag{A4}$$

We can complete our description of the domain by stating that switching on the alarm activates the timing mechanism (provided it is not already activated), that the ringing event switches off the timing mechanism, that when the timing mechanism is switched off the countdown is permanently fixed at 8, that the alarm is initially not switched on and that someone switches it on at time 2:

$$\mathit{Initiates}(\mathit{Set}, \mathit{SwitchedOn}, t) \leftarrow \neg \mathit{HoldsAt}(\mathit{SwitchedOn}, t) \tag{A5}$$

$$\mathit{Terminates}(\mathit{StartRing}, \mathit{SwitchedOn}, t) \tag{A6}$$

$$ValueAt(Countdown, t) = 8 \leftarrow \neg HoldsAt(SwitchedOn, t) \quad \text{(A7)}$$

$$\neg HoldsAt(SwitchedOn, 0) \quad \text{(A8)}$$

$$Happens(Set, 2) \quad \text{(A9)}$$

The theory consisting of (EC1)-(EC6), (EC11), (A3), (A7), (A8), $CIRC$[(A4) $\wedge$ (A9) ; $Happens$] and $CIRC$[(A2) $\wedge$ (A5) $\wedge$ (A6) ; $Initiates$, $Terminates$] entails, for example, $Happens(StartRing, 10)$ and $HoldsAt(Ringing, 11)$.

The Event Calculus is symmetric as regards positive and negative $HoldsAt$ literals and as regards $Initiates$ and $Terminates$. Hence (EC11) has its counterpart in terms of $Terminates$:

$$\begin{aligned} ValueAt(p, t_1 + t_2) = x \leftarrow & \\ & [Happens(a, t_1) \wedge Terminates(a, f, t_1) \\ & \wedge\ 0 < t_2 \wedge AntiTrajectory(f, t_1, p, t_2, x) \\ & \wedge \neg Declipped(t_1, f, t_1 + t_2)] \end{aligned} \quad \text{(EC12)}$$

This axiom uses the predicate $AntiTrajectory \subseteq \mathcal{F} \times \mathcal{T} \times \mathcal{P} \times \mathcal{T} \times \mathcal{X}$. The intended meaning of $AntiTrajectory(F, T_1, P, T_2, X)$ is that if fluent F is terminated at time T_1 and continues not to hold until time $T_1 + T_2$, this results in parameter P having a value of X at time $T_1 + T_2$. We can illustrate the use of anti-trajectories by representing the fact that a hot-air balloon rises when the air-heater is on, but falls when it is not:

$$\begin{aligned} & Trajectory(HeaterOn, t_1, Height, t_2, x_1 + t_2) \\ & \qquad \leftarrow ValueAt(Height, t_1) = x_1 \end{aligned} \quad \text{(H1)}$$

$$\begin{aligned} & AntiTrajectory(HeaterOn, t_1, Height, t_2, x_1 - t_2) \\ & \qquad \leftarrow ValueAt(Height, t_1) = x_1 \end{aligned} \quad \text{(H2)}$$

(Note that in the alarm clock example (A7) can also be expressed as $AntiTrajectory(SwitchedOn, t_1, Countdown, t_2, 8)$.)

Note that the functions captured in individual trajectories need not be continuous or even numerically valued. For example, we can use a trajectory to model the fact that the left indicator light of a car flashes once per second while the indicator switch is depressed:

$$Trajectory(IndicatorDepressed, t_1, Light, t_2, BlinkFunction(t_2)) \quad \text{(L1)}$$

$$AntiTrajectory(IndicatorDepressed, t_1, Light, t_2, Off) \quad \text{(L2)}$$

$$BlinkFunction(t) = On \leftarrow [t \ mod \ 2 < 1] \quad \text{(L3)}$$

$$BlinkFunction(t) = Off \leftarrow [t \ mod \ 2 \geq 1] \quad \text{(L4)}$$

In domains which include non-deterministic actions (in the sense that actions or combinations of actions can simultaneously initiate and terminate fluents) axioms (EC11) and (EC12) are too weak. For example, if the switching on mechanism is faulty in our alarm clock example, so that we have both (A5) and

$$Terminates(Set, SwitchedOn, t) \quad \text{(A10e)}$$

axiom (EC11) will not inform us that the countdown is activated even in the circumstance where fluent *SwitchedOn* holds immediately after time 2. One solution is to replace (EC11) and (EC12) with equivalent axioms which have an extra *HoldsAt* condition in their right-hand sides, but use *StoppedIn* and *StartedIn* (see axioms (EC9b) and (EC10b)) instead of *Clipped* and *Declipped*:

$$\begin{aligned}&\mathit{ValueAt}(p, t_1+t_2)=x \leftarrow \\ &\qquad [\mathit{Happens}(a, t_1) \wedge \mathit{Initiates}(a, f, t_1) \\ &\qquad \wedge\ 0<t_2 \wedge \mathit{Trajectory}(f, t_1, p, t_2, x) \\ &\qquad \wedge\ \mathit{HoldsAt}(f, t_1+t_2) \wedge \neg \mathit{StoppedIn}(t_1, f, t_1+t_2)]\end{aligned} \tag{EC11e}$$

$$\begin{aligned}&\mathit{ValueAt}(p, t_1+t_2)=x \leftarrow \\ &\qquad [\mathit{Happens}(a, t_1) \wedge \mathit{Terminates}(a, f, t_1) \\ &\qquad \wedge\ 0<t_2 \wedge \mathit{AntiTrajectory}(f, t_1, p, t_2, x) \\ &\qquad \wedge\ \neg\mathit{HoldsAt}(f, t_1+t_2) \wedge \neg \mathit{StartedIn}(t_1, f, t_1+t_2)]\end{aligned} \tag{EC12e}$$

