Chapter 1

Event calculus

Erik T. Mueller

Citation: Mueller, Erik T. (in press). Event calculus. In Frank van Hermelen, Vladimir Lifschitz, & Bruce Porter (Eds.), Handbook of Knowledge Representation. Amsterdam: Elsevier.

1.1 Introduction

The event calculus [45, 66, 74, 98, 100] is a formalism for reasoning about action and change. Like the situation calculus, the event calculus has actions, which are called events, and time-varying properties or fluents. In the situation calculus, performing an action in a situation gives rise to a successor situation. Situation calculus actions are hypothetical, and time is tree-like. In the event calculus, there is a single time line on which actual events occur.

A narrative is a possibly incomplete specification of a set of actual event occurrences [63, 98]. The event calculus is narrative-based, unlike the standard situation calculus in which an exact sequence of hypothetical actions is represented.

Like the situation calculus, the event calculus supports context-sensitive effects of events, indirect effects, action preconditions, and the commonsense law of inertia. Certain phenomena are addressed more naturally in the event calculus, including concurrent events, continuous time, continuous change, events with duration, nondeterministic effects, partially ordered events, and triggered events.

We use a simple example to illustrate what the event calculus does. Suppose we wish to reason about turning on and off a light. We start by representing general knowledge about the effects of events:

If a light’s switch is flipped up, then the light will be on.
If a light’s switch is flipped down, then the light will be off.

We then represent a specific scenario:
1.1 Predicate/Function Meaning

<table>
<thead>
<tr>
<th>Predicate/Function</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Holds(p)</td>
<td>p holds</td>
</tr>
<tr>
<td>Start(p, e)</td>
<td>e starts p</td>
</tr>
<tr>
<td>End(p, e)</td>
<td>e ends p</td>
</tr>
<tr>
<td>Initiates(e, f)</td>
<td>e initiates f</td>
</tr>
<tr>
<td>Terminates(e, f)</td>
<td>e terminates f</td>
</tr>
<tr>
<td>$e_1 &lt; e_2$</td>
<td>$e_1$ precedes $e_2$</td>
</tr>
<tr>
<td>Broken($e_1, f, e_2$)</td>
<td>f is broken between $e_1$ and $e_2$</td>
</tr>
<tr>
<td>Incompatible($f_1, f_2$)</td>
<td>$f_1$ and $f_2$ are incompatible</td>
</tr>
<tr>
<td>After(e, f)</td>
<td>time period after e in which f holds</td>
</tr>
<tr>
<td>Before(e, f)</td>
<td>time period before e in which f holds</td>
</tr>
</tbody>
</table>

Table 1.1: Original event calculus (OEC) predicates and functions ($e, e_1, e_2 = event occurrences, f, f_1, f_2 = fluents, p = time period$)

The light was off at time 0.
Then the light’s switch was flipped up at time 5.
Then the light’s switch was flipped down at time 8.

We use the event calculus to conclude the following:

At time 3, the light was off.
At time 7, the light was on.
At time 10, the light was off.

In this chapter, we discuss several versions of the event calculus, the use of circumscription in the event calculus, methods of knowledge representation using the event calculus, automated event calculus reasoning, and applications of the event calculus. We use languages of classical many-sorted predicate logic with equality.1

1.2 Versions of the Event Calculus

The event calculus has evolved considerably from its original version. In this section, we trace the development of the event calculus and present its important versions.

1.2.1 Original Event Calculus (OEC)

The original event calculus (OEC) was introduced in 1986 by Kowalski and Sergot [45]. OEC has sorts for event occurrences, fluents, and time periods. The predicates and functions of the original event calculus are shown in Table 1.1. The axioms of the original event calculus are as follows.

---

1We do not treat modal logic versions of the event calculus [9].
OEC1. $\text{Initiates}(e, f) \equiv \text{Holds}(\text{After}(e, f))$

OEC2. $\text{Terminates}(e, f) \equiv \text{Holds}(\text{Before}(e, f))$

OEC3. $\text{Start}(\text{After}(e, f), e)$

OEC4. $\text{End}(\text{Before}(e, f), e)$

OEC5. $\text{After}(e_1, f) = \text{Before}(e_2, f) \supset \text{Start}(\text{Before}(e_2, f), e_1)$

OEC6. $\text{After}(e_1, f) = \text{Before}(e_2, f) \supset \text{End}(\text{After}(e_1, f), e_2)$

OEC7. $\text{Holds}(\text{After}(e_1, f)) \land \text{Holds}(\text{Before}(e_2, f)) \land e_1 < e_2 \land \neg \text{Broken}(e_1, f, e_2) \supset \text{After}(e_1, f) = \text{Before}(e_2, f)$

OEC8. $\text{Broken}(e_1, f, e_2) \equiv \exists e, f_1 ((\text{Holds}(\text{After}(e, f_1)) \lor \text{Holds}(\text{Before}(e, f_1))) \land \text{Incompatible}(f, f_1) \land e_1 < e < e_2)$

Let OEC be the conjunction of OEC1 through OEC8.

**Example 1.** Consider the example of turning on and off a light. We have an event occurrence $E_1$, which precedes event occurrence $E_2$:

$$E_1 < E_2 \quad (1.1)$$

$E_1$ turns on the light and $E_2$ turns it off:

$$\text{Initiates}(e, f) \equiv (e = E_1 \land f = \text{On}) \lor (e = E_2 \land f = \text{Off}) \quad (1.2)$$

$$\text{Terminates}(e, f) \equiv (e = E_1 \land f = \text{Off}) \lor (e = E_2 \land f = \text{On}) \quad (1.3)$$

The light cannot be both on and off:

$$\text{Incompatible}(f_1, f_2) \equiv (f_1 = \text{On} \land f_2 = \text{Off}) \lor (f_1 = \text{Off} \land f_2 = \text{On}) \quad (1.4)$$

We also assume the following:

$$E_1 \neq E_2 \quad (1.5)$$

$$\text{On} \neq \text{Off} \quad (1.6)$$

$$\neg (e < e) \quad (1.7)$$

We can then prove that the time period after $E_1$ in which the light is on equals the time period before $E_2$ in which the light is on. Let $\Sigma$ be the conjunction of (1.1) through (1.7).

**Proposition 1.** $\Sigma \land \text{OEC} \models \text{After}(E_1, \text{On}) = \text{Before}(E_2, \text{On})$

---

*Kowalski and Sergot [45] use implication ($\supset$) in OEC1 and OEC2. Sadri [87, p. 134] points out that bi-implication ($\equiv$) was intended by Kowalski and Sergot but not used in order to prevent looping when running the axioms in Prolog.*
Proof. From (1.2), (1.3), OEC1, and OEC2, we have
\[ \text{Holds}(\text{After}(e,f)) \equiv (e = E_1 \land f = \text{On}) \lor (e = E_2 \land f = \text{Off}) \]  
(1.8)
\[ \text{Holds}(\text{Before}(e,f)) \equiv (e = E_1 \land f = \text{Off}) \lor (e = E_2 \land f = \text{On}) \]  
(1.9)
From (1.8), (1.9), (1.4), (1.6), (1.7), and OEC8, we get \( \neg \text{Broken}(E_1, \text{On}, E_2) \). From this, \( \text{Holds}(\text{Before}(E_1, \text{On})) \equiv (e = E_1 \land f = \text{Off}) \lor (e = E_2 \land f = \text{On}) \) (which follows from (1.9)), (1.1), and OEC7, we have \( \text{After}(E_1, \text{On}) = \text{Before}(E_2, \text{On}) \).

We can also prove that \( E_1 \) starts the time period before \( E_2 \) in which \( \text{On} \) holds and that \( E_2 \) ends the time period after \( E_1 \) in which \( \text{On} \) holds.

**Proposition 2.** \( \Sigma \land \text{OEC} \models \text{Start}(\text{Before}(E_2, \text{On}), E_1) \land \text{End}(\text{After}(E_1, \text{On}), E_2) \)

Proof. This follows from Proposition 1, OEC5, and OEC6.

Pinto and Reiter [82] argue that the \text{Holds} predicate of the original event calculus is problematic, because it represents that “time periods hold.” They point out some undesirable consequences of axioms OEC3 and OEC4. In our light example, \( \text{Start}(\text{After}(E_1, \text{Off}), E_1) \) follows from OEC3. But what is the time period \( \text{After}(E_1, \text{Off}) \)? Whatever it is, it does not hold. From (1.5), (1.6), and (1.8), we have \( \neg \text{Holds}(\text{After}(E_1, \text{Off})) \). Similarly, from OEC3, we get \( \text{Start}(\text{After}(E_2, \text{On}), E_2) \) and from OEC4, we get \( \text{End}(\text{Before}(E_2, \text{Off}), E_2) \). Sadri and Kowalski [88] suggest modifying OEC3 to \( \text{Holds}(\text{After}(e,f)) \supset \text{Start}(\text{After}(e,f), e) \) and OEC4 to \( \text{Holds}(\text{Before}(e,f), e) \supset \text{End}(\text{Before}(e,f), e) \).

### 1.2.2 Simplified Event Calculus (SEC)

The simplified event calculus (SEC) was proposed in 1986 by Kowalski [40, p. 25] (see also [41]) and developed by Sadri [87, pp. 137–139], Eshghi [17], and Shanahan [93, 94, 98]. It differs from the original event calculus in the following ways:

- It replaces time periods with timepoints, which are either nonnegative integers or nonnegative real numbers.
- It replaces event occurrences or tokens with event types. The predicate \( \text{Happens}(e, t) \) represents that event (type) \( e \) occurs at timepoint \( t \).
- It eliminates the notion of incompatible fluents.
- It adds a predicate \( \text{Initially}(f) \), which represents that fluent \( f \) is initially true [95, p. 254] (see also [12] and [98, p. 253]).

The predicates of the simplified event calculus are shown in Table 1.2. The axioms of the simplified event calculus are as follows.

\[
\text{SEC1.} \quad ((\text{Initially}(f) \land \neg \text{StoppedIn}(0, f, t)) \lor \\
\exists e, t_1 (\text{Happens}(e, t_1) \land \text{Initiates}(e, f, t_1) \land t_1 < t \land \neg \text{StoppedIn}(t_1, f, t))) \equiv
\]
<table>
<thead>
<tr>
<th>Predicate</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Initially(f)$</td>
<td>$f$ is true at timepoint 0</td>
</tr>
<tr>
<td>$HoldsAt(f, t)$</td>
<td>$f$ is true at $t$</td>
</tr>
<tr>
<td>$Happens(e, t)$</td>
<td>$e$ occurs at $t$</td>
</tr>
<tr>
<td>$Initiates(e, f, t)$</td>
<td>if $e$ occurs at $t$, then $f$ is true after $t$</td>
</tr>
<tr>
<td>$Terminates(e, f, t)$</td>
<td>if $e$ occurs at $t$, then $f$ is false after $t$</td>
</tr>
<tr>
<td>$StoppedIn(t_1, f, t_2)$</td>
<td>$f$ is stopped between $t_1$ and $t_2$</td>
</tr>
</tbody>
</table>

Table 1.2: Simplified event calculus (SEC) predicates ($e$ = event, $f$ = fluent, $t, t_1, t_2$ = timepoints)

$HoldsAt(f, t)^3$

SEC2. $StoppedIn(t_1, f, t_2) \equiv \exists e, t (Happens(e, t) \land t_1 < t < t_2 \land Terminates(e, f, t))$

Let SEC be the conjunction of SEC1 and SEC2.

