Abstract

We add branching time to the linear discrete event calculus, which yields a formalism for commonsense reasoning that combines the benefits of the situation calculus and the event calculus. We show how the branching discrete event calculus can be used to solve commonsense reasoning problems involving hypothetical events, concurrent events with cumulative and canceling effects, and triggered events.

Introduction

The classical logic event calculus (Miller & Shanahan 2002; Shanahan 1997) can serve as a foundation for commonsense reasoning. It can be used to reason about important areas of the commonsense world including action and change (Shanahan 1999a), space (Morgenstern 2001; Shanahan 1996; 2004), and mental states (Mueller 2006a). An important aspect of commonsense reasoning is reasoning about hypothetical events. Unlike the situation calculus (McCarthy & Hayes 1969), the classical logic event calculus typically uses a linear time structure (Miller & Shanahan 2002, p. 453) and does not handle hypothetical events (Shanahan 1997, p. 364).

In this paper, we show how a version of the classical logic event calculus can be modified to yield a new formalism that combines the benefits of the situation calculus and the event calculus. Like the situation calculus, the new formalism supports reasoning about hypothetical events. Like the classical logic event calculus, the new formalism supports reasoning about the commonsense law of inertia, release from the commonsense law of inertia, concurrent events with cumulative and canceling effects, context-sensitive effects, indirect effects, nondeterministic effects, preconditions, and triggered events.

We start with the linear discrete event calculus (LDEC) (Mueller 2004a; 2006a), a discrete version of the classical logic event calculus. LDEC has been proved logically equivalent to the continuous event calculus for integer time (Mueller 2004a), and to temporal action logics (Doherty et al. 1998) for inertial fluents and single-step actions (Mueller 2006b). We modify LDEC to obtain the branching discrete event calculus (BDEC) by (1) removing the requirement that every situation must have a unique successor, and (2) adding an argument for successor situation to Happens, Initiates, Terminates, and Releases. We show how BDEC can be used to solve commonsense reasoning problems involving hypothetical events, concurrent events, and triggered events. We extend BDEC to distinguish between hypothetical and actual situations and events. In an extended version of this paper (Mueller in submission), we characterize the precise relationship between LDEC and BDEC, and prove that a restricted version of BDEC is equivalent to a version of the situation calculus.

The discrete event calculus was developed to facilitate automated event calculus reasoning. It simplifies the classical logic event calculus axioms, reducing the number of axioms from 17 to 12 and eliminating triply quantified time from most axioms. The discrete event calculus is the basis for the Discrete Event Calculus Reasoner program for automated commonsense reasoning (Mueller 2004b). We have implemented BDEC within this program, extending it with the ability to reason about hypothetical events.

Linear Discrete Event Calculus

We use many-sorted languages with equality. The linear discrete event calculus has sorts for events, fluents, and situations. In an extended version of this paper (Mueller in submission), we characterize the precise relationship between LDEC and BDEC, and prove that a restricted version of BDEC is equivalent to a version of the situation calculus.

The discrete event calculus was developed to facilitate automated event calculus reasoning. It simplifies the classical logic event calculus axioms, reducing the number of axioms from 17 to 12 and eliminating triply quantified time from most axioms. The discrete event calculus is the basis for the Discrete Event Calculus Reasoner program for automated commonsense reasoning (Mueller 2004b). We have implemented BDEC within this program, extending it with the ability to reason about hypothetical events.

The discrete event calculus was developed to facilitate automated event calculus reasoning. It simplifies the classical logic event calculus axioms, reducing the number of axioms from 17 to 12 and eliminating triply quantified time from most axioms. The discrete event calculus is the basis for the Discrete Event Calculus Reasoner program for automated commonsense reasoning (Mueller 2004b). We have implemented BDEC within this program, extending it with the ability to reason about hypothetical events.

Copyright © 2007, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
will be released from the commonsense law of inertia at the successor of \( s \).

The commonsense law of inertia (Shanahan 1997) states that a fluent’s truth value persists unless the fluent is affected by an event. When a fluent is released from this law, its truth value can fluctuate. Fluids that are released from the commonsense law of inertia can be used to model nondeterministic effects (Shanahan 1999a) and indirect effects (Shanahan 1999b).  