In Event Calculus axiomatisations where a distinction is made between fluents which are (temporarily or permanently) inside or outside the frame (such as in Section 3.4), we may dispense with the extra sort $\mathcal{P}$ in favour of non-frame fluents, and replace (EC11) and (EC12) with axioms such as

$$\begin{aligned}\mathit{HoldsAt}(f_2, t_1+t_2) \leftarrow\ &[\neg \mathit{Frame}(f_2) \wedge \mathit{Happens}(a, t_1) \\ &\wedge \mathit{Initiates}(a, f_1, t_1) \wedge 0<t_2 \\ &\wedge \mathit{Trajectory}(f_1, t_1, f_2, t_2) \\ &\wedge \neg \mathit{Clipped}(t_1, f_1, t_1+t_2)]\end{aligned} \tag{EC11f}$$

$$\begin{aligned}\mathit{HoldsAt}(f_2, t_1+t_2) \leftarrow\ &[\neg \mathit{Frame}(f_2) \wedge \mathit{Happens}(a, t_1) \\ &\wedge \mathit{Terminates}(a, f_1, t_1) \wedge 0<t_2 \\ &\wedge \mathit{AntiTrajectory}(f_1, t_1, f_2, t_2) \\ &\wedge \neg \mathit{Declipped}(t_1, f_1, t_1+t_2)]\end{aligned} \tag{EC12f}$$

Here $\mathit{Trajectory} \subseteq \mathcal{F} \times \mathcal{T} \times \mathcal{F} \times \mathcal{T}$, and the intended meaning of $\mathit{Trajectory}(F_1, T_1, F_2, T_2)$ is that if fluent F_1 is initiated at time T_1 and continues to hold until time T_1+T_2, this results in F_2 holding at time T_1+T_2 (similarly for *AntiTrajectory*). In the alarm clock example *Countdown* would then be parameterised, (A3), (A4) and (A6) would be written

$$\mathit{Trajectory}(\mathit{SwitchedOn}, t_1, \mathit{Countdown}(8-t_2), t_2) \tag{A3f}$$

$$\mathit{Happens}(\mathit{StartRing}, t) \leftarrow \mathit{HoldsAt}(\mathit{Countdown}(0), t) \tag{A4f}$$

$$\mathit{HoldsAt}(\mathit{Countdown}(8), t) \leftarrow \neg \mathit{HoldsAt}(\mathit{SwitchedOn}, t) \tag{A7f}$$

and the domain description would include the additional constraint

$$\begin{aligned}&[\mathit{HoldsAt}(\mathit{Countdown}(x_1), t) \wedge \mathit{HoldsAt}(\mathit{Countdown}(x_2), t)] \\ &\qquad \rightarrow x_1 = x_2\end{aligned} \tag{A10f}$$

3.6 The Event Calculus and Actions with Duration

The Event Calculus can be modified in various ways so that actions can be represented as occurring over intervals of time. To illustrate, we present here a simple modification in which actions are assigned a numerical duration using the function $Dur : \mathcal{A} \mapsto \mathcal{T}$. This avoids the need to introduce extra arguments of sort $\mathcal{T}$ in the predicates *Happens*, *Initiates* and *Terminates*. For example, we will interpret the assertion $Happens(A,T)$ to mean "the action A starts to occur at T" (so that it finishes at $T+Dur(A)$).

We will be cautious in the assumptions we make about the effects of actions. We will assume that actions may affect relevant fluents from the moment they start, but the effects only become certain after the actions have finished. Hence the values of affected fluents should be undetermined by the axiomatisation during action occurrences. To incorporate these assumptions in the domain independent axioms (EC1)-(EC6) it is necessary only to modify the various inequality relations between the timepoint variables in (EC1)-(EC4):

$$\begin{aligned} Clipped(t_1,f,t_2) \stackrel{\text{def}}{\equiv} \exists a,t[&Happens(a,t) \wedge t_1 \leq (t+Dur(a)) \\ &\wedge t<t_2 \wedge Terminates(a,f,t)] \end{aligned} \tag{EC1g}$$

$$\begin{aligned} Declipped(t_1,f,t_2) \stackrel{\text{def}}{\equiv} \exists a,t[&Happens(a,t) \wedge t_1 \leq (t+Dur(a)) \\ &\wedge t<t_2 \wedge Initiates(a,f,t)] \end{aligned} \tag{EC2g}$$

$$\begin{aligned} HoldsAt(f,t_2) \leftarrow [&Happens(a,t_1) \wedge Initiates(a,f,t_1) \\ &\wedge (t_1+Dur(a))<t_2 \\ &\wedge \neg Clipped(t_1,f,t_2)] \end{aligned} \tag{EC3g}$$

$$\begin{aligned} \neg HoldsAt(f,t_2) \leftarrow [&Happens(a,t_1) \wedge Terminates(a,f,t_1) \\ &\wedge (t_1+Dur(a))<t_2 \\ &\wedge \neg Declipped(t_1,f,t_2)] \end{aligned} \tag{EC4g}$$

The issue of preconditions becomes more complex when actions have duration. We may for example wish to make a distinction between preconditions which must hold at the start of the action and those which must hold throughout the action. It is therefore often convenient to define auxiliary predicates such as $HoldsIn \subseteq \mathcal{F} \times \mathcal{T} \times \mathcal{T}$:

$$HoldsIn(f,t_1,t_3) \stackrel{\text{def}}{\equiv} \forall t_2[t_1 \leq t_2 \leq t_3 \rightarrow HoldsAt(f,t_2)] \tag{EC13}$$