SEC1 represents that (1) a fluent that is initially true remains true until it is terminated, and (2) a fluent that is initiated remains true until it is terminated. Thus fluents are subject to the commonsense law of inertia [48, 49, 98], which states that a fluent’s truth value persists unless the fluent is affected by an event.

**Example 2.** Consider again the example of turning on and off a light. If a light is turned on, it will be on, and if a light is turned off, it will no longer be on:

\[
Initiates(e, f, t) \equiv (e = \text{TurnOn} \land f = \text{On})
\]  \hspace{1cm} (1.10)

\[
Terminates(e, f, t) \equiv (e = \text{TurnOff} \land f = \text{On})
\]  \hspace{1cm} (1.11)

Initially, the light is off:

\[\neg Initially(\text{On})\]  \hspace{1cm} (1.12)

The light is turned on at timepoint 2 and turned off at timepoint 4:

\[
Happens(e, t) \equiv (e = \text{TurnOn} \land t = 2) \lor (e = \text{TurnOff} \land t = 4)
\]  \hspace{1cm} (1.13)

We further assume the following:

\[\text{TurnOn} \neq \text{TurnOff}\]  \hspace{1cm} (1.14)

We can then show that the light will be off at timepoint 1, on at timepoint 3, and off again at timepoint 5. Let $\Sigma$ be the conjunction of (1.10) through (1.14).

**Proposition 3.** $\Sigma \land \text{SEC} \models \neg HoldsAt(\text{On}, 1)$

**Proof.** From (1.13), we have $\neg \exists e, t_1 (Happens(e, t_1) \land \text{Initiates}(e, \text{On}, t_1) \land t_1 < 1 \land \neg\text{StoppedIn}(t_1, \text{On}, 1))$. From this, (1.12), and SEC1, we have $\neg HoldsAt(\text{On}, 1)$.

---

$^3$Kowalski [40] uses implication, whereas Sadri [87] uses bi-implication.
\textbf{Proposition 4.} \(\Sigma \land \text{SEC} \models \text{HoldsAt}(\text{On}, 3)\)

\textit{Proof.} From (1.13) and SEC2, we have \(\neg \text{StoppedIn}(2, \text{On}, 3)\). From this, \(\text{Happens}(\text{TurnOn}, 2)\) (which follows from (1.13)), \(\text{Initiates}(\text{TurnOn}, \text{On}, 2)\) (which follows from (1.10)), \(2 < 3\), and SEC1, we have \(\text{HoldsAt}(\text{On}, 3)\). \hfill \Box

\textbf{Proposition 5.} \(\Sigma \land \text{SEC} \models \neg \text{HoldsAt}(\text{On}, 5)\)

\textit{Proof.} From \(\text{Happens}(\text{TurnOff}, 4)\) (which follows from (1.13)), \(2 < 4 < 5\), \(\text{Terminates}(\text{TurnOff}, \text{On}, 4)\) (which follows from (1.11)), and SEC2, we have \(\text{StoppedIn}(2, \text{On}, 5)\). From this, (1.13), and (1.10), we have \(\neg \exists e, t_1 (\text{Happens}(e, t_1) \land \text{Initiates}(e, \text{On}, t_1) \land t_1 < 5 \land \neg \text{StoppedIn}(t_1, \text{On}, 5))\). From this, (1.12), and SEC1, we have \(\neg \text{HoldsAt}(\text{On}, 5)\). \hfill \Box

\subsection{1.2.3 Basic Event Calculus (BEC)}

Shanahan [94, 95, 96, 98] extended the simplified event calculus by allowing fluents to be released from the commonsense law of inertia via the \textit{Releases} predicate, and adding the ability to represent continuous change via the \textit{Trajectory} predicate. The \textit{Initially} predicate is broken into two predicates \textit{InitiallyP} and \textit{InitiallyN}. We call this version of the event calculus the basic event calculus (BEC).

\textit{Releases}(e, f, t) represents that, if event \(e\) occurs at timepoint \(t\), then fluent \(f\) will be released from the commonsense law of inertia after \(t\). In SEC, a fluent that is initiated remains true until it is terminated, and a fluent that is terminated remains false until it is initiated. In BEC, a fluent that is initiated remains true until it is terminated or released, and a fluent that is terminated remains false until it is initiated or released. After a fluent is released, its truth value is not determined by BEC and is permitted to vary. Thus there are models in which the fluent is true, and models in which the fluent is false.

This opens up several possibilities. First, releasing a fluent frees it up so that other axioms in the domain description can be used to determine its truth value. This allows us to represent continuous change using \textit{Trajectory}, as discussed in Section 1.5.7, and indirect effects, as discussed in Section 1.5.9. Second, released fluents can be used to represent nondeterministic effects of events, as discussed in Section 1.5.8.

\textit{Trajectory}(f_1, t_1, f_2, t_2) represents that, if fluent \(f_1\) is initiated by an event that occurs at timepoint \(t_1\), then fluent \(f_2\) will be true at timepoint \(t_1 + t_2\). This can be used to represent fluents that change as a function of time. The domain description is usually written so that the fluent \(f_2\) is released by the events that initiate \(f_1\).

The predicates of the basic event calculus are shown in Table 1.3. The axioms of the basic event calculus are as follows.

\begin{enumerate}
  \item \textbf{BEC1.} \textit{StoppedIn}(t_1, f, t_2) \equiv \\
  \exists e, t (\text{Happens}(e, t) \land t_1 < t < t_2 \land (\text{Terminates}(e, f, t) \lor \text{Releases}(e, f, t)))
  \\
  \item \textbf{BEC2.} \textit{StartedIn}(t_1, f, t_2) \equiv \\
  \exists e, t (\text{Happens}(e, t) \land t_1 < t < t_2 \land (\text{Initiates}(e, f, t) \lor \text{Releases}(e, f, t)))
\end{enumerate}
<table>
<thead>
<tr>
<th>Predicate</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>InitiallyN(f)</td>
<td>f is false at timepoint 0</td>
</tr>
<tr>
<td>InitiallyP(f)</td>
<td>f is true at timepoint 0</td>
</tr>
<tr>
<td>HoldsAt(f, t)</td>
<td>f is true at t</td>
</tr>
<tr>
<td>Happens(e, t)</td>
<td>e occurs at t</td>
</tr>
<tr>
<td>Initiates(e, f, t)</td>
<td>if e occurs at t, then f is true and not released from the commonsense law of inertia after t</td>
</tr>
<tr>
<td>Terminates(e, f, t)</td>
<td>if e occurs at t, then f is false and not released from the commonsense law of inertia after t</td>
</tr>
<tr>
<td>Releases(e, f, t)</td>
<td>if e occurs at t, then f is released from the commonsense law of inertia after t</td>
</tr>
<tr>
<td>StoppedIn(t₁, f₁, t₂)</td>
<td>f is stopped between t₁ and t₂</td>
</tr>
<tr>
<td>StartedIn(t₁, f₁, t₂)</td>
<td>f is started between t₁ and t₂</td>
</tr>
<tr>
<td>Trajectory(f₁, t₁, f₂, t₂)</td>
<td>if f₁ is initiated by an event that occurs at t₁, then f₂ is true at t₁ + t₂</td>
</tr>
</tbody>
</table>

Table 1.3: Basic event calculus (BEC) predicates (e = event, f, f₁, f₂ = fluents, t, t₁, t₂ = timepoints)

BEC3. Happens(e, t₁) ∧ Initiates(e, f₁, t₁) ∧ 0 < t₂ ∧ Trajectory(f₁, t₁, f₂, t₂) ∧ ¬StoppedIn(t₁, f₁, t₁ + t₂) ⊃ HoldsAt(f₂, t₁ + t₂)

BEC4. InitiallyP(f) ∧ ¬StoppedIn(0, f, t) ⊃ HoldsAt(f, t)

BEC5. InitiallyN(f) ∧ ¬StartedIn(0, f, t) ⊃ ¬HoldsAt(f, t)

BEC6. Happens(e, t₁) ∧ Initiates(e, f, t₁) ∧ t₁ < t₂ ∧ ¬StoppedIn(t₁, f, t₂) ⊃ HoldsAt(f, t₂)

BEC7. Happens(e, t₁) ∧ Terminates(e, f, t₁) ∧ t₁ < t₂ ∧ ¬StartedIn(t₁, f, t₂) ⊃ ¬HoldsAt(f, t₂)

Let BEC be the conjunction of BEC1 through BEC7.

**Example 3.** Consider once again the example of turning on and off a light. We replace (1.12) with the following:

InitiallyN(On)                        \hspace{1cm} (1.15)

We add the following:

¬Releases(e, f, t)                     \hspace{1cm} (1.16)

Let Σ be the conjunction of (1.10), (1.11), (1.13), (1.14), (1.15), and (1.16). We then have the same results as for SEC.
Proposition 6. $\Sigma \land \text{BEC} \models \neg\text{HoldsAt}(On, 1)$

Proof. From (1.13) and BEC2, we have $\neg\text{StartedIn}(0, On, 1)$. From this, (1.15), and BEC5, we have $\neg\text{HoldsAt}(On, 1)$. $\square$

Proposition 7. $\Sigma \land \text{BEC} \models \text{HoldsAt}(On, 3)$

Proof. From (1.13) and BEC1, we have $\neg\text{StoppedIn}(2, On, 3)$. From this, \text{Happens}(\text{TurnOn}, 2) \text{ (which follows from (1.13))}, \text{Initiates}(\text{TurnOn}, On, 2) \text{ (which follows from (1.10))}, 2 < 3$, and BEC6, we have $\text{HoldsAt}(On, 3)$. $\square$

Proposition 8. $\Sigma \land \text{BEC} \models \neg\text{HoldsAt}(On, 5)$

Proof. From (1.13) and BEC2, we have $\neg\text{StartedIn}(4, On, 5)$. From this, \text{Happens}(\text{TurnOff}, 4) \text{ (which follows from (1.13))}, \text{Terminates}(\text{TurnOff}, On, 4) \text{ (which follows from (1.11))}, 4 < 5$, and BEC7, we have $\neg\text{HoldsAt}(On, 5)$. $\square$

Example 4. We can use \text{Releases} and \text{Trajectory} to represent a light that alternately emits red and green when it is turned on. If a light is turned on, it will be on:

$$\text{Initiates}(e, f, t) \equiv (e = \text{TurnOn} \land f = \text{On}) \quad (1.17)$$

If a light is turned on, whether it is red or green will be released from the commonsense law of inertia:

$$\text{Releases}(e, f, t) \equiv (e = \text{TurnOn} \land (f = \text{Red} \lor f = \text{Green})) \quad (1.18)$$