Let LDEC be the conjunction of the following axioms:

LDEC1. \( S_0(e) \neq S_0 \)
LDEC2. \( S(L_1(s_1)) = S(L_2(s_2)) \rightarrow s_1 = s_2 \)
LDEC3. \( \forall P ((P(S_0) \land \forall s (P(s) \rightarrow P(S(L_1(s)))) \rightarrow \forall s P(s)) \)
LDEC4. \( \neg \exists e (\text{Happens}(e, s) \land \text{Terminates}_L(e, f, s) \rightarrow \text{HoldsAt}(f, S(L_1(s))) \)
LDEC5. \( \neg \exists e (\text{Happens}_L(e, s) \land \text{Initiates}_L(e, f, s) \rightarrow \neg \text{HoldsAt}(f, S(L_1(s))) \)
LDEC6. \( \text{ReleasedAt}(f, s) \land \neg \exists e (\text{Happens}_L(e, s) \land \text{Initiates}_L(e, f, s) \rightarrow \neg \text{HoldsAt}(f, S(L_1(s))) \)
LDEC7. \( \neg \exists e (\text{Happens}(e, s) \land \text{Terminates}(e, f, s) \rightarrow \text{ReleasedAt}(f, S(L_1(s))) \)
LDEC8. \( \text{Happens}_L(e, s) \land \text{Initiates}_L(e, f, s) \rightarrow \text{HoldsAt}(f, S(L_1(s))) \)
LDEC9. \( \text{Happens}_L(e, s) \land \text{Terminates}_L(e, f, s) \rightarrow \neg \text{HoldsAt}(f, S(L_1(s))) \)
LDEC10. \( \text{Happens}_L(e, s) \land \text{Released}_L(e, f, s) \rightarrow \text{ReleasedAt}(f, S(L_1(s))) \)
LDEC11. \( \text{Happens}_L(e, s) \land \text{Initiates}_L(e, f, s) \rightarrow \neg \text{ReleasedAt}(f, S(L_1(s))) \)

Axioms LDEC1 through LDEC3 are the Peano axioms for the nonnegative integers. Axiom LDEC3 is a second-order axiom of induction.

**Branching Discrete Event Calculus**

We extend LDEC to branching time as follows. We replace the function \( S_1 \) with a relation \( S \), and allow a situation to have zero or more successors. We add an argument to \( \text{Happens} \), \( \text{Initiates} \), \( \text{Terminates} \), and \( \text{Released} \) specifying the successor situation. It is important to add the successor situation to \( \text{Initiates} \), \( \text{Terminates} \), and \( \text{Released} \) to treat concurrent events with cumulative and canceling effects (see below).

The language has the constant \( S_0 \) denoting the initial situation, and the following predicates:

- \( S(s_1, s_2) \): Situation \( s_2 \) is a successor of situation \( s_1 \).
- \( \text{Happens}(e, s_1, s_2) \): Event \( e \) occurs between situation \( s_1 \) and situation \( s_2 \).
- \( \text{HoldsAt}(f, s) \): Fluent \( f \) is true at situation \( s \).
- \( \text{ReleasedAt}(f, s) \): Fluent \( f \) is released from the commonsense law of inertia at situation \( s \).
- \( \text{Initiates}(e, f, s_1, s_2) \): Event \( e \) occurs between situation \( s_1 \) and situation \( s_2 \), but fluent \( f \) will be true and not released from the commonsense law of inertia at \( s_2 \).
- \( \text{Terminates}(e, f, s_1, s_2) \): Event \( e \) occurs between situation \( s_1 \) and situation \( s_2 \), but fluent \( f \) will be false and not released from the commonsense law of inertia at \( s_2 \).

**Commonsense Reasoning Problems**

BDEC can be used to reason about (1) hypothetical events as in the situation calculus, and (2) phenomena of action and change as in the event calculus. In this section, we show how BDEC can be used to perform commonsense reasoning about three scenarios: the hypothetical Yale shooting scenario, the soup bowl scenario, and the reactive cat scenario.