To illustrate the use of *HoldsIn*, consider a simple description of an automated train which can move at a fixed speed S along a track running from West to East, provided its motor is engaged. Using the action term $MoveEast(T)$ to represent the action of moving east for T time units, we can for example write axioms such as

$$Dur(MoveEast(t))=t \tag{T1}$$

$$\begin{aligned} &Initiates(MoveEast(t), Location(x_2), t_1) \leftarrow \\ &\quad [HoldsAt(Location(x_1), t_1) \\ &\quad \wedge x_2=(x_1+S \times Dur(MoveEast(t))) \\ &\quad \wedge HoldsIn(MotorEngaged, t_1, (t_1+Dur(MoveEast(t))))] \end{aligned} \tag{T2}$$

An alternative way of dealing with actions with duration is to split them into an (instantaneous) "start of action" (e.g. *StartMoveEast*), an "end of action" (e.g. *StopMoveEast*) and introduce an extra fluent representing the fact that the action is taking place (e.g. *MovingEast*). This approach is more easily integrated with the mechanisms described in Section 3.5 for dealing with gradual change, and allows straightforward description of interruptions of partly executed actions.

3.7 Dynamic Management of the Frame

We have already seen in Sections 3.4 and 3.5 how it can sometimes be advantageous to regard some fluents ("frame" fluents) as having an intrinsic (default) persistence, but regard other fluents as liable to change truth values between action occurrences. It can also be useful to be able to express that particular fluents have a default persistence during some intervals of time but not during others. This can, for example, help succinctly describe domains involving non-determinism, continuous change and indirect effects of actions (see [Shan 97a] for details). In this section we illustrate how this facility for "dynamic management of the frame" can be incorporated into the Event Calculus by use of a new predicate $Releases \subseteq \mathcal{A} \times \mathcal{F} \times \mathcal{T}$. A form of this predicate was first introduced in [KaLi 95] and it is related to Sandewall's idea of *occlusion* [Sand 94].

$Releases(A, F, T)$ expresses that if A occurs at T it will disable the fluent F's innate persistence. The truth value of F will then be free to fluctuate until the next action occurrence which initiates or terminates it. *Releases* is defined in the domain-dependent part of the theory and circumscribed in parallel with *Initiates* and *Terminates*. For example, in the alarm clock example of Section 3.5, we may write

$$Releases(Set, Countdown, t)$$

and if this is the only such statement in our theory, the circumscription will then give

$$Releases(a, f, t) \equiv [a = Set \wedge f = Countdown]$$

$Initiates(A, F, T)$ (respectively $Terminates(A, F, T)$) now expresses that if A occurs at T it will both initiate (respectively terminate) the fluent F and enable F's innate persistence. At any given time-point, therefore, a fluent can be in one of four states – true and persisting, false and persisting, true and released or false and released. To describe these states explicitly, we introduce a predicate $ReleasedAt \subseteq \mathcal{F} \times \mathcal{T}$ analogous to $HoldsAt$. Finally we need two new auxiliary predicates $ReleasedBetween \subseteq \mathcal{T} \times \mathcal{F} \times \mathcal{T}$ and $PersistsBetween \subseteq \mathcal{T} \times \mathcal{F} \times \mathcal{T}$. $ReleasedBetween(T_1, F, T_2)$ means "an action releases the fluent F between times T_1 and T_2" and $PersistsBetween(T_1, F, T_2)$ means "the fluent is not in a state of release at any time between T_1 and T_2."

The Event Calculus described in (EC1)–(EC6) needs fairly radical modifications to incorporate these extra concepts and predicates. The modified axiomatisation is as follows (for ease of reading (EC1) and (EC2) are listed again, although they are unmodified). The first three axioms are all similar and give definitions for *Clipped*, *Declipped* and *ReleasedBetween*:

$$Clipped(t_1, f, t_2) \stackrel{\text{def}}{\equiv} \exists a, t[Happens(a, t) \wedge t_1 \leq t < t_2 \wedge Terminates(a, f, t)] \quad \text{(EC1)}$$

$$Declipped(t_1, f, t_2) \stackrel{\text{def}}{\equiv} \exists a, t[Happens(a, t) \wedge t_1 \leq t < t_2 \wedge Initiates(a, f, t)] \quad \text{(EC2)}$$

$$ReleasedBetween(t_1, f, t_2) \stackrel{\text{def}}{\equiv} \exists a, t[Happens(a, t) \wedge t_1 \leq t < t_2 \wedge Releases(a, f, t)] \quad \text{(EC14h)}$$

The next four axioms indicate how particular actions can put a fluent in one of the four states described above:

$$HoldsAt(f, t_2) \leftarrow [Happens(a, t_1) \wedge Initiates(a, f, t_1) \wedge t_1 < t_2 \wedge \neg Clipped(t_1, f, t_2) \wedge \neg ReleasedBetween(t_1, f, t_2)] \quad \text{(EC3h)}$$

$$\neg HoldsAt(f, t_2) \leftarrow [Happens(a, t_1) \wedge Terminates(a, f, t_1) \wedge t_1 < t_2 \wedge \neg Declipped(t_1, f, t_2) \wedge \neg ReleasedBetween(t_1, f, t_2)] \quad \text{(EC4h)}$$

$$ReleasedAt(f, t_2) \leftarrow [Happens(a, t_1) \wedge Releases(a, f, t_1) \wedge t_1 < t_2 \wedge \neg Clipped(t_1, f, t_2) \wedge \neg Declipped(t_1, f, t_2)] \quad \text{(EC15h)}$$

$$\neg ReleasedAt(f, t_2) \leftarrow [Happens(a, t_1) \wedge t_1 < t_2 \wedge [Initiates(a, f, t_1) \vee Terminates(a, f, t_1)] \wedge \neg ReleasedBetween(t_1, f, t_2)] \quad \text{(EC16h)}$$