If a light is turned off, it will not be on, red, or green:

$$\text{Terminates}(e, f, t) \equiv (e = \text{TurnOff} \land (f = \text{On} \lor f = \text{Red} \lor f = \text{Green})) \quad (1.19)$$

After a light is turned on, it will alternately emit red for two seconds and green for two seconds:

$$\begin{align*}
(t_2 \mod 4) < 2 & \supset \text{Trajectory}(\text{On}, t_1, \text{Red}, t_2) \\
(t_2 \mod 4) \geq 2 & \supset \text{Trajectory}(\text{On}, t_1, \text{Green}, t_2)
\end{align*} \quad (1.20)$$

The light is not simultaneously red and green:

$$\neg\text{HoldsAt}(\text{Red}, t) \lor \neg\text{HoldsAt}(\text{Green}, t) \quad (1.22)$$

The light is turned on at timepoint 2:

$$\text{Happens}(e, t) \equiv (e = \text{TurnOn} \land t = 2) \quad (1.23)$$

We also assume

$$\text{TurnOn} \neq \text{TurnOff} \quad (1.24)$$

We can then show that the light will be red at timepoint 3, green at timepoint 5, red at timepoint 7, and so on. Let $\Sigma$ be the conjunction of (1.17) through (1.24).
Proposition 9. \( \Sigma \land \text{BEC} \vdash \text{HoldsAt}(\text{Red}, 3) \)

Proof. From (1.20) by universal instantiation, we have

\[
\text{Trajectory}(\text{On}, 2, \text{Red}, 1)
\]  
(1.25)

From (1.23) and BEC1, we have \( \neg \text{StoppedIn}(2, \text{On}, 3) \). From this, \( \text{Happens}(\text{TurnOn}, 2) \) (which follows from (1.23)), \( \text{Initiates}(\text{TurnOn}, \text{On}, 2) \) (which follows from (1.17)), \( 0 < 1 \), (1.25), and BEC3, we have \( \text{HoldsAt}(\text{Red}, 3) \). \( \square \)

Proposition 10. \( \Sigma \land \text{BEC} \vdash \text{HoldsAt}(\text{Green}, 5) \)

Proof. From (1.21) by universal instantiation, we have

\[
\text{Trajectory}(\text{On}, 2, \text{Green}, 3)
\]  
(1.26)

From (1.23) and BEC1, we have \( \neg \text{StoppedIn}(2, \text{On}, 5) \). From this, \( \text{Happens}(\text{TurnOn}, 2) \) (which follows from (1.23)), \( \text{Initiates}(\text{TurnOn}, \text{On}, 2) \) (which follows from (1.17)), \( 0 < 3 \), (1.26), and BEC3, we have \( \text{HoldsAt}(\text{Green}, 5) \). \( \square \)

Proposition 11. \( \Sigma \land \text{BEC} \vdash \text{HoldsAt}(\text{Red}, 7) \)

Proof. From (1.20) by universal instantiation, we have

\[
\text{Trajectory}(\text{On}, 2, \text{Red}, 5)
\]  
(1.27)

From (1.23) and BEC1, we have \( \neg \text{StoppedIn}(2, \text{On}, 7) \). From this, \( \text{Happens}(\text{TurnOn}, 2) \) (which follows from (1.23)), \( \text{Initiates}(\text{TurnOn}, \text{On}, 2) \) (which follows from (1.17)), \( 0 < 5 \), (1.27), and BEC3, we have \( \text{HoldsAt}(\text{Red}, 7) \). \( \square \)

1.2.4 Event Calculus (EC)

Miller and Shanahan [65, 66] introduced several alternative formulations of the basic event calculus. A number of their axioms can be combined [70] to produce what we call EC, which differs from the basic event calculus in the following ways:

- It allows negative time. Timepoints are either integers or real numbers.
- It eliminates the \( \text{InitiallyN} \) and \( \text{InitiallyP} \) predicates.
- It explicitly represents that a fluent is released from the commonsense law of inertia using the \( \text{ReleasedAt} \) predicate.
- It adds \( \text{AntiTrajectory} \).
- It treats \( \text{StoppedIn} \) and \( \text{StartedIn} \) as abbreviations rather than predicates, and introduces other abbreviations.
ReleasedAt(e, t) represents that fluent f is released from the commonsense law of inertia at timepoint t. AntiTrajectory(f₁, t₁, f₂, t₂) represents that, if fluent f₁ is terminated by an event that occurs at timepoint t₁, then fluent f₂ will be true at timepoint t₁ + t₂.

The predicates of EC are shown in Table 1.4. The axioms and definitions of EC are as follows.

EC1. \( Clipped(t₁, f, t₂) \equiv \exists e, t (Happens(e, t) \land t₁ \leq t < t₂ \land Terminates(e, f, t)) \)

EC2. \( Declipped(t₁, f, t₂) \equiv \exists e, t (Happens(e, t) \land t₁ \leq t < t₂ \land Initiates(e, f, t)) \)

EC3. \( StartedIn(t₁, f, t₂) \equiv \exists e, t (Happens(e, t) \land t₁ < t < t₂ \land Terminates(e, f, t)) \)

EC4. \( StartedIn(t₁, f, t₂) \equiv \exists e, t (Happens(e, t) \land t₁ < t < t₂ \land Initiates(e, f, t)) \)

EC5. \( Happens(e, t₁) \land Initiates(e, f₁, t₁) \land 0 < t₂ \land Trajectory(f₁, t₁, f₂, t₂) \land \neg StoppedIn(t₁, f₁, t₁ + t₂) \supset HoldsAt(f₂, t₁ + t₂) \)

EC6. \( Happens(e, t₁) \land Terminates(e, f₁, t₁) \land 0 < t₂ \land AntiTrajectory(f₁, t₁, f₂, t₂) \land \neg StartedIn(t₁, f₁, t₁ + t₂) \supset HoldsAt(f₂, t₁ + t₂) \)

EC7. \( PersistsBetween(t₁, f, t₂) \equiv \neg \exists t (ReleasedAt(f, t) \land t₁ < t \leq t₂) \)

EC8. \( ReleasedBetween(t₁, f, t₂) \equiv \exists e, t (Happens(e, t) \land t₁ \leq t < t₂ \land Releases(e, f, t)) \)

### Table 1.4: EC and DEC predicates (e = event, f, f₁, f₂ = fluents, t, t₁, t₂ = timepoints)

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>HoldsAt(e, t)</td>
<td>e is true at t</td>
</tr>
<tr>
<td>Happens(e, t)</td>
<td>e occurs at t</td>
</tr>
<tr>
<td>ReleasedAt(e, t)</td>
<td>e is released from the commonsense law of inertia at t</td>
</tr>
<tr>
<td>Initiates(e, f, t)</td>
<td>if e occurs at t, then f is true and not released from the commonsense law of inertia after t</td>
</tr>
<tr>
<td>Terminates(e, f, t)</td>
<td>if e occurs at t, then f is false and not released from the commonsense law of inertia after t</td>
</tr>
<tr>
<td>Releases(e, f, t)</td>
<td>if e occurs at t, then f is released from the commonsense law of inertia after t</td>
</tr>
<tr>
<td>Trajectory(f₁, t₁, f₂, t₂)</td>
<td>if f₁ is initiated by an event that occurs at t₁, then f₂ is true at t₁ + t₂</td>
</tr>
<tr>
<td>AntiTrajectory(f₁, t₁, f₂, t₂)</td>
<td>if f₁ is terminated by an event that occurs at t₁, then f₂ is true at t₁ + t₂</td>
</tr>
</tbody>
</table>
EC9. \( \text{HoldsAt}(f, t_1) \land t_1 < t_2 \land \text{PersistsBetween}(t_1, f, t_2) \land \neg\text{Clipped}(t_1, f, t_2) \supset \text{HoldsAt}(f, t_2) \)

EC10. \( \neg\text{HoldsAt}(f, t_1) \land t_1 < t_2 \land \text{PersistsBetween}(t_1, f, t_2) \land \neg\text{Declipped}(t_1, f, t_2) \supset \neg\text{HoldsAt}(f, t_2) \)

EC11. \( \text{ReleasedAt}(f, t_1) \land t_1 < t_2 \land \neg\text{Clipped}(t_1, f, t_2) \land \neg\text{Declipped}(t_1, f, t_2) \supset \neg\text{ReleasedAt}(f, t_2) \)

EC12. \( \neg\text{ReleasedAt}(f, t_1) \land t_1 < t_2 \land \neg\text{Clipped}(t_1, f, t_2) \land \neg\text{Declipped}(t_1, f, t_2) \supset \neg\text{ReleasedAt}(f, t_2) \)

EC13. \( \text{ReleasedIn}(t_1, f, t_2) \overset{\text{def}}{=} \exists e, t (\text{Happens}(e, t) \land t_1 < t < t_2 \land \text{Releases}(e, f, t)) \)

EC14. \( \text{Happens}(e, t_1) \land \text{Initiates}(e, f, t_1) \land t_1 < t_2 \land \neg\text{StoppedIn}(t_1, f, t_2) \land \neg\text{ReleasedIn}(t_1, f, t_2) \supset \text{HoldsAt}(f, t_2) \)

EC15. \( \text{Happens}(e, t_1) \land \text{Terminates}(e, f, t_1) \land t_1 < t_2 \land \neg\text{StartedIn}(t_1, f, t_2) \land \neg\text{ReleasedIn}(t_1, f, t_2) \supset \neg\text{HoldsAt}(f, t_2) \)

EC16. \( \text{Happens}(e, t_1) \land \text{Releases}(e, f, t_1) \land t_1 < t_2 \land \neg\text{StoppedIn}(t_1, f, t_2) \land \neg\text{StartedIn}(t_1, f, t_2) \supset \text{ReleasedAt}(f, t_2) \)

EC17. \( \text{Happens}(e, t_1) \land (\text{Initiates}(e, f, t_1) \lor \text{Terminates}(e, f, t_1)) \land t_1 < t_2 \land \neg\text{ReleasedIn}(t_1, f, t_2) \supset \neg\text{ReleasedAt}(f, t_2) \)

Let \( \text{EC} \) be the formula generated by conjoining axioms EC5, EC6, EC9, EC10, EC11, EC12, EC14, EC15, EC16, and EC17 and then expanding the predicates \text{Clipped}, \text{Declipped}, \text{StoppedIn}, \text{StartedIn}, \text{PersistsBetween}, \text{ReleasedBetween}, \) and \text{ReleasedIn} using definitions EC1, EC2, EC3, EC4, EC7, EC8, and EC13.

**Example 5.** Consider again the light example. We replace (1.15) with the following:

\( \neg\text{HoldsAt}(\text{On}, 0) \) \hspace{1cm} (1.28)

We add the following:

\( \neg\text{ReleasedAt}(f, t) \) \hspace{1cm} (1.29)

Let \( \Sigma \) be the conjunction of (1.10), (1.11), (1.13), (1.14), (1.16), (1.28), and (1.29). Again, we get the same results.