**Hypothetical Yale Shooting Scenario** Van Belleghem, De Necker, and De Schreye (1997) describe the following problem of hypothetical reasoning, which is based on the Yale shooting scenario (Hanks & McDermott 1987). A turkey, which was initially alive, was shot by a person. It is not known whether the gun was loaded. But it is known that, if the person had waited instead of shooting, the gun would have been loaded afterward. The problem is to infer that the gun was initially loaded, and that the turkey died. Van Belleghem et al. argue that this problem can be solved

\[ \text{See the proof of Proposition 2.1 of Pinto (1994, pp. 103–105).} \]
using the situation calculus, but not by the event calculus. We show how BDEC can be used to solve this problem.

We use a domain theory similar to that of Shanahan (1997, pp. 322–323). We have three events, Load, Shoot, and Wait, and two fluents, Alive and Loaded. If a gun is loaded, then it will be loaded:

\[ \text{Initiates}(\text{Load}, \text{Loaded}, s_1, s_2) \]  
(1)

If a gun is loaded and the gun is shot, then the victim will no longer be alive:

\[ \text{HoldsAt}(\text{Loaded}, s_1) \rightarrow \text{Terminates}(\text{Shoot}, \text{Alive}, s_1, s_2) \]  
(2)

If a gun is shot, then it will no longer be loaded:

\[ \text{Terminates}(\text{Shoot}, \text{Loaded}, s_1, s_2) \]  
(3)

We consider the following narrative. The victim is initially alive:

\[ \text{HoldsAt}(\text{Alive}, S_0) \]  
(4)

The gun is shot between situation \( S_0 \) and situation \( S_1 \):

\[ \text{Happens}(\text{Shoot}, S_0, S_1) \]  
(5)

We add the following hypothetical information. If the person had waited between situation \( S_0 \) and situation \( S_2 \), then the gun would have been loaded at \( S_2 \):

\[ \text{Happens}(\text{Wait}, S_0, S_2) \]  
(6)

\[ \text{HoldsAt}(\text{Loaded}, S_2) \]  
(7)

The events and fluents are distinct:

\[ \text{Load} \neq \text{Shoot} \]  
(8)

\[ \text{Shoot} \neq \text{Wait} \]  
(9)

\[ \text{Load} \neq \text{Wait} \]  
(10)

\[ \text{Alive} \neq \text{Loaded} \]  
(11)

Fluents are never released from the commonsense law of inertia:

\[ \neg \text{ReleasedAt}(f, s) \]  
(12)

We can then show that the gun was loaded at \( S_0 \) and that the victim was dead at \( S_1 \).

Just as in the classical logic event calculus, we use the nonmonotonic method of circumscription (Lifschitz 1994; McCarthy 1980) for default reasoning about time. We circumscribe Initiates, Terminates, and Releases to minimize unexpected effects of events, and we circumscribe Happens to minimize unexpected events.

**Proposition 1.** Let \( \Sigma = (1) \land (2) \land (3), \Delta = (5) \land (6), \Omega = (8) \land (9) \land (10) \land (11) \), and \( \Gamma = (4) \land (7) \land (12) \). Then we have

\[ \text{CIRC}[\Sigma; \text{Initiates}, \text{Terminates}, \text{Releases}] \land \]

\[ \text{CIRC}[\Delta; \text{Happens}] \land \Omega \land \Gamma \land \text{BDEC} \]

\[ \vdash \text{HoldsAt}(\text{Loaded}, S_0) \land \neg \text{HoldsAt}(\text{Alive}, S_1). \]

**Proof.** From CIRC[\Sigma; \text{Initiates}, \text{Terminates}, \text{Releases}] and Propositions 2 and 14 of Lifschitz (1994) reducing circumscription to predicate completion and reducing parallel circumscription to basic circumscription, we have

\[ \text{Initiates}(e, f, s_1, s_2) \leftrightarrow (e = \text{Load} \land f = \text{Loaded}) \]  
(13)

\[ \text{Terminates}(e, f, s_1, s_2) \leftrightarrow (e = \text{Shoot} \land f = \text{Alive} \land \text{HoldsAt}(\text{Loaded}, s_1)) \lor \]

\[ (e = \text{Shoot} \land f = \text{Loaded}) \]

\[ \neg \text{Releases}(e, f, s_1, s_2) \]  
(15)