A weakened version of the "commonsense law of inertia" is captured in the following three axioms:

$$PersistsBetween(t_1, f, t_2) \stackrel{\text{def}}{\equiv} \neg\exists t[ReleasedAt(f, t) \wedge t_1 \leq t \leq t_2] \quad \text{(EC17h)}$$

$$HoldsAt(f, t_2) \leftarrow [HoldsAt(f, t_1) \wedge t_1 < t_2 \wedge PersistsBetween(t_1, f, t_2) \wedge \neg Clipped(t_1, f, t_2)] \quad \text{(EC5h)}$$

$$\neg HoldsAt(f, t_2) \leftarrow [\neg HoldsAt(f, t_1) \wedge t_1 < t_2 \wedge PersistsBetween(t_1, f, t_2) \wedge \neg Declipped(t_1, f, t_2)] \quad \text{(EC6h)}$$

Finally, we need to state that the meta-property of being "released" is itself subject to a form of meta-persistence between action occurrences:

$$ReleasedAt(f, t_2) \leftarrow [ReleasedAt(f, t_1) \wedge t_1 < t_2 \wedge \neg Clipped(t_1, f, t_2) \wedge \neg Declipped(t_1, f, t_2)] \quad \text{(EC18h)}$$

$$\neg ReleasedAt(f, t_2) \leftarrow [\neg ReleasedAt(f, t_1) \wedge t_1 < t_2 \wedge \neg ReleasedBetween(t_1, f, t_2)] \quad \text{(EC19h)}$$

Individual *ReleasedAt* literals can be included in the domain dependent part of the theory in the same way as *HoldsAt* literals (see for example axiom (R3) in Section 2.1).

The above axiomatisation is fairly complex – it replaces our original six axioms with twelve (longer) ones and introduces four new predicates. However, for practical and computational purposes (e.g. ease of translation into logic programs) and where we are using non-negative time, we can dispense with the predicates *ReleasedAt*, *ReleasedBetween* and *PersistsBetween* and simply incorporate *Releases* in the definitions of *Clipped* and *Declipped*. This gives rise to the following alternative (and complete) set of domain independent axioms:

$$\begin{aligned} & \mathit{Clipped}(t_1, f, t_2) \stackrel{\text{def}}{\equiv} \\ & \quad \exists a, t[\mathit{Happens}(a, t) \ \wedge \ t_1 \leq t < t_2 \\ & \quad\quad \wedge \ [\mathit{Terminates}(a, f, t) \ \vee \ \mathit{Releases}(a, f, t)]] \end{aligned} \tag{EC1i}$$

$$\begin{aligned} & \mathit{Declipped}(t_1, f, t_2) \stackrel{\text{def}}{\equiv} \\ & \quad \exists a, t[\mathit{Happens}(a, t) \ \wedge \ t_1 \leq t < t_2 \\ & \quad\quad \wedge \ [\mathit{Initiates}(a, f, t) \ \vee \ \mathit{Releases}(a, f, t)]] \end{aligned} \tag{EC2i}$$

$$\begin{aligned} \mathit{HoldsAt}(f, t_2) \ \leftarrow \ & [\mathit{Happens}(a, t_1) \ \wedge \ \mathit{Initiates}(a, f, t_1) \\ & \wedge \ t_1 < t_2 \ \wedge \ \neg \mathit{Clipped}(t_1, f, t_2)] \end{aligned} \tag{EC3}$$

$$\begin{aligned} \neg \mathit{HoldsAt}(f, t_2) \ \leftarrow \ & [\mathit{Happens}(a, t_1) \ \wedge \ \mathit{Terminates}(a, f, t_1) \\ & \wedge \ t_1 < t_2 \ \wedge \ \neg \mathit{Declipped}(t_1, f, t_2)] \end{aligned} \tag{EC4}$$

$$\mathit{HoldsAt}(f, t) \ \leftarrow \ [\mathit{InitiallyP}(f) \ \wedge \ \neg \mathit{Clipped}(0, f, t)] \tag{EC5a}$$

$$\neg \mathit{HoldsAt}(f, t) \ \leftarrow \ [\mathit{InitiallyN}(f) \ \wedge \ \neg \mathit{Declipped}(0, f, t)] \tag{EC6a}$$

Note that fluents which are *InitiallyP* or *InitiallyN* are initially in the frame, whereas those which are neither *InitiallyP* nor *InitiallyN* are effectively initially “released”. Hence axiom (EC8a) (see Section 3.1) implies that all fluents are initially in the frame, and may or may not be appropriate for a given domain.

3.8 The Event Calculus, Continuous Change and Mathematical Modelling

The techniques using the *Trajectory* and *AntiTrajectory* predicates discussed in Section 3.5 are sufficient for modelling domains with very simple forms of continuous change, in particular where an explicit function of time is known for a particular parameter after a particular fluent has been initiated or terminated. However, this method is in general insuffcient for integrating standard mathematical modelling techniques with the Event Calculus, for several reasons. First, the majority of mathematical models are expressed as sets of differential equations, and these cannot in general be solved so as to produce explicit functions of time for each parameter involved. Second, there might only be incomplete knowledge, expressed perhaps using inequalities, about the mathematical relationship between various parameters and/or their derivatives. Third, the circumstances under which various mathematical relationships hold between parameters might not be (easily) expressible in terms of a single fluent. Fourth, trajectories and antitrajectories do not provide mechanisms for describing continuous change in time intervals before any relevant initiating and/or terminating actions have occurred.