**Proposition 12.** \( \Sigma \land \text{EC} \models \neg\text{HoldsAt}(\text{On}, 1) \)

*Proof.* From (1.13) and EC2, we have \( \neg\text{Declipped}(0, \text{On}, 1) \). From this, (1.28), \( 0 < 1 \), \text{PersistsBetween}(0, \text{On}, 1) (which follows from (1.29) and EC7), and EC10, we have \( \neg\text{HoldsAt}(\text{On}, 1) \).

\( \blacksquare \)
Proposition 13. $\Sigma \land \text{EC} \models \text{HoldsAt}(\text{On}, 3)$

Proof. From (1.13) and EC3, we have $\neg \text{StoppedIn}(2, \text{On}, 3)$. From this, $\text{Happens}(\text{TurnOn}, 2)$ (which follows from (1.13)), $\text{Initiates}(\text{TurnOn}, \text{On}, 2)$ (which follows from (1.10)), $2 < 3$, $\neg \text{ReleasedIn}(2, \text{On}, 3)$ (which follows from (1.13) and EC13), and EC14, we have $\text{HoldsAt}(\text{On}, 3)$.

Proposition 14. $\Sigma \land \text{EC} \models \neg \text{HoldsAt}(\text{On}, 5)$

Proof. From (1.13) and EC4, we have $\neg \text{StartedIn}(4, \text{On}, 5)$. From this, $\text{Happens}(\text{TurnOff}, 4)$ (which follows from (1.13)), $\text{Terminates}(\text{TurnOff}, \text{On}, 4)$ (which follows from (1.11)), $4 < 5$, $\neg \text{ReleasedIn}(4, \text{On}, 5)$ (which follows from (1.13) and EC13), and EC15, we have $\neg \text{HoldsAt}(\text{On}, 5)$.

1.2.5 Discrete Event Calculus (DEC)

Mueller [70, 74] developed the discrete event calculus (DEC) to improve the efficiency of automated reasoning in the event calculus. DEC improves efficiency by limiting time to the integers, and eliminating triply quantified time from many of the axioms.

The predicates of DEC are the same as those of EC, as shown in Table 1.4. The axioms and definitions of DEC are as follows.

DEC1. $\text{StoppedIn}(t_1, f, t_2) \overset{\text{def}}{=} \exists e, t \ (\text{Happens}(e, t) \land t_1 < t < t_2 \land \text{Terminates}(e, f, t))$

DEC2. $\text{StartedIn}(t_1, f, t_2) \overset{\text{def}}{=} \exists e, t \ (\text{Happens}(e, t) \land t_1 < t < t_2 \land \text{Initiates}(e, f, t))$

DEC3. $\text{Happens}(e, t_1) \land \text{Initiates}(e, f_1, t_1) \land 0 < t_2 \land \text{Trajectory}(f_1, t_1, f_2, t_2) \land \neg \text{StoppedIn}(t_1, f_1, t_1 + t_2) \supset \text{HoldsAt}(f_2, t_1 + t_2)$

DEC4. $\text{Happens}(e, t_1) \land \text{Terminates}(e, f_1, t_1) \land 0 < t_2 \land \text{AntiTrajectory}(f_1, t_1, f_2, t_2) \land \neg \text{StartedIn}(t_1, f_1, t_1 + t_2) \supset \text{HoldsAt}(f_2, t_1 + t_2)$

DEC5. $\text{HoldsAt}(f, t) \land \neg \text{ReleasedAt}(f, t + 1) \land \neg \exists e \ (\text{Happens}(e, t) \land \text{Terminates}(e, f, t)) \supset \text{HoldsAt}(f, t + 1)$

DEC6. $\neg \text{HoldsAt}(f, t) \land \neg \text{ReleasedAt}(f, t + 1) \land \neg \exists e \ (\text{Happens}(e, t) \land \text{Initiates}(e, f, t)) \supset \neg \text{HoldsAt}(f, t + 1)$

DEC7. $\text{ReleasedAt}(f, t) \land \neg \exists e \ (\text{Happens}(e, t) \land (\text{Initiates}(e, f, t) \lor \text{Terminates}(e, f, t))) \supset \text{ReleasedAt}(f, t + 1)$

DEC8. $\neg \text{ReleasedAt}(f, t) \land \neg \exists e \ (\text{Happens}(e, t) \land \text{Releases}(e, f, t)) \supset \neg \text{ReleasedAt}(f, t + 1)$

DEC9. $\text{Happens}(e, t) \land \text{Initiates}(e, f, t) \supset \text{HoldsAt}(f, t + 1)$
DEC10. \[ \text{Happens}(e, t) \land \text{Terminates}(e, f, t) \supset \neg \text{HoldsAt}(f, t + 1) \]

DEC11. \[ \text{Happens}(e, t) \land \text{Releases}(e, f, t) \supset \text{ReleasedAt}(f, t + 1) \]

DEC12. \[ \text{Happens}(e, t) \land (\text{Initiates}(e, f, t) \lor \text{Terminates}(e, f, t)) \supset \neg \text{ReleasedAt}(f, t + 1) \]

Let DEC be the formula generated by conjoining axioms DEC3 through DEC12 and then expanding the predicates \text{StoppedIn} and \text{StartedIn} using definitions DEC1 and DEC2.

The difference between EC and DEC is that EC operates over spans of timepoints, whereas DEC operates timepoint by timepoint. For example, EC14 states that a fluent that is initiated remains true until it is terminated or released. This corresponds to several DEC axioms. DEC9 states that a fluent that is initiated is true at the next timepoint. DEC5 states that a fluent that is true, not released from the commonsense law of inertia, and not terminated, is true at the next timepoint. The axioms dealing with \text{Trajectory} and \text{AntiTrajectory}, DEC3 and DEC4, are the same as EC5 and EC6. The definitions of \text{StoppedIn} and \text{StartedIn}, DEC1 and DEC2, are the same as EC3 and EC4.

Example 6. Consider again the light example. Let \( \Sigma \) be as for EC.

**Proposition 15.** \( \Sigma \land \text{DEC} \models \neg \text{HoldsAt}(\text{On}, 1) \)

\[ \text{Proof.} \text{ From (1.13), we have } \neg \exists e \left( \text{Happens}(e, 0) \land \text{Initiates}(e, \text{On}, 0) \right). \text{ From this, (1.28), } \neg \text{ReleasedAt}(\text{On}, 1) \text{ (which follows from (1.29)), and DEC6, we have } \neg \text{HoldsAt}(\text{On}, 1). \]

**Proposition 16.** \( \Sigma \land \text{DEC} \models \text{HoldsAt}(\text{On}, 3) \)

\[ \text{Proof.} \text{ From } \text{Happens}(\text{TurnOn}, 2) \text{ (which follows from (1.13)), } \text{Initiates}(\text{TurnOn}, \text{On}, 2) \text{ (which follows from (1.10)), and DEC9, we have } \text{HoldsAt}(\text{On}, 3). \]

**Proposition 17.** \( \Sigma \land \text{DEC} \models \neg \text{HoldsAt}(\text{On}, 5) \)

\[ \text{Proof.} \text{ From } \text{Happens}(\text{TurnOff}, 4) \text{ (which follows from (1.13)), } \text{Terminates}(\text{TurnOff}, \text{On}, 4) \text{ (which follows from (1.11)), and DEC10, we have } \neg \text{HoldsAt}(\text{On}, 5). \]

1.2.6 Equivalence of DEC and EC

We have the following equivalence between DEC and EC.

**Proposition 18.** If the domain of the timepoint sort is restricted to the integers, then DEC is logically equivalent to EC.
Proof. \((EC \models \text{DEC})\) By universal instantiation, substituting \(t_1 + 1\) for \(t_2\). For example, \(\text{DEC}9\) is obtained from \(\text{EC}14\) via the following chain of equivalences:

\[
\text{Happens}(e, t_1) \land \text{Initiates}(e, f, t_1) \land t_1 < t_1 + 1 \land \neg \text{StoppedIn}(t_1, f, t_1 + 1) \land \\
\neg \text{ReleasedIn}(t_1, f, t_1 + 1) \supset \text{HoldsAt}(f, t_1 + 1) \\
\equiv \\
\text{Happens}(e, t_1) \land \text{Initiates}(e, f, t_1) \\
\neg \exists e, t (\text{Happens}(e, t) \land t_1 < t < t_1 + 1 \land \text{Terminates}(e, f, t)) \land \\
\neg \exists e, t (\text{Happens}(e, t) \land t_1 < t < t_1 + 1 \land \text{Releases}(e, f, t)) \supset \\
\text{HoldsAt}(f, t_1 + 1) \\
\equiv \text{ (for integer time)} \\
\text{Happens}(e, t_1) \land \text{Initiates}(e, f, t_1) \supset \text{HoldsAt}(f, t_1 + 1)
\]

\((\text{DEC} \models \text{EC})\) By a series of lemmas showing that each EC axiom follows from DEC. See [70] or [74].

1.2.7 Other Versions

Other versions of the event calculus have been developed to support

- causal constraints for instantaneously interacting indirect effects [101]
- continuously changing parameters using differential equations [64, 66]
- events with duration\(^4\) [66, 97, 100]
- hierarchical or compound events [97, 100]

1.3 Relationship to other Formalisms

The event calculus is closely related to the situation calculus (see Chapter ??) and temporal action logics (see Chapter ??). The relation between the event calculus and the situation calculus is treated by Kowalski and Sadri [43, 44] and Van Belleghem, Denecker, and De Schreye [112]. The relation between the event calculus and temporal action logics is treated by Mueller [75]. Bennett and Galton [4] define a versatile event logic (VEL) and use it to describe versions of the situation calculus and the event calculus. A problem for future research is the relation of the event calculus and nonmonotonic causal logic.

\(^4\)Events with duration may also be represented as fluents that are initiated and terminated by instantaneous events. For example, a moving event with duration can be represented using the axioms \(\text{Initiates} \left(\text{StartMoving}, \text{Moving}, f, t \right)\) and \(\text{Terminates} \left(\text{StopMoving}, \text{Moving}, f, t \right)\). See also the discussion of continuous change in Section 1.5.7.
1.4 Default Reasoning

An axiomatization is *elaboration tolerant* to the degree that it can be extended easily [61]. In the examples given so far, we have fully specified the effects of events and the event occurrences. That is, we have supplied bi-implications for the predicates *Initiates*, *Terminates*, *Releases*, and *Happens*. This is not very elaboration tolerant, because whenever we wish to add event effects and occurrences, we must modify these bi-implications.

Instead, we should be able to write individual axioms specifying what effects particular events have on particular fluents and what events occur. But then we have two problems:

1. how to derive what effects particular events do *not* have on particular fluents, or the *frame problem* [8, 62, 98, 105] (see also Section ??), and

2. how to derive what events do *not* occur.

These problems can be solved using any framework for default or nonmonotonic reasoning [5, 7] (see also Chapters ?? and ??). In this section, we discuss the use of circumscription [51, 56, 57, 59] (see Section ??) and negation as failure [11].