From CIRC[\Delta; \text{Happens}] and Proposition 2 of Lifschitz, we have

\[ \text{Happens}(e, s_1, s_2) \leftrightarrow (e = \text{Shoot} \land s_1 = S_0 \land s_2 = S_1) \lor \]

\[ (e = \text{Wait} \land s_1 = S_0 \land s_2 = S_2) \]  
(16)

Seeking a contradiction, suppose that

\[ \neg \text{HoldsAt}(\text{Loaded}, S_0) \]  
(17)

From (13), (16), (8), and (10), we have

\[ \neg \exists e (\text{Happens}(e, s_0, s_2) \land \text{Initiates}(e, \text{Loaded}, s_0, s_2)). \]

From this, \( S(S_0, S_2) \) (which follows from (16) and BDEC12), (17), (12), and BDEC5, we have

\[ \neg \text{HoldsAt}(\text{Loaded}, S_2) \]

which contradicts (7). Therefore,

\[ \text{HoldsAt}(\text{Loaded}, S_0). \]

From this and (14), we have

\[ \text{Terminates}(\text{Shoot}, \text{Alive}, S_0, S_1). \]

From this, \( \text{Happens}(\text{Shoot}, S_0, S_1) \) (which follows from (16)), and BDEC9, we have \( \neg \text{HoldsAt}(\text{Alive}, S_1). \) □

**Soup Bowl Scenario** Gelfond, Lifschitz, and Rabinov (1991) describe the following soup bowl scenario. A person is trying to lift a bowl of soup. The problem is to infer that, if the person lifts the bowl with one hand, then the soup spills, whereas, if the person lifts the bowl with both hands, then the soup does not spill. Miller and Shanahan (2002, pp. 460–461) have formalized this problem in the classical logic event calculus. We show that their formalization works in BDEC as well. By using BDEC, we are able to consider two hypothetical alternatives.

If the bowl is lifted with both hands, then it will be raised:

\[ \text{Happens}(\text{LiftLeft}, s_1, s_2) \rightarrow \text{Initiates}(\text{LiftRight}, \text{Raised}, s_1, s_2) \]  
(18)

If the bowl is only lifted with one hand, then it will be spilled:

\[ \neg \text{Happens}(\text{LiftRight}, s_1, s_2) \rightarrow \text{Initiates}(\text{LiftLeft}, \text{Spilled}, s_1, s_2) \]  
(19)

\[ \neg \text{Happens}(\text{LiftLeft}, s_1, s_2) \rightarrow \text{Initiates}(\text{LiftRight}, \text{Spilled}, s_1, s_2) \]  
(20)

Initially, the bowl is not raised and not spilled:

\[ \neg \text{HoldsAt}(\text{Raised}, S_0) \]  
(21)

\[ \neg \text{HoldsAt}(\text{Spilled}, S_0) \]  
(22)
We consider two alternatives. The first alternative is that the bowl is lifted with both hands:

\begin{align*}
\text{Happens}(\text{LiftLeft}, S_0, S_1) \quad (23) \\
\text{Happens}(\text{LiftRight}, S_0, S_1) \quad (24)
\end{align*}

We can show that the bowl will be raised and not spilled. The second alternative is that the bowl is lifted with the right hand:

\begin{align*}
\text{Happens}(\text{LiftRight}, S_0, S_2) \quad (25)
\end{align*}

We can show that the bowl will be spilled and not raised.