A more general approach is to include domain independent axioms which explicitly utilise the mathematical definitions of continuity and differentiability of real-valued functions of time. Under this approach, which is partly inspired by Sandewall's work [Sand 89a, Sand 89b] and described in more detail in [MiSh 96], continuity of real-valued parameters is regarded as a default analogous to default persistence of fluents, so that discontinuities only arise in particular parameters when specific actions occur. For this section, we will assume that time is represented either as the real numbers or as the non-negative real numbers. We will assume that some or all terms of sort $\mathcal{P}$ (introduced in Section 3.5) represent real-valued functions of time, and accordingly introduce two new function symbols $Value : \mathcal{P} \times \mathcal{T} \mapsto \Re$ and $\delta : \mathcal{P} \mapsto \mathcal{P}$. The term $Value(P,T)$ represents the numerical value of parameter P at time T, and the axiomatisation below ensures that the term $Value(\delta(P),T)$ represents the numerical value at time T of its first derivative (at all time-points where this exists).

To integrate the standard mathematical concepts of continuity and differentiability into the Event Calculus, we need to express them in terms of *Value* and δ. It is also convenient to introduce the predicates $LeftContinuous \subseteq \mathcal{P} \times \mathcal{T}$ and $RightLimit \subseteq \mathcal{P} \times \mathcal{T}$ to capture the corresponding (standard) mathematical concepts[6]:

$$Continuous(p,t) \equiv \forall r \exists t_1 \forall t_2 [[|t - t_2| < t_1 \wedge 0 < r] \rightarrow |Value(p,t) - Value(p,t_2)| < r] \tag{EC20j}$$

$$\begin{aligned} &Differentiable(p,t) \equiv \\ &\quad \forall r \exists t_1 \forall t_2 [[0 < |t - t_2| < t_1 \wedge 0 < r] \rightarrow \\ &\qquad |(\tfrac{Value(p,t) - Value(p,t_2)}{t - t_2}) - Value(\delta(p),t)| < r] \end{aligned} \tag{EC21j}$$

$$\begin{aligned} &LeftContinuous(p,t) \equiv \\ &\quad \forall r \exists t_1 \forall t_2 [[t_2 < t \wedge (t - t_2) < t_1 \wedge 0 < r] \rightarrow \\ &\qquad |Value(p,t) - Value(p,t_2)| < r] \end{aligned} \tag{EC22j}$$

$$\begin{aligned} &RightLimit(p,t,r) \equiv \\ &\quad \forall r_1 \exists t_1 \forall t_2 [[t < t_2 \wedge (t_2 - t) < t_1 \wedge 0 < r_1] \\ &\qquad \rightarrow |Value(p,t_2) - r| < r_1] \end{aligned} \tag{EC23j}$$

To respect the convention that actions take effect immediately *after* they occur, it is necessary to axiomatise the mathematical constraint that, at every time-point (including those at which actions occur), the function associated with each parameter is left-hand continuous:

$$LeftContinuous(p,t) \tag{EC24j}$$

To describe instantaneous changes in the values of parameters at times when actions occur, and discontinuities in their corresponding functions of time, the predicates $BreaksTo \subseteq \mathcal{A} \times \mathcal{P} \times \mathcal{T} \times \Re$ and $Breaks \subseteq \mathcal{A} \times \mathcal{P} \times \mathcal{T}$ are introduced. Both are minimised (by circumscribing them in parallel). $BreaksTo(A,P,T,R)$ should be read as 'at time T, an occurrence of action A will cause parameter P to instantaneously take on value R'. More precisely, Axiom (EC27j) below states that if A does indeed occur at time

[6] A function is *left-continuous* if discontinuities occur only between successive intervals where the first is closed on the right and the second is open on the left. For example the function $f(t) = 0$ for all $t \leq 1$, $f(t) = 2$ otherwise, is left-continuous at all time-points, whereas the function $f'(t) = 0$ for all $t < 1$, $f'(t) = 2$ otherwise, is not. The *right-limit* of a function at a particular point is the limit value as the point is approached from the right. So, for example, the right-limit of both f and f' at 1 is 2.

T, then R is the value of the right-hand limit of P at T. $Breaks(A, P, T)$ can be read as 'at time T, action A potentially causes a discontinuity in parameter P'. The following domain-independent axioms make direct use of *BreaksTo* and *Breaks*. Axioms (EC25j) and (EC26j) can be likened to 'frame axioms' for parameters. Axiom (EC28j) states the relationship between *BreaksTo* and *Breaks*, and Axiom (EC29j) states that if an action potentially causes a discontinuity in a given parameter, it also potentially causes discontinuities in its higher derivatives.

$$\neg[Happens(a,t) \wedge Breaks(a,p,t)] \rightarrow Continuous(p,t) \tag{EC25j}$$

$$\neg[Happens(a,t) \wedge Breaks(a,\delta(p),t)] \rightarrow Differentiable(p,t) \tag{EC26j}$$

$$[BreaksTo(a,p,t,r) \wedge Happens(a,t)] \rightarrow RightLimit(p,t,r) \tag{EC27j}$$

$$BreaksTo(a,p,t,r) \rightarrow Breaks(a,p,t) \tag{EC28j}$$

$$Breaks(a,p,t) \rightarrow Breaks(a,\delta(p),t) \tag{EC29j}$$

To make useful derivations using this axiomatisation, for any given time point T it is useful to be able to refer to the next point after T at which an action occurs, if there is such a point. Axioms (EC30j), (EC31j) and (EC32j) state that if any action occurs at any time point after T, then the term $Next(T)$ refers to the least such time point. (Such points are somewhat analogous to the "least natural time points" discussed in [Reit 96].)