1.4.1 Circumscription

Consider the light example. Instead of writing the single axiom

\[ \text{Happens}(e, t) \equiv (e = \text{TurnOn} \land t = 2) \lor (e = \text{TurnOff} \land t = 4) \]  (1.30)

we write several axioms:

\[ \text{Happens}(\text{TurnOn}, 2) \]  (1.31)
\[ \text{Happens}(\text{TurnOff}, 4) \]  (1.32)

Then we circumscribe *Happens* in (1.31) \( \land \) (1.32), which yields (1.30).

Circumscription allows us to assume by default that the events known to occur are the only events that occur. That is, there are no extraneous events. If we allowed extraneous events, then we could no longer prove, say, that the light is off at timepoint 6, because we could no longer prove the absence of events turning on the light between timepoints 4 and 6. If we later add the axiom

\[ \text{Happens}(\text{TurnOn}, 5) \]

then we recompute the circumscription, which allows us to prove that in fact the light is on at timepoint 6.

Similarly, we write separate axioms for *Initiates*, *Terminates*, and *Releases*, and circumscribe these predicates, which allows us to assume by default that the known effects of events are the only effects of events. That is, there are no extraneous event effects. If we allowed extraneous event effects, then we could no longer prove that the light is off at timepoint 6 if some unrelated event occurred between timepoints 4 and 6, because we could no longer prove that the unrelated event does not turn on the light.
1.4.2 Computing Circumscription

It is difficult to compute circumscription in general [16]. The circumscription of a predicate in a formula, which is defined by a formula of second-order logic, does not always reduce to a formula of first-order logic [47]. In many cases, however, we can compute circumscription using the following two propositions. The first proposition asserts that certain circumscriptions reduce to predicate completion. (See Section ??.)

**Proposition 19.** Let \( \rho \) be an \( n \)-ary predicate symbol and \( \Delta(x_1, \ldots, x_n) \) be a formula whose only free variables are \( x_1, \ldots, x_n \). If \( \Delta(x_1, \ldots, x_n) \) does not contain \( \rho \), then the circumscription \( \text{CIRC}[\forall x_1, \ldots, x_n (\Delta(x_1, \ldots, x_n) \supset \rho(x_1, \ldots, x_n)); \rho] \) is equivalent to \( \forall x_1, \ldots, x_n (\Delta(x_1, \ldots, x_n) \equiv \rho(x_1, \ldots, x_n)) \).

**Proof.** See the proof of Proposition 2 of Lifschitz [51]. (See also [84].)

This gives us the following method for computing circumscription of \( \rho \) in a formula:

1. Rewrite the formula in the form \( \forall x_1, \ldots, x_n (\Delta(x_1, \ldots, x_n) \supset \rho(x_1, \ldots, x_n)) \), where \( \Delta(x_1, \ldots, x_n) \) does not contain \( \rho \).

2. Apply Proposition 19.

**Example 7.** Let \( \Sigma = \text{Initiates}(E_1, F_1, t) \land \text{Initiates}(E_2, F_2, t) \). We compute \( \text{CIRC}[\Sigma; \text{Initiates}] \) by rewriting \( \Sigma \) as

\[
(e = E_1 \land f = F_1) \lor (e = E_2 \land f = F_2) \supset \text{Initiates}(e, f, t)
\]

and then applying Proposition 19, which gives

\[
\text{Initiates}(e, f, t) \equiv (e = E_1 \land f = F_1) \lor (e = E_2 \land f = F_2)
\]

The second proposition allows us to compute the circumscription of several predicates, called parallel circumscription. We start with a definition.

**Definition 1.** A formula \( \Delta \) is positive relative to a predicate symbol \( \rho \) if and only if all occurrences of \( \rho \) in \( \Delta \) are in the range of an even number of negations in an equivalent formula obtained by eliminating \( \supset \) and \( \equiv \) from \( \Delta \).

**Proposition 20.** Let \( \rho_1, \ldots, \rho_n \) be predicate symbols and \( \Delta \) be a formula. If \( \Delta \) is positive relative to every \( \rho_i \), then the parallel circumscription \( \text{CIRC}[\Delta; \rho_1, \ldots, \rho_n] \) is equivalent to \( \bigwedge_{i=1}^n \text{CIRC}[\Delta; \rho_i] \).

**Proof.** See the proof of Proposition 14 of Lifschitz [51].

Further methods for computing circumscription are discussed in Section ??.

**Example 8.** Let \( \Sigma \) be the conjunction of the following axioms:

\[
\text{Initiates}(\text{TurnOn}, \text{On}, t)
\]
\[
\text{Terminates}(\text{TurnOff}, \text{On}, t)
\]
Let $\Delta$ be the conjunction of the following axioms:

\[
\text{Happens}(\text{TurnOn}, 2) \\
\text{Happens}(\text{TurnOff}, 4)
\]

Let $\Gamma$ be the conjunction of (1.14), (1.28), and (1.29). We can use circumscription to prove that the light is on at timepoint 3.

**Proposition 21.**

\[
\text{CIRC}\![\Sigma; \text{Initiates, Terminates, Releases}] \land \text{CIRC}\![\Delta; \text{Happens}] \land \Gamma \land \text{EC} \\
= \text{HoldsAt}(\text{On}, 3)
\]

**Proof.** From $\text{CIRC}\![\Sigma; \text{Initiates, Terminates, Releases}]$, Proposition 20, and Proposition 19, we have

\[
(\text{Initiates}(e, f, t) \equiv (e = \text{TurnOn} \land f = \text{On})) \land
(\text{Terminates}(e, f, t) \equiv (e = \text{TurnOff} \land f = \text{On})) \land
\neg \text{Releases}(e, f, t)
\]

From $\text{CIRC}\![\Delta; \text{Happens}]$ and Proposition 19, we have

\[
\text{Happens}(e, t) \equiv (e = \text{TurnOn} \land t = 2) \lor (e = \text{TurnOff} \land t = 4)
\]

From this and EC3, we have $\neg \text{StoppedIn}(2, \text{On}, 3)$. From this, $\text{Happens}(\text{TurnOn}, 2)$ (which follows from (1.34)), $\text{Initiates}(\text{TurnOn}, \text{On}, 2)$ (which follows from (1.33)), $2 < 3$, $\neg \text{ReleasedIn}(2, \text{On}, 3)$ (which follows from (1.34) and EC13), and EC14, we have $\text{HoldsAt}(\text{On}, 3)$. \hfill $\blacksquare$

### 1.4.3 Historical Note

Notice that we keep the event calculus axioms EC outside the scope of any circumscription. This technique, known as *filtering*, was introduced in the features and fluents framework [14, 15, 89, 90] (see also Chapter ??) and incorporated into the event calculus by Shanahan [96, 98]. The need for filtering became clear after Hanks and McDermott [32] introduced the Yale shooting scenario, which exposed problems with simply circumscribing the entire situation calculus axiomatization of the scenario. Shanahan [98] describes treatments of the Yale shooting scenario in the situation calculus and the event calculus; Shanahan [105] and Lifschitz [52] provide a modern perspective.

### 1.4.4 Negation As Failure

Instead of using circumscription for default reasoning in the event calculus, logic programming with the negation as failure operator not may be used [41, 45, 93, 94]. For example, we write rules such as the following:

\[
\text{clipped}(T1, F, T2) \leftarrow \text{happens}(E, T), T1 <= T, T < T2, \text{terminates}(E, F, T).
\]

\[
\text{holds_at}(F, T2) \leftarrow \text{holds_at}(F, T1), T1 < T2, \text{not clipped}(T1, F, T2).
\]
These rules are similar to axioms EC1 and EC9. Then we add rules such as the following to our domain description:

- \textit{initiates}(\textit{turn\_on}, \textit{on}, T).
- \textit{terminates}(\textit{turn\_off}, \textit{on}, T).
- \textit{happens}(\textit{turn\_on}, 2).
- \textit{happens}(\textit{turn\_off}, 4).


### 1.5 Event Calculus Knowledge Representation

This section describes methods for representing knowledge using the event calculus. These methods can be used with BEC, EC, and DEC. Those that do not involve trajectories or release from the commonsense law of inertia can also be used with SEC.

#### 1.5.1 Parameters

We represent events and fluents with parameters as functions that return event and fluent terms. For example, we may represent the event of person \(p\) turning on light \(l\) using a function \texttt{TurnOn}(\(p\), \(l\)), and the property that light \(l\) is turned on using a function \texttt{On}(\(l\)). We then require the following unique names axioms:

\[
\text{TurnOn}(p_1, l_1) = \text{TurnOn}(p_2, l_2) \supset p_1 = p_2 \land l_1 = l_2
\] (1.35)

\[
\text{On}(l_1) = \text{On}(l_2) \supset l_1 = l_2
\] (1.36)

If we have another event \texttt{TurnOff}(\(p\), \(l\)), then we also require the unique names axiom:

\[
\text{TurnOn}(p_1, l_1) \neq \text{TurnOff}(p_2, l_2)
\] (1.37)

The \textit{U} notation [48] is convenient for defining unique names axioms. If \(\phi_1, \ldots, \phi_k\) are function symbols, then \(\text{U}[\phi_1, \ldots, \phi_k]\) is an abbreviation for the conjunction of the formulas

\[
\phi_i(x_1, \ldots, x_m) \neq \phi_j(y_1, \ldots, y_n)
\]

where \(m\) is the arity of \(\phi_i\), \(n\) is the arity of \(\phi_j\), and \(x_1, \ldots, x_m\) and \(y_1, \ldots, y_n\) are distinct variables such that the sort of \(x_p\) is the sort of the \(p\)th argument position of \(\phi_i\) and the sort of \(y_p\) is the sort of the \(p\)th argument position of \(\phi_j\), for each \(1 \leq i < j \leq k\), and the conjunction of the formulas

\[
\phi_i(x_1, \ldots, x_m) = \phi_i(y_1, \ldots, y_m) \supset x_1 = y_1 \land \ldots \land x_m = y_m
\]

where \(m\) is the arity of \(\phi_i\) and \(x_1, \ldots, x_m\) and \(y_1, \ldots, y_m\) are distinct variables such that the sort of \(x_p\) and \(y_p\) is the sort of the \(p\)th argument position of \(\phi_i\), for each \(1 \leq i \leq k\).

We may then use this notation to replace (1.35), (1.36), and (1.37) with

\[
\text{U}[\text{TurnOn}, \text{TurnOff}] \land \text{U}[\text{On}]
\]

In the remainder of this section, we assume that appropriate unique names axioms are defined.
1.5.2 Event Effects

The effects of events are represented using effect axioms, which are of the form

\[ \gamma \supset \text{Initiates}(\alpha, \beta, \tau), \text{ or } \gamma \supset \text{Terminates}(\alpha, \beta, \tau) \]

where \( \gamma \) is a condition, \( \alpha \) is an event, \( \beta \) is a fluent, and \( \tau \) is a timepoint. A condition is a formula containing atoms of the form \( \text{HoldsAt}(\beta, \tau) \) and \( \neg \text{HoldsAt}(\beta, \tau) \), where \( \beta \) is a fluent and \( \tau \) is a timepoint.