The events and fluents are distinct:

\begin{align*}
\text{LiftLeft} \neq \text{LiftRight} \quad (26) \\
\text{ Raised} \neq \text{ Spilled} \quad (27)
\end{align*}

Situations \(S_1\) and \(S_2\) are distinct:

\begin{align*}
S_1 \neq S_2 \quad (28)
\end{align*}

Fluents are never released from the commonsense law of inertia:

\begin{align*}
\neg \text{ReleasedAt}(f, s) \quad (29)
\end{align*}

**Proposition 2.** Let \(\Sigma = (18) \land (19) \land (20), \quad \Delta = (23) \land (24) \land (25), \quad \Omega = (26) \land (27) \land (28)\), and \(\Gamma = (21) \land (22) \land (29)\). Then we have

\begin{align*}
\text{CIRC}[\Sigma; \text{Initiates, Terminates, Releases}] \land \\
\text{CIRC}[\Delta; \text{Happens}] \land \Omega \land \Gamma \land \text{BDEC} \\
\vdash \text{HoldsAt}(\text{Raised}, S_1) \land \neg \text{HoldsAt}(\text{Spilled}, S_1) \land \\
\text{HoldsAt}(\text{Spilled}, S_2) \land \neg \text{HoldsAt}(\text{Raised}, S_2).
\end{align*}

**Proof.** From \(\text{CIRC}[\Sigma; \text{Initiates, Terminates, Releases}]\) and Propositions 2 and 14 of Lifschitz (1994), we have

\begin{align*}
\text{Initiates}(e, f, s_1, s_2) \quad (30) \\
(e = \text{LiftLeft} \land f = \text{Raised} \land \\
\text{Happens}(\text{LiftLeft}, s_1, s_2)) \lor \\
(e = \text{LiftLeft} \land f = \text{Spilled} \land \\
\neg \text{Happens}(\text{LiftRight}, s_1, s_2)) \lor \\
(e = \text{LiftRight} \land f = \text{Spilled} \land \\
\neg \text{Happens}(\text{LiftLeft}, s_1, s_2)) \\
\neg \text{Terminates}(e, f, s_1, s_2) \quad (31) \\
\neg \text{Released}(e, f, s_1, s_2) \quad (32)
\end{align*}

From \(\text{CIRC}[\Delta; \text{Happens}]\) and Proposition 2 of Lifschitz, we have

\begin{align*}
\text{Happens}(e, s_1, s_2) \quad (33) \\
(e = \text{LiftLeft} \land s_1 = S_0 \land s_2 = S_1) \lor \\
(e = \text{LiftRight} \land s_1 = S_0 \land s_2 = S_1) \lor \\
(e = \text{LiftRight} \land s_1 = S_0 \land s_2 = S_2)
\end{align*}

From \(\text{Happens}(\text{LiftLeft}, S_0, S_1)\) (which follows from (33)) and (30), we have \(\text{Initiates}(\text{LiftRight, Raised}, S_0, S_1)\). From this, \(\text{Happens}(\text{LiftRight, Spilled, S_0, S_1})\) (which follows from (33)), and BDEC8, we have \(\text{HoldsAt}(\text{Raised}, S_1)\).

From (30), (33), (26), (27), and (28), we have \(\neg \exists e ((\text{Happens}(e, S_0, S_1) \land \text{Initiates}(e, \text{Spilled}, S_0, S_1))\). From this, \(S(S_0, S_1)\) (which follows from (33) and BDEC12), (22), (29), and BDEC5, we have \(\neg \text{HoldsAt}(\text{Spilled}, S_1)\).

From \(\neg \text{Happens}(\text{LiftLeft}, S_0, S_2)\) (which follows from (33), (26), and (28)), and (30), we have \(\text{Initiates}(\text{LiftRight, Spilled, S_0, S_2})\). From this, \(S(S_0, S_2)\) (which follows from (33) and BDEC12), (21), (29), and BDEC5, we have \(\neg \text{HoldsAt}(\text{Raised}, S_2)\).