$$t < Next(t) \tag{EC30j}$$

$$[t < t_1 \ \wedge \ t_1 < Next(t)] \ \rightarrow \ \neg Happens(a,t_1) \tag{EC31j}$$

$$[Happens(a_1,t_1) \ \wedge \ t < t_1] \ \rightarrow \ \exists a.Happens(a, Next(t)) \tag{EC32j}$$

The above axiomatisation leaves us free to include (unsolved) sets of simultaneous differential equations in domain descriptions. As a simple illustration, suppose we wish to represent that the rate of change of the level of liquid in a tank is negatively proportional to the flow through a valve in its bottom, and that when the valve is open the flow is in turn proportional to the level (i.e. pressure). We need a single fluent *ValveOpen*, two parameters *Level* and *Flow*, and actions *OpenValve* and *CloseValve*. As well as *Happens*, *Initiates* and *Terminates* facts such as

$$Initiates(OpenValve,\ ValveOpen, t) \tag{V1}$$

$$Terminates(CloseValve,\ ValveOpen, t) \tag{V2}$$

we can represent information about the instantaneous effects of actions on parameters using *Breaks*,

$$Breaks(OpenValve, Flow, t) \tag{V3}$$

$$Breaks(OpenValve, \delta(Level), t) \tag{V4}$$

and include mathematical constraints (differential equations) which hold in different circumstances, e.g.

$$Value(\delta(Level), t) = -Value(Flow, t) \tag{V5}$$

$$HoldsAt(ValveOpen, t) \rightarrow \exists r[Value(Flow, t) = r.Value(Level, t)] \tag{V6}$$

In this case the full theory will include the circumscription $CIRC[(\text{EC28j}) \wedge (\text{EC29j}) \wedge (\text{V3}) \wedge (\text{V4})\ ;\ Breaks, BreaksTo]$. The Event Calculus now allows us to infer new boundary conditions for sets of differential equations which become applicable when actions such as *OpenValve* and *CloseValve* occur. A variation of this example is discussed in more detail in [MiSh 96].

The above axiomatisation lays a foundation for integrating the Event Calculus with representational and computational techniques from the field of Qualitative Reasoning [CrEt 92] [Kuip 94]. An Event Calculus based axiomatisation of some of the basic concepts in [Kuip 94] is given in [MiSh 96].

4 Other Formulations and Applications of the Event Calculus

The primary intention of this article has been to describe alternative axiomatisations of the Event Calculus in classical logic, using circumscription as a mechanism for default reasoning. However, in this section we briefly catalogue a selection of work in developing or applying a wider range of formulations of the Event Calculus.

The Event Calculus was originally formulated as a **logic program** in [KoSe 86], and many alternative and extended logic program formulations have subsequently been proposed, including [DeMi 92], [DeVa 96], [KaMi 97], [KaMi 99], [Kowa 92], [SaKo 95], [Shan 90], [VaDe 94a], [VaDe 94b], [VaDe 95] and [VaDe 96].

The Event Calculus has also been formulated in **modal logic** in [CeCh 95], [CeCh 96], [CeFr 97a], [CeFr 97b], [CeFr 98] and [ChMo 94], as an **action description language** in [KaMi 97] and [KaMi 98], and in an **argumentation framework** in [KaMi 99].

The Event Calculus has been applied to **planning** using **abduction** in [Eshg 88], [Chle 96], [JuFi 96], [Shan 97b], [Shan 98] and [Jung 98] and in particular to **cognitive robotics** in [Shan 96a], [Shan 96b], [Shan 98] and [Shan 99b]. Abduction in the context of the Event Calculus is also discussed in [DeMi 92], [VaDe 94a] and [VaDe 94b]. Other applications of the Event Calculus include **database updates** in [Kowa 92] and [VaDe 94b], **accident report processing** in [LeQu 98], and **legal reasoning** in [Kowa 95].

Various formulations of Event Calculus have been extended to deal with **continuous change** and **mathematical modelling** in [MiSh 96], [Shan 90] and [VaDe 94a], to deal with **ramifications** in [DeTh 98], [KaMi 98] and [Shan 99a], to deal with representing **agent beliefs** in [LeQu 98], and to deal with **programming constructs** and **compound events** in [Davi 96] and [Shan 99c].

Finally, much work has been done on both formal and informal comparisons between various formulations of the **Situation Calculus** (see e.g. [McHa 69], [Bake 91], [Reit 91]) and the Event Calculus. This work includes [KaMi 97], [KoSa 94], [KoSa 97], [Mill 95], [PiRe 93], [Prov 95], [VaDe 95] and [VaDe 96].

References

[Bake 91] A. Baker, *Nonmonotonic Reasoning in the Framework of the Situation Calculus*, Artificial Intelligence, Vol 49(5-23), 1991.

[CeCh 95] I. Cervesato, L. Chittaro and A. Montanari, *A Modal Calculus of Partially Ordered Events in a Logic Programming Framework*, in Proceedings ICLP'95, MIT Press, pages 299-313, 1995.

[CeCh 96] I. Cervesato, L. Chittaro and A. Montanari, *A General Modal Framework for the Event Calculus and its Skeptical and Creduluos Variants*, in in W. Wahlster, editor, Proceedings of the Twelth European Conference on Artificial Intelligence (ECAI'96), pp. 33-37, John Wiley and Sons, 1996.