**Example 9.** Consider a counter that can be incremented and reset. The fluent \( \text{Value}(c, v) \) represents that counter \( c \) has value \( v \). The event \( \text{Increment}(c) \) represents that counter \( c \) is incremented, and the event \( \text{Reset}(c) \) represents that the counter \( c \) is reset. We use two effect axioms to represent that, if the value of a counter is \( v \) and the counter is incremented, its value will be \( v + 1 \) and will no longer be \( v \):

\[
\begin{align*}
\text{HoldsAt}(\text{Value}(c, v), t) & \supset \text{Initiates}(\text{Increment}(c), \text{Value}(c, v + 1), t) \tag{1.38} \\
\text{HoldsAt}(\text{Value}(c, v), t) & \supset \text{Terminates}(\text{Increment}(c), \text{Value}(c, v), t) \tag{1.39}
\end{align*}
\]

We use two more effect axioms to represent that, if the value of a counter is \( v \) and the counter is reset, its value will be 0 and will no longer be \( v \):

\[
\begin{align*}
\text{Initiates}(\text{Reset}(c), \text{Value}(c, 0), t) \tag{1.40} \\
\text{HoldsAt}(\text{Value}(c, v), t) \land c \neq 0 & \supset \text{Terminates}(\text{Reset}(c), \text{Value}(c, v), t) \tag{1.41}
\end{align*}
\]

The effect of an event can depend on the context in which it occurs. The condition \( \gamma \) represents the context. In the example of the counter, the effect of incrementing the counter depends on its current value.

1.5.3 Preconditions

We might represent the effect of turning on a device as follows:

\[ \text{Initiates}(\text{TurnOn}(p, d), \text{On}(d), t) \]

But there are many things that could prevent a device from going on. It could be unplugged or broken, its on-off switch could be broken, and so on. A qualification is a condition that prevents an event from having its intended effects. The qualification problem is the problem of representing and reasoning about qualifications.

A partial solution to the qualification problem is to use preconditions. The condition \( \gamma \) of effect axioms can be used to represent preconditions.

**Example 10.** If a person turns on a device, then, provided the device is not broken, the device will be on:

\[
\neg \text{HoldsAt}(\text{Broken}(d), t) \supset \text{Initiates}(\text{TurnOn}(p, d), \text{On}(d), t) \tag{1.42}
\]

But this is not elaboration tolerant, because whenever we wish to add qualifications, we must modify (1.42). Instead we can use default reasoning.
Example 11. Instead of writing (1.42), we write

\[ \neg Ab_1(d, t) \supset Initiates(TurnOn(p, d), On(d), t) \]  

(1.43)

\( Ab_1(d, t) \) is an abnormality predicate [28, 58, 59, 60]. It represents that at timepoint \( t \), device \( d \) is abnormal in some way that prevents it from being turned on. In general, we use a distinct abnormality predicate for each type of abnormality. We then add qualifications by writing cancellation axioms [23, 51, 59]:

\[ HoldsAt(Broken(d), t) \supset Ab_1(d, t) \]  

(1.44)

\[ \neg HoldsAt(PluggedIn(d), t) \supset Ab_1(d, t) \]  

(1.45)

We then circumscribe the abnormality predicate \( Ab_1 \) in the conjunction of cancellation axioms (1.44) and (1.45), which yields

\[ Ab_1(d, t) \equiv HoldsAt(Broken(d), t) \lor \neg HoldsAt(PluggedIn(d), t) \]  

(1.46)

We then reason using (1.43) and (1.46). Whenever we wish to add additional qualifications, we add cancellation axioms and recompute the circumscription of the abnormality predicates in the cancellation axioms.

1.5.4 State Constraints

Law-like relationships among properties are represented using state constraints, which are formulas containing atoms of the form \( HoldsAt(\beta, \tau) \) and \( \neg HoldsAt(\beta, \tau) \), where \( \beta \) is a fluent and \( \tau \) is a timepoint.

For example, we may use two state constraints to represent that a counter has exactly one value at a time:

\[ \exists v \quad HoldsAt(Value(c, v), t) \]  

(1.47)

\[ HoldsAt(Value(c, v_1), t) \land HoldsAt(Value(c, v_2), t) \supset v_1 = v_2 \]  

(1.48)

1.5.5 Concurrent Events

In the event calculus, events may occur concurrently. That is, we may have \( Happens(e_1, t_1) \) and \( Happens(e_2, t_2) \) where \( e_1 \neq e_2 \) and \( t_1 = t_2 \). We represent the effects of concurrent events using effect axioms whose conditions contain atoms of the form \( Happens(\alpha, \tau) \) and \( \neg Happens(\alpha, \tau) \), where \( \alpha \) is an event and \( \tau \) is a timepoint [66].

Example 12. Consider again the example of the counter. Suppose that the value of a counter \( C \) is 5 at timepoint 0:

\[ HoldsAt(Value(C, 5), 0) \]  

(1.49)

Further suppose that the counter is simultaneously incremented and reset at timepoint 1:

\[ Happens(Increment(C), 1) \]  

(1.50)

\[ Happens(Reset(C), 1) \]  

(1.51)
(1.50) leads us to conclude

\[ \text{HoldsAt}(\text{Value}(C, 6), 2) \]

whereas (1.51) leads us to conclude

\[ \text{HoldsAt}(\text{Value}(C, 0), 2) \]

These formulas contradict the state constraint (1.48).

In order to deal with this, we may specify exactly what happens when a counter is simultaneously incremented and reset. There are a number of possibilities.

One possibility is that nothing happens. We replace the effect axioms (1.38), (1.39), (1.40), and (1.41) with the following effect axioms:

\[ \neg \text{Happens}(\text{Reset}(c), t) \land \text{HoldsAt}(\text{Value}(c, v), t) \supset \text{Initiates}(\text{Increment}(c), \text{Value}(c, v + 1), t) \]  
\[ (1.52) \]

\[ \neg \text{Happens}(\text{Reset}(c), t) \land \text{HoldsAt}(\text{Value}(c, v), t) \supset \text{Terminates}(\text{Increment}(c), \text{Value}(c, v), t) \]  
\[ (1.53) \]

\[ \neg \text{Happens}(\text{Increment}(c), t) \supset \text{Initiates}(\text{Reset}(c), \text{Value}(c, 0), t) \]  
\[ (1.54) \]

\[ \neg \text{Happens}(\text{Increment}(c), t) \land \text{HoldsAt}(\text{Value}(c, v), t) \land c \neq 0 \supset \text{Terminates}(\text{Reset}(c), \text{Value}(c, v), t) \]  
\[ (1.55) \]

Another possibility is that the counter is neither incremented nor reset, but that the counter enters an error state. We use the effect axioms (1.52), (1.53), (1.54), and (1.55), and a further effect axiom that represents that, if a counter is simultaneously reset and incremented, it will be in an error state:

\[ \text{Happens}(\text{Reset}(c), t) \supset \text{Initiates}(\text{Increment}(c), \text{Error}(c), t) \]

We could also have written this as

\[ \text{Happens}(\text{Increment}(c), t) \supset \text{Initiates}(\text{Reset}(c), \text{Error}(c), t) \]

Another possibility is that, if a counter is simultaneously reset and incremented, the
incrementing takes priority and the counter is incremented:

\[ \text{HoldsAt}(\text{Value}(c, v), t) \supset \text{Initiates}(\text{Increment}(c), \text{Value}(c, v+1), t) \]

\[ \text{HoldsAt}(\text{Value}(c, v), t) \supset \text{Terminates}(\text{Increment}(c), \text{Value}(c, v), t) \]

\[ \neg \text{Happens}(\text{Increment}(c), t) \supset \text{Initiates}(\text{Reset}(c), \text{Value}(c, 0), t) \]

\[ \neg \text{Happens}(\text{Increment}(c), t) \land \text{HoldsAt}(\text{Value}(c, v), t) \land c \neq 0 \supset \text{Terminates}(\text{Reset}(c), \text{Value}(c, v), t) \]

Similarly, we could represent that, if a counter is simultaneously reset and incremented, the resetting takes priority and the counter is reset.

### 1.5.6 Triggered Events

Events that are triggered under certain circumstances are represented using trigger axioms [94, 96, 98], which are of the form

\[ \gamma \supset \text{Happens}(\alpha, \tau) \]

where \( \gamma \) is a condition, \( \alpha \) is an event, and \( \tau \) is a timepoint.

**Example 13.** Consider a thermostat that turns on a heater when the temperature drops below \( A \), and turns off the heater when the temperature rises above \( B \). We represent this using two effect axioms and two trigger axioms:

\[ \text{Initiates}(\text{TurnOn}, \text{On}, t) \]
\[ \text{Terminates}(\text{TurnOff}, \text{On}, t) \]
\[ \text{HoldsAt}(\text{Temperature}(v), t) \land v < A \land \neg \text{HoldsAt}(\text{On}, t) \supset \text{Happens}(\text{TurnOn}, t) \]
\[ \text{HoldsAt}(\text{Temperature}(v), t) \land v > B \land \text{HoldsAt}(\text{On}, t) \supset \text{Happens}(\text{TurnOff}, t) \]

The conditions \( \neg \text{HoldsAt}(\text{On}, t) \) and \( \text{HoldsAt}(\text{On}, t) \) are required to prevent \( \text{TurnOn} \) and \( \text{TurnOff} \) from repeatedly triggering.

### 1.5.7 Continuous Change

Examples of continuous change include falling objects, expanding balloons, and containers being filled. Continuous change is represented using trajectory axioms [94, 95, 98], which are of the form

\[ \gamma \supset \text{Trajectory}(\beta_1, \tau_1, \beta_2, \tau_2), \text{or} \]
\[ \gamma \supset \text{AntiTrajectory}(\beta_1, \tau_1, \beta_2, \tau_2) \]
where $\gamma$ is a condition, $\beta_1$ and $\beta_2$ are fluents, and $\tau_1$ and $\tau_2$ are timepoints. Trajectory is used to determine the truth value of fluent $\beta_2$ after fluent $\beta_1$ is initiated, until $\beta_1$ is terminated. AntiTrajectory is used to determine the truth value of fluent $\beta_2$ after fluent $\beta_1$ is terminated, until $\beta_1$ is initiated.

Although DEC does not support continuous time, we may still use Trajectory and AntiTrajectory in DEC to represent gradual change. Gradual change is a discrete approximation to continuous change in which the value of a changing fluent is only represented for integer timepoints.

