# Event Calculus Reasoning Through Satisfiability

ERIK T. MUELLER, *IBM Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598 USA.*
*E-mail: etm@us.ibm.com*

## Abstract

We present an implemented method for encoding reasoning problems of a discrete version of the classical logic event calculus in propositional conjunctive normal form, enabling the problems to be solved efficiently by off-the-shelf complete satisfiability (SAT) solvers. We build on the previous encoding method of Shanahan and Witkowski, extending it to support causal constraints, concurrent events, determining fluents, effect axioms with conditions, events triggered by conditions, gradual change, incompletely specified initial situations, state constraints, and release from the commonsense law of inertia. We present an alternative classical logic axiomatization of the event calculus and prove its equivalence to a standard axiomatization for integer time. We describe our encoding method based on the alternative axiomatization and prove its correctness. We evaluate the method on 14 benchmark reasoning problems for the event calculus and compare performance with the causal calculator on eight problems in the zoo world domain.

*Keywords*: Commonsense reasoning, reasoning about action and change, event calculus, satisfiability, automated deduction.

## 1 Introduction

The classical logic event calculus [37, 39, 26], which is based on the original event calculus of Kowalski and Sergot [18], is a well-developed formalism for reasoning about action and change. It has been used to represent such things as beliefs and car crashes [20], egg cracking [27, 43], robot mail delivery [42], and robot sensors [38]. It is able to cope with representational problems such as the representation of conditional effects of events, triggered events, events with nondeterministic effects, events with indirect effects, gradual change, and the commonsense law of inertia.

To date, most reasoning with the classical logic event calculus has been carried out using one of two methods: (1) manual theorem proving by humans [27, 43] or (2) automated theorem proving through logic programming [42]. Citing research demonstrating the efficiency of planning using satisfiability (SAT) [16, 17], Shanahan and Witkowski [44] proposed that event calculus planning be carried out using satisfiability and presented a method for encoding event calculus planning problems as satisfiability problems. They demonstrated the efficiency of satisfiability over abductive logic programming for solving event calculus planning problems. However, the method of Shanahan and Witkowski applies only to a highly restricted subset of the event calculus.

The goals of this paper are:

1. to describe and prove the correctness of a method for encoding event calculus reasoning problems as satisfiability problems for a larger subset of the event calculus,
2. to describe a method that enables event calculus reasoning problems to be solved efficiently,
3. to present an alternative classical logic axiomatization of the event calculus useful for satisfiability encoding and prove its equivalence to a standard axiomatization for integer time, and
4. to evaluate the method on benchmark reasoning problems.

We build on Shanahan and Witkowski's method, extending it to support causal constraints, concurrent events, determining fluents, effect axioms with conditions, events triggered by conditions, gradual change, incompletely specified initial situations, state constraints, and release from the commonsense law of inertia.

The efficiency of our method hinges on the use of two techniques: (1) reformulation of the classical logic axiomatization of the event calculus in order to eliminate triply quantified time from most axioms, and (2) elimination from the reasoning problem of a large number of ground atoms stemming from effect axioms and gradual change axioms.

This paper is organized as follows. In Section 2 we present a standard classical logic axiomatization of the event calculus that serves as a point of reference. In Section 3 we present an alternative axiomatization and prove its equivalence to the axiomatization of Section 2 for integer time. In Section 4 we present and prove the correctness of our satisfiability encoding method for model finding, deduction, and abduction. In Section 5 we evaluate the method. Finally we present conclusions.

## 2 The event calculus

The classical logic event calculus [39] is based on many-sorted predicate calculus with equality. There are sorts for events, fluents, timepoints, and domain objects. A classical logic axiomatization of the event calculus has been described in a paper by Miller and Shanahan [26]. In that paper, several alternative axiomatizations are provided that subtract or add various features of the event calculus. In this section, we fix one set of axioms to serve as a point of reference. We combine axioms from the following sections of the paper:

- Section 2, which provides the basic classical logic axiomatization of the event calculus,
- Section 3.2, which revises the axioms of Section 2 for a version of the event calculus in which initiating and terminating a fluent at the same time produces inconsistency,
- Section 3.5, which adds axioms to those of Section 2 to support gradual change, and
- Section 3.7, which revises the axioms of Section 2 for a version of the event calculus in which fluents may be released from the commonsense law of inertia.