[CeFr 97a] I. Cervesato, M. Franceschet and A. Montanari, *A Heirarchy of Modal Event Calculi: Expressiveness and Complexity*, in H.Barringer *et al*, Proceedings of the 2nd International Conference on Temporal Logic (ICTL'97, pp. 1-17, Kluwer Applied Logic Series, 1997.

[CeFr 97b] I. Cervesato, M. Franceschet and A. Montanari, *Modal Event Calculi with Preconditions*, in R. Morris and L. Khatib, Proceedings of the Fourth International Workshop on Temporal Reasoning (TIME'97), pp. 38-45, IEEE Computer Society Press, 1997.

[CeFr 98] I. Cervesato, M. Franceschet and A. Montanari, *The Complexity of Model Checking in Modal Event Calculi with Quantifiers*, Journal of Electronic Transactions on Artificial Intelligence, Linköping University Electronic Press, `http://www.ida.liu.se/ext/etai/`, 1998.

[ChMo 94] L. Chittaro, A. Montanari and A. Provetti, *Skeptical and Credulous Event Calculi for Supporting Modal Queries*, in A. Cohn, Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI'94), pp. 361-365, John Wiley and Sons, 1994.

[Chle 96] N. Chleq, *Constrained Resolution and Abductive Temporal Reasoning*, Computational Intelligence, vol. 12, no. 3, pp. 383-406, 1996.

[CrEt 92] J. M. Crawford and D. W. Etherington, *Formalizing Reasoning about Change: A Qualitative Reasoning Approach*, Proceedings AAAI'92, pp. 577-583, 1992.

[Davi 96] J. Davila, *Reactive Pascal and the Event Calculus*, Proceedings FAPR'96 Workshop on Reasoning about Actions and Planning in Complex Environments, eds. U. Siegmund and M. Thielscher, vol. 11 of Techncial Report AIDA, 1996.

[DeMi 92] M. Denecker, L. Missiaen and M. Bruynooghe, *Temporal Reasoning with Abductive Event Calculus*, in Proceedings ECAI 92, Vienna, 1992.

[DeVa 96] M. Denecker, K. Van Belleghem, G. Duchatelet, F. Piessens and D. De Schreye *A Realistic Experiment in Knowledge Representation in Open Event Calculus : Protocol Specification*, in Proceedings of the Joint International Conference and Symposium on Logic Programming, 1996.

[DeTh 98] M. Denecker, D. Theseider Dupré, and K. Van Belleghem, *An Inductive Definition Approach to Ramifications*, in Electronic Transactions on Artificial Intelligence, vol 2, 1998.

[Dohe 94] P. Doherty, *Reasoning about Action and Change Using Occlusion*, Proceedings ECAI'94, pp. 401-405, 1994.

[Eshg 88] K. Eshghi, *Abductive Planning with Event Calculus*, Proceedings of the 5th International Conference and Symposium on Logic Programming, ed.s Robert Kowalski and Kenneth Bowen, MIT Press, pp. 562-579, 1988.

[JuFi 96] C. G. Jung, K. Fischer and A. Burt, *Multi-Agent Planning Using an Abductive Event Calculus*, DFKI Report RR-96-04 (1996), DFKI, Germany, 1996.

[Jung 98] C. G. Jung, *Situated Abstraction Planning by Abductive Temporal Reasoning*, Proceedings ECAI'98, pp. 383-387, 1998.

[KaMi 97] A. Kakas and R. Miller, *A Simple Declarative Language for Describing Narratives with Actions*, JLP 31(1–3) (Special Issue on Reasoning about Action and Change) 157–200, 1997.

[KaMi 98] A. Kakas and R. Miller, *Reasoning about Actions, Narratives and Ramifications*, Journal of Electronic Transactions on Artificial Intelligence 1(4), Linköping University Electronic Press, `http://www.ida.liu.se/ext/etai/`, 1998.

[KaMi 99] A. Kakas, R. Miller and F. Toni, *An Argumentation Framework for Reasoning about Actions and Change*, Proceedings of LPNMR'99, 1999.

[KaLi 95] G. N. Kartha and V. Lifschitz, *A Simple Formalization of Actions Using Circumscription*, Proceedings IJCAI'95, pp. 1970-1975, 1995.

[Kowa 92] R. A. Kowalski, *Database Updates in the Event Calculus*, Journal of Logic Programming, vol. 12, pp. 121-146, 1992.

[Kowa 95] R. A. Kowalski, *Legislation as Logic Programs*, Informatics and the Foundations of Legal Reasoning, Kluwer Academic Publishers, ed.s Z. Bankowski et al., pp. 325-356, 1995.

[KoSa 94] R. A. Kowalski and F. Sadri, *The Situation Calculus and Event Calculus Compared*, in Proceedings of the International Logic Programming Symposium (ILPS'94), 1994.

[KoSa 97] R. A. Kowalski and F. Sadri, *Reconciling the Event Calculus with the Situation Calculus*, Journal of Logic Programming, Special Issue on Reasoning about Action and Change, vol. 31, pp. 39-58, 1997.

[KoSe 86] R. A. Kowalski and M. J. Sergot, *A Logic-Based Calculus of Events*, New Generation Computing, vol. 4, pp. 67-95, 1986.

[Kuip 94] B. Kuipers, *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*, MIT Press, 1994.

[LeQu 98] F. Lévy and Joachim Quantz, *Representing Beliefs in a Situated Event Calculus*, Proceedings ECAI'98, pp. 547-551, 1998.