Example 14. Consider a falling object. We use effect axioms to represent that, if a person drops an object, then it will be falling, and if an object hits the ground, then it will no longer be falling:

$$\text{Initiates}(\text{Drop}(p,o), \text{Falling}(o), t)$$
$$\text{Terminates}(\text{HitGround}(o), \text{Falling}(o), t)$$

We represent that, if a person drops an object, then its height will be released from the commonsense law of inertia:

$$\text{Releases}(\text{Drop}(p,o), \text{Height}(o,h), t)$$

(For an object $o$, $\text{Height}(o,h)$ is released for all $h$.) We use a trajectory axiom to represent that the height of the object is given by an equation of free-fall motion, where $G$ is the acceleration due to gravity ($9.8 \text{ m/sec}^2$):

$$\text{HoldsAt}(\text{Height}(o,h), t_1) \supset \text{Trajectory}(\text{Falling}(o), t_1, \text{Height}(o, h - \frac{1}{2}Gt_1^2), t_2)$$

We use a trigger axiom to represent that, when an object is falling and its height is 0, it hits the ground:

$$\text{HoldsAt}(\text{Falling}(o), t) \land \text{HoldsAt}(\text{Height}(o, 0), t) \supset \text{Happens}(\text{HitGround}(o), t)$$

We specify that, if an object hits the ground and its height is $h$, then its height will be $h$ and its height will no longer be released from the commonsense law of inertia:

$$\text{HoldsAt}(\text{Height}(o,h), t) \supset \text{Initiates}(\text{HitGround}(o), \text{Height}(o,h), t)$$

We specify that an object has a unique height:

$$\text{HoldsAt}(\text{Height}(o,h_1), t) \land \text{HoldsAt}(\text{Height}(o,h_2), t) \supset h_1 = h_2$$

At timepoint 0, Nathan drops an apple whose height is $G/2$:

$$\neg \text{HoldsAt}(\text{Falling}(\text{Apple}), 0)$$
$$\text{HoldsAt}(\text{Height}(\text{Apple}, G/2), 0)$$
$$\text{Happens}(\text{Drop}(\text{Nathan, Apple}), 0)$$

We can then show that the apple will hit the ground at timepoint 1, and its height at timepoint 2 will be zero.
Proposition 22. Let $\Sigma = (1.56) \land (1.57) \land (1.58) \land (1.61), \Delta = (1.60) \land (1.65), \Omega = U[D\text{Drop}, HitGround] \land U[Falling, Height], \Gamma = (1.59) \land (1.62) \land (1.63) \land (1.64)$. Then we have

$$CIRC[\Sigma; \text{Initiates}, \text{Terminates}, \text{Releases}] \land CIRC[\Delta; \text{Happens}] \land \Omega \land \Gamma \land EC \models \text{HoldsAt}(\text{Height(Apple,0)},1) \land \text{Happens}(\text{HitGround(Apple)},1) \land \text{HoldsAt}(\text{Height(Apple,0)},2).$$

Proof. See the proofs of Propositions 7.2 and 7.3 of Mueller [74].

1.5.8 Nondeterministic Effects

Nondeterministic effects of events can be represented in the event calculus using determining fluents [98], or fluents released from the commonsense law of inertia that are used within the conditions of effect axioms.

Example 15. Consider the example of rolling a die with six sides. We define a determining fluent $\text{DieDF}(d, s)$ which represents that die $d$ will land on side $s$. This fluent is released from the commonsense law of inertia. In EC and DEC, we require the axiom

$$\text{ReleasedAt}(\text{DieDF}(d, s), t)$$

In BEC, a fluent that is never initiated or terminated and is neither InitiallyN nor InitiallyP is released from the commonsense law of inertia, so no further axioms are required to released $\text{DieDF}$ from the commonsense law of inertia.

We use state constraints to represent that, at any timepoint, $\text{DieDF}(d, s)$ assigns one of the sides $\{1, \ldots, 6\}$ to a die:

$$\exists s \text{HoldsAt}(\text{DieDF}(d, s), t)$$

$$\text{HoldsAt}(\text{DieDF}(d, s_1), t) \land \text{HoldsAt}(\text{DieDF}(d, s_2), t) \supset s_1 = s_2$$

$$\text{HoldsAt}(\text{DieDF}(d, s), t) \supset s = 1 \lor s = 2 \lor s = 3 \lor s = 4 \lor s = 5 \lor s = 6$$

We use effect axioms to represent that, if a die is rolled at a timepoint, it will land on the side assigned to the die by $\text{DieDF}$ at that timepoint:

$$\text{HoldsAt}(\text{DieDF}(d, s), t) \supset \text{Initiates}(\text{Roll}(d), \text{Side}(d, s), t)$$

$$\text{HoldsAt}(\text{Side}(d, s_1), t) \land \text{HoldsAt}(\text{DieDF}(d, s_2), t) \land s_1 \neq s_2 \supset \text{Terminates}(\text{Roll}(d), \text{Side}(d, s_1), t)$$

Suppose a die $D$ is rolled at timepoint 0:

$$\text{Happens}(\text{Roll}(D), 0)$$

What side of the die faces up at timepoint 1? Because $\text{DieDF}$ is free to take on any of six values at timepoint 0, we get six classes of models: one in which $\text{HoldsAt}(\text{DieDF}(D, 1), 0)$ and therefore $\text{HoldsAt}(\text{Side}(D, 1), 1)$, one in which $\text{HoldsAt}(\text{DieDF}(D, 2), 0)$ and therefore $\text{HoldsAt}(\text{Side}(D, 2), 1)$, and so on.
1.5.9 Indirect Effects

Suppose that a person and a book are in the living room of a house. When the person walks out of the living room, the book will normally remain in the living room. But if the person is holding the book and walks out of the living room, then the book will no longer be in the living room. That is, an indirect effect or ramification of the person walking out of the living room is that the book the person is holding changes location. The ramification problem \cite{19, 24, 101} is the problem of representing and reasoning about the indirect effects of events. Much research has been performed on the ramification problem \cite{2, 19, 24, 29, 30, 35, 48, 50, 53, 54, 55, 85, 91, 101, 111}. Several methods can be used for solving this problem in the event calculus.

Example 16 (State Constraints). Consider again the example of a light. We represent the direct effect of turning on a light using an effect axiom:

\begin{align*}
\text{Initiates}(\text{TurnOn}(l), \text{On}(l), t)
\end{align*}

We may use a state constraint to represent the indirect effect of turning on the light, namely that the light is not off:

\begin{align*}
\neg \text{HoldsAt}(\text{Off}(l), t) \equiv \text{HoldsAt}(\text{On}(l), t)
\end{align*}

The fluent \text{Off}(l) must be released from the commonsense law of inertia. In EC and DEC, we require the axiom

\begin{align*}
\text{ReleasedAt}(\text{Off}(l), t)
\end{align*}

This method of representing indirect effects works if it is possible to divide fluents into primitive and derived fluents \cite{39, 48, 50, 101}. Here \text{On} is primitive and \text{Off} is derived. The direct effects of events on primitive fluents are represented using effect axioms, whereas the indirect effects of events on derived fluents are represented using state constraints.

Example 17 (Release from Inertia and State Constraints). Suppose that we wish to represent the indirect effects of walking while holding an object, namely that the object moves along with the person holding it. We create a simple axiomatization of space. We start by representing the direct effects of walking. If a person walks from location \(l_1\) to location \(l_2\), then the person will be at \(l_2\) and will no longer be at \(l_1\):

\begin{align*}
\text{Initiates}(\text{Walk}(p, l_1, l_2), \text{At}(p, l_2), t) \\
\quad \quad \quad \quad l_1 \neq l_2 \quad \text{\text{Terminates}}(\text{Walk}(p, l_1, l_2), \text{At}(p, l_1), t)
\end{align*}

We also represent the direct effects of picking up and setting down an object. If a person and an object are at the same location and the person picks up the object, then the person will be holding the object:

\begin{align*}
\text{HoldsAt} (\text{At}(p, l), t) \land \text{HoldsAt}(\text{At}(o, l), t) &\land \text{Initiates}(\text{PickUp}(p, o), \text{Holding}(p, o), t)
\end{align*}

If a person sets down an object, then the person will no longer be holding it:

\begin{align*}
\text{Terminates}(\text{SetDown}(p, o), \text{Holding}(p, o), t)
\end{align*}
We then represent the indirect effects of walking with a *Releases* axiom, a state constraint, and an effect axiom. If a person and an object are at the same location and the person picks up the object, then the object’s location will be released from the commonsense law of inertia:

\[
\begin{align*}
\text{HoldsAt}(\text{At}(p, l), t) \land \text{HoldsAt}(\text{At}(o, l), t) & \supset \quad (1.66) \\
\text{Releases}(&\text{PickUp}(p, o), \text{At}(o, l'), t)
\end{align*}
\]

(For any given object \(o\), \(\text{At}(o, l')\) is released for all \(l'\).) If a person who is holding an object is located at \(l\), then the object is also located at \(l\):

\[
\begin{align*}
\text{HoldsAt}(\text{Holding}(p, o), t) \land \text{HoldsAt}(\text{At}(p, l), t) & \supset \quad \text{HoldsAt}(\text{At}(o, l), t) \quad (1.67)
\end{align*}
\]

If a person is holding an object, the person is located at \(l\), and the person sets down the object, then the object will be located at \(l\) and the object’s location will no longer be released from the commonsense law of inertia:

\[
\begin{align*}
\text{HoldsAt}(\text{Holding}(p, o), t) & \land \text{Initiates}(\text{SetDown}(p, o), \text{At}(o, l), t) \supset \\
\text{HoldsAt}(\text{At}(o, l), t) & \land l_1 \neq l_2 \supset \\
\text{Terminates}(\text{SetDown}(p, o), \text{At}(o, l), t)
\end{align*}
\]

**Example 18 (Effect Axioms).** Another way of representing indirect effects is simply to add more effect axioms. We replace (1.66), (1.67), and (1.68) with effect axioms that state that, if a person who is holding an object walks from location \(l_1\) to location \(l_2\), then the object will be at location \(l_2\) and will no longer be at \(l_1\):

\[
\begin{align*}
\text{HoldsAt}(\text{Holding}(p, o), t) & \supset \text{Initiates}(\text{Walk}(p, l_1, l_2), \text{At}(o, l_2), t) \\
\text{HoldsAt}(\text{Holding}(p, o), t) & \land l_1 \neq l_2 \supset \\
\text{Terminates}(\text{Walk}(p, l_1, l_2), \text{At}(o, l_1), t)
\end{align*}
\]

**Example 19 (Effect Constraints).** Another way of representing indirect effects is to use *effect constraints* [98, 101], which are of the form

\[
\gamma \land \pi_1(\alpha, \beta_1, \tau) \supset \pi_2(\alpha, \beta_2, \tau)
\]

where \(\gamma\) is a condition, \(\pi_1\) and \(\pi_2\) are *Initiates* or *Terminates*, \(\alpha\) is an event variable, \(\beta_1\) and \(\beta_2\) are fluents, and \(\tau\) is a timepoint. We use effect constraints to represent that an object moves along with the person holding it:

\[
\begin{align*}
\text{HoldsAt}(\text{Holding}(p, o), t) & \land \text{Initiates}(e, \text{At}(p, l), t) \supset \text{Initiates}(e, \text{At}(o, l), t) \\
\text{HoldsAt}(\text{Holding}(p, o), t) & \land \text{Terminates}(e, \text{At}(p, l), t) \supset \text{Terminates}(e, \text{At}(o, l), t)
\end{align*}
\]

The event calculus can also be extended to deal with instantaneously interacting indirect effects [101].