**Reactive Cat Scenario** Consider a cat that eats food whenever food is present. If food is present at situation \(s_1\), then the food is eaten between \(s_1\) and every successor situation \(s_2\) of \(s_1\):

\begin{align*}
S(s_1, s_2) \land \text{HoldsAt}(\text{FoodPresent}, s_1) \to (34) \\
\text{Happens}(\text{EatFood}, s_1, s_2)
\end{align*}

If the food is eaten, then it will no longer be present:

\begin{align*}
\text{Terminates}(\text{EatFood, FoodPresent, s_1, s_2}) \quad (35)
\end{align*}

Food is initially present:

\begin{align*}
\text{HoldsAt}(\text{FoodPresent}, S_0) \quad (36)
\end{align*}

The initial situation has one successor situation:

\begin{align*}
S(S_0, S_1) \quad (37)
\end{align*}

We can then show that the cat will eat the food and that the food will no longer be present afterward.

**Proposition 3.** Let \(\Sigma = (35), \quad \Delta = (34), \quad \Upsilon = (37), \quad \text{and} \quad \Gamma = (36)\). Then we have

\begin{align*}
\text{CIRC}[\Sigma; \text{Initiates, Terminates, Releases}] \land \\
\text{CIRC}[\Delta; \text{Happens}] \land \Upsilon \land \Gamma \land \text{BDEC} \\
\vdash \text{Happens}(\text{EatFood, S_0, S_1}) \land \\
\neg \text{HoldsAt}(\text{FoodPresent, S_1}).
\end{align*}

**Proof.** From \(\text{CIRC}[\Sigma; \text{Initiates, Terminates, Releases}]\) and Propositions 2 and 14 of Lifschitz (1994), we have

\begin{align*}
\neg \text{Initiates}(e, f, s_1, s_2) \land \\
\neg \text{Terminates}(e, f, s_1, s_2) \land \\
\neg \text{Released}(e, f, s_1, s_2)
\end{align*}

From \(\text{CIRC}[\Delta; \text{Happens}]\) and Proposition 2 of Lifschitz, we have

\begin{align*}
\text{Happens}(e, s_1, s_2) \quad (33) \\
(e = \text{LiftLeft} \land s_1 = S_0 \land s_2 = S_1) \lor \\
(e = \text{LiftRight} \land s_1 = S_0 \land s_2 = S_1) \lor \\
(e = \text{LiftRight} \land s_1 = S_0 \land s_2 = S_2)
\end{align*}

From \(\text{Happens}(\text{LiftLeft, S_0, S_1})\) (which follows from (33)) and (30), we have \(\text{Initiates}(\text{LiftRight, Raised, S_0, S_1})\). From this, \(\text{Happens}(\text{LiftRight, Spilled, S_0, S_1})\) (which follows from (33)), and BDEC8, we have \(\text{HoldsAt}(\text{Raised, S_1})\).

From (30), (33), (26), (27), and (28), we have \(\neg \exists e ((\text{Happens}(e, S_0, S_1) \land \text{Initiates}(e, \text{Spilled}, S_0, S_1))\). From this, \(S(S_0, S_1)\) (which follows from (33) and BDEC12), (22), (29), and BDEC5, we have \(\neg \text{HoldsAt}(\text{Spilled, S_1})\).

From \(\neg \text{Happens}(\text{LiftLeft, S_0, S_2})\) (which follows from (33), (26), and (28)), and (30), we have \(\text{Initiates}(\text{LiftRight, Spilled, S_0, S_2})\). From this, \(S(S_0, S_2)\) (which follows from (33) and BDEC12), (21), (29), and BDEC5, we have \(\neg \text{HoldsAt}(\text{Raised, S_2})\).
Actual Situations and Events

In this section, we extend BDEC to distinguish between hypothetical and actual situations, and to distinguish between hypothetical and actual event occurrences, along the lines of Pinto and Reiter (1995). We add the predicate Actual(s), which represents that s is an actual situation, and the predicate ActuallyHappens(e, s), which represents that event e actually occurs in situation s. We add several axioms. If a situation is actual, then a predecessor of that situation is actual:

\[ \text{Actual}(s_2) \land S(s_1, s_2) \rightarrow \text{Actual}(s_1) \] (42)

A situation has at most one actual successor:

\[ S(s, s_1) \land S(s, s_2) \land \text{Actual}(s_1) \land \text{Actual}(s_2) \rightarrow s_1 = s_2 \] (43)