[Lifs 94] V. Lifschitz, *Circumscription*, in The Handbook of Logic in Artificial Intelligence and Logic Programming, Volume 3: Nonmonotonic Reasoning and Uncertain Reasoning, ed. D. M. Gabbay, C .J. Hogger and J. A. Robinson, Oxford University Press, pp. 297-352, 1994.

[McCa 80] J. McCarthy, *Circumscription - A Form of Non-Monotonic Reasoning*, Artificial Intelligence, vol. 13, pp. 27-39, 1980.

[McHa 69] J. McCarthy and P. J. Hayes, *Some Philosophical Problems from the Standpoint of Artificial Intelligence*, in Machine Intelligence 4, ed. D. Michie and B. Meltzer, Edinburgh University Press, pp. 463-502, 1969.

[Mill 95] R. Miller, *Situation Calculus Specifications for Event Calculus Logic Programs*, in Proceedings of the Third International Conference on Logic Programming and Non-monotonic Reasoning, Lexington, KY, USA, Springer Verlag, 1995.

[MiSh 96] R. S. Miller and M. P. Shanahan, *Reasoning about Discontinuities in the Event Calculus*, Proceedings 1996 Knowledge Representation Conference (KR'96), pp. 63-74, 1996.

[PiRe 93] J. Pinto and R. Reiter, *Temporal Reasoning in Logic Programming: A Case for the Situation Calculus*, Proceedings ICLP 93, page 203, 1993.

[Prov 95] A. Provetti, *Hypothetical Reasoning about Actions: From Situation Calculus to Event Calculus*, Computational Intelligence, volume 12, number 2, 1995.

[Reit 91] R. Reiter, *The Frame Problem in the Situation Calculus: A Simple Solution (Sometimes) and a Completeness Result for Goal Regression*, in Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy, ed. V. Lifschitz, Academic Press, pp. 359-380, 1991.

[Reit 96] R. Reiter, *Natural actions, concurrency and continuous time in the situation calculus*, in Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR'96), Cambridge, Massachusetts, U.S.A, November 5-8, 1996.

[SaKo 95] F. Sadri and R. Kowalski, *Variants of the Event Calculus*, Proceedings of the International Conference on Logic Programming, Kanagawa, Japan, Stirling L. (Ed), The MIT Press, pp. 67-81, 1995.

[Sand 89a] E. Sandewall, *Combining Logic and Differential Equations for Describing Real World Systems*, Proceedings KR'89, Morgan Kaufman, 1989.

[Sand 89b] E. Sandewall, *Filter Preferential Entailment for the Logic of Action in Almost Continuous Worlds*, Proceedings IJCAI'89, pages 894-899, 1989.

[Sand 94] E. Sandewall, *The Representation of Knowledge about Dynamical Systems, Volume 1*, Oxford University Press, 1994.

[Shan 90] M. P. Shanahan, *Representing Continuous Change in the Event Calculus*, Proceedings ECAI'90, pp. 598-603, 1990.

[Shan 95] M. P. Shanahan, *A Circumscriptive Calculus of Events*, Artificial Intelligence, vol 77 (1995), pages 249-284, 1995.

[Shan 96a] M. P. Shanahan, *Robotics and the Common Sense Informatic Situation*, Proceedings ECAI'96, pp. 684-688, 1996.

[Shan 96b] M. P. Shanahan, *Noise and the Common Sense Informatic Situation for a Mobile Robot*, Proceedings AAAI'96, pp. 1098-1103, 1996.

[Shan 97a] M. P. Shanahan, *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*, MIT Press, 1997.

[Shan 97b] M. P. Shanahan, *Event Calculus Planning Revisited*, Proceedings 4th European Conference on Planning (ECP'97), Springer Lecture Notes in Artificial Intelligence no. 1348, pp. 390-402,1997.

[Shan 97c] M. P. Shanahan, *Noise, Non-Determinism and Spatial Uncertainty*, Proceedings AAAI'97, pp. 153-158, 1997.

[Shan 98] M. P. Shanahan, *Reinventing Shakey*, Working Notes of the 1998 AAAI Fall Symposium on Cognitive Robotics, pp. 125-135, 1998.

[Shan 99a] M. P. Shanahan, *The Ramification Problem in the Event Calculus*, Proceedings IJCAI'99, 1999.

[Shan 99b] M. P. Shanahan, *A Logical Account of the Common Sense Informatic Situation for a Mobile Robot*, Electronic Transactions on Artificial Intelligence, 1999.

[Shan 99c] M. P. Shanahan, *The Event Calculus Explained*, in Artificial Intelligence Today, eds. M. J. Wooldridge and M. Veloso, Springer-Verlag Lecture Notes in Artificial Intelligence no. 1600, Springer-Verlag, pages 409-430, 1999.

[VaDe 94a] K. Van Belleghem, M. Denecker and D. De Schreye, *Representing Continuous Change in the Abductive Event Calculus*, in Proceedings 1994 International Conference on Logic Programming, ed. P. Van Hentenrijck, pages 225-240, 1994.

[VaDe 94b] K. Van Belleghem, M. Denecker and D. De Schreye, *The Abductive Event Calculus as a General Framework for Temporal Databases*, Proceedings of the International Conference on Temporal Logic, 1994.

[VaDe 95] K. Van Belleghem, M. Denecker and D. De Schreye, *Combining Situation Calculus and Event Calculus*, in Proceedings of the International Conference on Logic Programming, 1995.

[VaDe 96] K. Van Belleghem, M. Denecker and D. De Schreye, *On the Relation Between Situation Calculus and Event Calculus*, Journal of Logic Programming, 31(1–3) (Special Issue on Reasoning about Action and Change), 1996.