The aforementioned methods for dealing with ramifications have various advantages and disadvantages. The method of state constraints is simple, but it requires a clear separation of fluents into those directly affected by events (primitive fluents) and those indirectly affected by events (derived fluents).
The method of releasing a fluent from the commonsense law of inertia allows a fluent to be primitive at some timepoints and derived at other timepoints. But then more bookkeeping is required. We must release the fluent from the commonsense law of inertia, and later make the fluent again subject to this law.

The method of using effect axioms is also simple, but it is less elaboration tolerant. In our example, if we add another way for a person to change location, such as running, we must also add axioms for the indirect effects of running:

\[
\text{HoldsAt}(\text{Holding}(p, o), t) \supset \text{Initiates}(\text{Run}(p, l_1, l_2), \text{At}(o, l_2), t)
\]

\[
\text{HoldsAt}(\text{Holding}(p, o), t) \land l_1 \neq l_2 \supset \text{Terminates}(\text{Run}(p, l_1, l_2), \text{At}(o, l_1), t)
\]

The method of using effect constraints is the most elaboration tolerant. But we cannot apply Proposition 19 in order to compute the circumscription of \text{Initiates} and \text{Terminates} in effect constraints.

1.5.10 Partially Ordered Events

We may represent partially ordered events using inequalities involving timepoints. For example, we may represent that John picked up a pen and a pad in some unspecified order, and then walked from the office to the living room as follows:

\[
\text{Happens}(\text{PickUp}(\text{John}, \text{Pen}), T_1)
\]

\[
\text{Happens}(\text{PickUp}(\text{John}, \text{Pad}), T_2)
\]

\[
\text{Happens}(\text{Walk}(\text{John}, \text{Office}, \text{LivingRoom}), T_3)
\]

\[
T_1 < T_3
\]

\[
T_2 < T_3
\]

Using the simple axiomatization of space of Example 17 in Section 1.5.9, we can conclude that John was holding both the pen and the pad at \(T_3\), and that the pen and the pad are both in the living room after \(T_3\). But we cannot conclude that John was holding the pad when he picked up the pen, or that John was holding the pen when he picked up the pad. There are three classes of models:

1. those in which John picks up the pen and then the pad \((T_1 < T_2)\),
2. those in which John picks up the pad and then the pen \((T_2 < T_1)\), and
3. those in which John picks up the pen and pad simultaneously \((T_1 = T_2)\).

1.6 Action Language \(E\)

Instead of using classical logic for reasoning about action and change, specialized action languages [22, 25, 26, 81] can be used. The \(E\) action language introduced by Antonis C. Kakas and Rob Miller [35, 36] is closely related to the event calculus.

A language of \(E\) is specified by a set of fluents, a set of events, a set of timepoints, and a partial order on the set of timepoints. An \(E\) domain description consists of a set of statements, which are defined as follows.
Definition 2. If $\beta$ is a fluent, then $\beta$ and $\neg\beta$ are fluent literals.

Definition 3. If $\gamma$ is a fluent literal and $\tau$ is a timepoint, then

$$\gamma \text{ holds-at } \tau$$

is a statement.

Definition 4. If $\alpha$ is an event and $\tau$ is a timepoint, then

$$\alpha \text{ happens-at } \tau$$

is a statement.

Definition 5. If $\alpha$ is an event, $\beta$ is a fluent, and $\Gamma$ is a set of fluent literals, then

$$\alpha \text{ initiates } \beta \text{ when } \Gamma$$

and

$$\alpha \text{ terminates } \beta \text{ when } \Gamma$$

are statements.

The notation $\alpha \text{ initiates } \beta$ is an abbreviation for $\alpha \text{ initiates } \beta \text{ when } \emptyset$, and the notation $\alpha \text{ terminates } \beta$ is an abbreviation for $\alpha \text{ terminates } \beta \text{ when } \emptyset$.

Example 20. We represent the example of turning on and off a light using the following $\mathcal{E}$ domain description:

$$\text{TurnOn initiates On}$$
$$\text{TurnOff terminates On}$$
$$\neg \text{On holds-at 0}$$
$$\text{TurnOn happens-at 2}$$
$$\text{TurnOff happens-at 4}$$

This domain description entails the following:

$$\neg \text{On holds-at 1}$$
$$\text{On holds-at 3}$$
$$\neg \text{On holds-at 5}$$

Kakas and Miller [35, 36] specify the semantics of $\mathcal{E}$ using simple definitions of structures and models. Miller and Shanahan [66] show that $\mathcal{E}$ corresponds to the EC of Section 1.2.4 without the predicates ReleasedAt, Releases, Trajectory, and AntiTrajectory. They define conditions under which an $\mathcal{E}$ domain description matches an EC domain description and prove that, if an $\mathcal{E}$ domain description matches an EC domain description, the domain descriptions entail the same fluent truth values. Dimopoulos, Kakas, and Michael [13] give a translation of $\mathcal{E}$ domain descriptions into answer set programs [3, 20, 21] (see also Chapter ??).
An \(E\) domain description can be translated into an EC or DEC domain description as follows. We assume that the timepoints are the integers and the partial order is \(\leq\). We divide the \(E\) domain description into sets of holds-at, happens-at, initiates, and terminates statements. We translate each holds-at statement

\[\neg \beta \text{ holds-at } \tau\]

into the formula

\[\neg \text{HoldsAt}(\beta, \tau)\]

We translate the set of happens-at statements

\[\alpha_1 \text{ happens-at } \tau_1\]
\[\vdots\]
\[\alpha_n \text{ happens-at } \tau_n\]

into the formula

\[\text{Happens}(e, t) \equiv (e = \alpha_1 \land t = \tau_1) \lor \ldots \lor (e = \alpha_n \land t = \tau_n)\]

We translate the set of initiates/terminates statements

\[\alpha_1 \text{ initiates/terminates } \beta_1 \text{ when } \neg \gamma_1, 1, \ldots, \neg \gamma_1, p\]
\[\vdots\]
\[\alpha_n \text{ initiates/terminates } \beta_n \text{ when } \neg \gamma_n, 1, \ldots, \neg \gamma_n, q\]

into the formula

\[\text{Initiates/Terminates}(e, f, t) \equiv\]
\[(e = \alpha_1 \land f = \beta_1 \land \neg \text{HoldsAt}(\gamma_1, 1, t) \land \ldots \land \neg \text{HoldsAt}(\gamma_1, p, t)) \lor \ldots \lor\]
\[(e = \alpha_n \land f = \beta_n \land \neg \text{HoldsAt}(\gamma_n, 1, t) \land \ldots \land \neg \text{HoldsAt}(\gamma_n, q, t))\]

An extension to \(E\) [35] provides support for indirect effects. The statement

\[\gamma \text{ whenever } \Gamma\]

where \(\gamma\) is a fluent literal and \(\Gamma\) is a set of fluent literals represents that (1) \(\gamma\) holds at every timepoint at which \(\Gamma\) holds, and (2) every event occurrence that brings about \(\Gamma\) also brings about \(\gamma\). The language \(E\) has been further developed into the language \textit{Modular-}\(E\) [34], which addresses the ramification and qualification problems along with the issues of elaboration tolerance and modularity.

### 1.7 Automated Event Calculus Reasoning

A number of techniques can be used to perform automated reasoning in the event calculus, including logic programming in Prolog, answer set programming, satisfiability solving, and first-order logic automated theorem proving. Table 1.5 provides pointers to online resources for event calculus reasoning.
Table 1.5: Online resources for automated event calculus reasoning

<table>
<thead>
<tr>
<th>Resource</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><a href="http://www.tcs.hut.fi/Software/smodels/">http://www.tcs.hut.fi/Software/smodels/</a> (solver)</td>
</tr>
<tr>
<td>E-RES [37, 38]</td>
<td><a href="http://www2.cs.ucy.ac.cy/~pslogic/">http://www2.cs.ucy.ac.cy/~pslogic/</a></td>
</tr>
</tbody>
</table>

1.7.1 Prolog

The original event calculus was formulated as a logic program, and logic programming in Prolog can be used to perform event calculus reasoning. If Prolog is used, however, special care must be taken to avoid infinite loops [10, 45, 87, 93, 94, 98, 103]. Event calculus reasoning can be performed through abductive logic programming [6, 12, 17, 103].

1.7.2 Answer Set Programming

Answer set solvers [3] such as smodels [79] can be used to solve event calculus deduction problems [73]. Answer set solvers can also be used for reasoning in the E language [13].

1.7.3 Satisfiability (SAT) Solving

As a result of the growth in the capabilities of propositional satisfiability (SAT) solvers [92], several event calculus reasoning programs have been built that exploit off-the-shelf SAT solvers. The program of Shanahan and Witkowski [109] solves planning problems using SAT solvers. The Discrete Event Calculus Reasoner [70, 71] uses SAT solvers to perform various types of event calculus reasoning including deduction, abduction, postdiction, and model finding. The E-RES program [37, 38] for solving E reasoning problems uses SAT solvers to generate classical models of state constraints.

The Discrete Event Calculus Reasoner uses several techniques to reduce the size of the SAT encoding of event calculus problems [70]:

1. The domains of arguments to predicates are restricted by using many-sorted logic.
2. Atom definitions are expanded [80, p. 361] in order to eliminate a large number of Initiates, Terminates, Releases, Trajectory, and AntiTrajectory ground atoms.
3. Triply quantified time is eliminated from most event calculus axioms by using DEC [70].
4. A compact conjunctive normal form is computed using the technique of renaming subformulas [27, 80, 83].
Erik T. Mueller

The Discrete Event Calculus Reasoner distribution includes a library of 99 event calculus reasoning problems that can be solved using the program.

1.7.4 First-Order Logic Automated Theorem Proving

Although first-order logic entailment is undecidable, first-order logic automated theorem proving (ATP) systems [86] have been applied successfully to event calculus deduction problems [77, 78]. But in some cases, the systems require human guidance in the form of lemmas. Event calculus problems are included in the TPTP problem library [110] along with the results of running ATP systems on them.

1.8 Applications of the Event Calculus

An important area of application of the event calculus is commonsense reasoning [74]. Event calculus formalizations have been developed for a number of commonsense domains, including beliefs [46], egg cracking [68, 99, 106], emotions [74], goals and plans [74], object identity [74], space [68, 96], and the zoo world [1, 33, 71]. The event calculus has also been used to model electronic circuits [101] and water tanks [64].

The event calculus can be applied to problems in high-level cognition including natural language understanding and vision. It has been used to build models of story events and states in space and time [69, 72, 76], represent the semantics of natural language tense and aspect [113], and represent event occurrences in stories [31]. The event calculus has been used to implement the higher-level vision component of an upper-torso humanoid robot [102, 107, 108].

Another application area of the event calculus is business systems. The event calculus has been used to track the state of contracts for performance monitoring [18], to model workflows [10, 114], and to improve the flexibility of applications that use electronic payment systems [115]. Other applications of the event calculus include database updates [41], planning [12, 17, 67, 97, 103, 109], and representing legislation [42].

Bibliography


[21] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and


[53] Fangzhen Lin. Embracing causality in specifying the indirect effects of actions. In


[100] Murray Shanahan. The event calculus explained. In Michael J. Wooldridge and