An event actually occurs in a situation if and only if the event occurs between the situation and some actual situation:

\[ \text{ActuallyHappens}(e, s_1) \leftrightarrow \exists s_2 (\text{Happens}(e, s_1, s_2) \land \text{Actual}(s_2)) \] (44)

Consider again the hypothetical Yale shooting scenario. We need only specify that S1 is an actual situation:

\[ \text{Actual}(S_1) \] (45)

From this, S(S0, S1) (which follows from (16) and BDEC12), and (42), we have \( \text{Actual}(S_0) \). From S(S0, S1), S(S0, S2) (which follows from (16) and BDEC12), (45), and (43), we have \( \neg \text{Actual}(S_2) \). From this, (16), and (44), we have \( \neg \text{ActuallyHappens}(\text{Wait}, S_0) \). From Happens(Shoot, S0, S1) (which follows from (16)), (45), and (44), we have \( \text{ActuallyHappens}(\text{Shoot}, S_0) \).

Related Work

The main difference between BDEC and other proposals for combining the event calculus and situation calculus is that BDEC considers situations and timepoints to be one and the same. Other proposals define situations differently from timepoints.

Provetti (1996) proposes a hybrid of the event calculus and the situation calculus. Every timepoint is associated with a situation, but not every situation is associated with a timepoint. Every timepoint in the linear timeline of the event calculus corresponds to the root of a tree of hypothetical situations in the situation calculus. Unlike BDEC, the formalism does not handle hypothetical reasoning about concurrent actions.

Kowalski and Sadri (1997) propose a version of the event calculus with branching time and situations. Unlike BDEC, this variant disallows concurrent events, and it identifies timepoints with both situations and transitions between situations.

Van Belleghem, Denecker, and De Schreye (1997) propose a formalism that extends both the event calculus and the situation calculus. They start with a version of the event calculus with the linear time axiom \( T_1 < T_2 \lor T_1 = T_2 \lor T_2 < T_1 \). They then replace this axiom with the branching time axiom \( (T_1 < T_2 \land T_2 < T_3) \rightarrow (T_1 < T_2 \lor T_1 = T_2 \lor T_2 < T_1) \). They then define a situation started by an event at timepoint \( T_1 \) as the set of timepoints \( T_2 \) after \( T_1 \) such that there are no events between \( T_1 \) and \( T_2 \). Unlike BDEC, this formalism disallows concurrent events, and defines situations as sets of timepoints.

Lévy and Quantz (1998) propose an extension of the event calculus in which a situation argument is added to every event calculus predicate, and the event calculus axioms are modified accordingly. This formalism does not have a successor relation between situations. Situations are instead related to one another by a predicate Equal until \((s_1, s_2, t)\), which represents that situations \( s_1 \) and \( s_2 \) are equal until time \( t \).

The primary difference between BDEC and the situation calculus is that BDEC separates the do function into the two predicates \( S \) and \( \text{Happens} \). BDEC is different from the formulation of the situation calculus of Reiter and colleagues (Lin & Reiter 1994; Pinto 1994; Pirri & Reiter 1999; Reiter 1993; 2001). Pirri and Reiter (1999) provide the following foundational axioms for the situation calculus:

\[ \text{do}(a_1, s_1) = \text{do}(a_2, s_2) \supset a_1 = a_2 \land s_1 = s_2 \] (46)
\[ \forall a, s [P(s) \supset P(\text{do}(a, s))] \supset (\forall s) P(s) \] (47)
\[ \neg(s \sqsubseteq s_0) \] (48)
\[ s \sqsubseteq \text{do}(a, s') \equiv s' = s \lor s \sqsubseteq s' \] (49)

In any model of these axioms, every situation has a successor for every element of the action domain. In any model of BDEC, if an event occurs between situation \( s_1 \) and \( s_2 \), then \( s_1 \) has \( s_2 \) as a successor (see axiom BDEC12).

Whereas BDEC allows more than one event between two situations, the situation calculus of Reiter et al. does not. In BDEC, we can write \( \text{Happens}(E_1, S_0, S_1) \land \text{Happens}(E_2, S_0, S_1) \land E_1 \neq E_2 \). The analogous situation calculus formula \( S_1 = \text{do}(E_1, S_0) \land S_1 = \text{do}(E_2, S_0) \land E_1 \neq E_2 \) is inconsistent with axiom (46). Multiple events between situations are also ruled out by Davis’s (1994) axiom \( \text{result}(S_1, E_A, S_2) \land \text{result}(S_1, E_B, S_2) \rightarrow E_A = E_B \).

Based on previous proposals, Reiter (1996) adds concurrent actions to the situation calculus by defining a concurrent action as a set of simple actions, and redefining \text{do} \text{ on work on concurrent actions}. This enables representation of multiple events between situations. One can write \( S_1 = \text{do}([A_1, A_2], S_0) \). Whereas BDEC allows zero events between two situations, Reiter requires a concurrent action to contain at least one simple action. (Zero actions between situations can be simulated in the situation calculus using an action that has no effects.)

BDEC is different from McCarthy’s (1997; 2002) formulation of the situation calculus with concurrent events and narratives. In this formulation, the predicate \( \text{Occurs}(e, s) \) represents that event \( e \) occurs in situation \( s \), the function \( \text{Next}(s) \) represents the next situation after \( s \), and the function \( \text{Result}(e, s) \) represents the situation that results from \( e \) occurring in situation \( s \). These are related via the axiom

\[ \text{Occurs}(e, s) \rightarrow \text{Next}(s) = \text{Result}(e, s) \] (50)
We can represent that two events occur in $S_0$: \( \text{Occurs}(E_1, S_0) \land \text{Occurs}(E_2, S_0) \land E_1 \neq E_2 \). From this and (50), we have \( \text{Next}(S_0) = \text{Result}(E_1, S_0) \) and \( \text{Next}(S_0) = \text{Result}(E_2, S_0) \). Thus as in BDEC, we can have distinct events $E_1$ and $E_2$ between two situations:

\[
S_1 = \text{Result}(E_1, S_0) = \text{Result}(E_2, S_0) \quad (51)
\]

But we cannot then represent hypothetical events $E_1$ and $E_3$ that occur between $S_0$ and some other situation $S_2$:

\[
S_2 = \text{Result}(E_1, S_0) \quad (52) \\
S_2 = \text{Result}(E_3, S_0) \quad (53) \\
S_1 \neq S_2 \quad (54)
\]

The formulas (52) and (54) are inconsistent with (51). To allow hypothetical reasoning, McCarthy (1997) adds a narrative argument to \( \text{Occurs} \), and uses contexts (Guha 1992; McCarthy 1993) to reason with \( \text{Occurs}(e, s, n) \) inside a narrative. We rewrite (50) as \( \text{Occurs}(e, s, n) \rightarrow \text{Next}(s, n) = \text{Result}(e, s, n) \) and use lifting formulas such as \( \text{SpecializeNarrative}(e, c', e) \land \text{Ist}(c', \text{Occurs}(e, s)) \rightarrow \text{Ist}(c, \text{Occurs}(e, s, n)) \). Nossum and Thielscher (1999) propose to use contexts in the event calculus.

Conclusions

We introduced BDEC, a branching time discrete version of the classical logic event calculus. BDEC is useful for commonsense reasoning about hypothetical events as well as other phenomena of action and change. A catalog of commonsense phenomena treated by the classical logic event calculus is provided by Mueller (2006a).

Some areas for further work are the following:

- BDEC allows two identical sets of event occurrences to lead to two different situations. For example, $E_1$ and $E_2$ could lead from $S_0$ to $S_1$, and $E_1$ and $E_2$ could also lead from $S_0$ to a situation $S_2$ distinct from $S_1$. The following axiom to rule this out could be added: $\forall e \ (\text{Happens}(e, s, s_1) \leftrightarrow \text{Happens}(e, s, s_2)) \rightarrow s_1 = s_2$.

- To allow reasoning about gradual change, the Trajectory and AntiTrajectory predicates (Miller & Shanahan 2002) could be added to BDEC. This requires definition of the distance between two situations along a path.

References