### *2.1 Event calculus predicates*

The basic predicates of the event calculus axiomatization are as follows:

1. *Happens*$(a, t)$: Event $a$ occurs at timepoint $t$.
2. *HoldsAt*$(f, t)$: Fluent $f$ is true at timepoint $t$.
3. *ReleasedAt*$(f, t)$: Fluent $f$ is released from the commonsense law of inertia at timepoint $t$.
4. *Initiates*$(a, f, t)$: If event $a$ occurs at timepoint $t$, then fluent $f$ becomes true after $t$ and is no longer released from the commonsense law of inertia after $t$.
5. *Terminates*$(a, f, t)$: If event $a$ occurs at timepoint $t$, then fluent $f$ becomes false after $t$ and is no longer released from the commonsense law of inertia after $t$.
6. *Releases*$(a, f, t)$: If event $a$ occurs at timepoint $t$, then fluent $f$ becomes released from the commonsense law of inertia after $t$.
7. *Trajectory*$(f_1, t_1, f_2, t_2)$: If fluent $f_1$ is initiated by an event that occurs at timepoint $t_1$ and $t_2 > 0$, then fluent $f_2$ is true at timepoint $t_1 + t_2$.

8. *AntiTrajectory*$(f_1, t_1, f_2, t_2)$: If fluent $f_1$ is terminated by an event that occurs at timepoint $t_1$ and $t_2 > 0$, then fluent $f_2$ is true at timepoint $t_1 + t_2$.

## 2.2 Event calculus (EC) axiomatization

The axiomatization of the event calculus is as follows. The following definitional axioms are from Miller and Shanahan Section 2 [26]:

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

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

A definitional axiom $\Gamma_1 \overset{\text{def}}{\equiv} \Gamma_2$ indicates that $\Gamma_1$ is a notational shorthand for $\Gamma_2$. That is, all occurrences of the compact notation $\Gamma_1$ are to be replaced by the more complex formula $\Gamma_2$. In this paper, free occurrences of variables in formulas are assumed to be universally quantified.

The following definitional axioms are from Miller and Shanahan Section 3.2 [26]:

AXIOM EC9B
$\mathit{StoppedIn}(t_1, f, t_2) \overset{\text{def}}{\equiv} \exists a, t[\mathit{Happens}(a, t) \wedge t_1 < t < t_2 \wedge \mathit{Terminates}(a, f, t)]$.

AXIOM EC10B
$\mathit{StartedIn}(t_1, f, t_2) \overset{\text{def}}{\equiv} \exists a, t[\mathit{Happens}(a, t) \wedge t_1 < t < t_2 \wedge \mathit{Initiates}(a, f, t)]$.

The following axioms for gradual change are obtained from EC11F and EC12F of Miller and Shanahan Section 3.5 [26] by removing $\neg \mathit{Frame}(f_2)$, which is not needed:

AXIOM EC11F′
$[\mathit{Happens}(a, t_1) \wedge \mathit{Initiates}(a, f_1, t_1) \wedge 0 < t_2 \wedge$
$\mathit{Trajectory}(f_1, t_1, f_2, t_2) \wedge \neg \mathit{Clipped}(t_1, f_1, t_1 + t_2)] \Rightarrow$
$\mathit{HoldsAt}(f_2, t_1 + t_2)$.

AXIOM EC12F′
$[\mathit{Happens}(a, t_1) \wedge \mathit{Terminates}(a, f_1, t_1) \wedge 0 < t_2 \wedge$
$\mathit{AntiTrajectory}(f_1, t_1, f_2, t_2) \wedge \neg \mathit{Declipped}(t_1, f_1, t_1 + t_2)] \Rightarrow$
$\mathit{HoldsAt}(f_2, t_1 + t_2)$.

The following definitional axiom is obtained from EC17H of Miller and Shanahan Section 3.7 [26] by changing $t_1 \leq t \leq t_2$ to $t_1 < t \leq t_2$ in order to conform to the version of the event calculus described in Miller and Shanahan Section 3.2 [26]:

AXIOM EC17H′
$\mathit{PersistsBetween}(t_1, f, t_2) \overset{\text{def}}{\equiv} \neg \exists t[\mathit{ReleasedAt}(f, t) \wedge t_1 < t \leq t_2]$.

The following definitional axiom is from Miller and Shanahan Section 3.7 [26]:

AXIOM EC14H
$\mathit{ReleasedBetween}(t_1, f, t_2) \overset{\text{def}}{\equiv} \exists a, t[\mathit{Happens}(a, t) \wedge t_1 \leq t < t_2 \wedge \mathit{Releases}(a, f, t)]$.

The following axioms, which deal with the frame problem [24, 9], are from Miller and Shanahan Section 3.7 [26]:

AXIOM EC5H
$[\mathit{HoldsAt}(f, t_1) \wedge t_1 < t_2 \wedge \mathit{PersistsBetween}(t_1, f, t_2) \wedge \neg \mathit{Clipped}(t_1, f, t_2)] \Rightarrow$
$\mathit{HoldsAt}(f, t_2)$.

AXIOM EC6H
$[\neg \mathit{HoldsAt}(f, t_1) \wedge t_1 < t_2 \wedge \mathit{PersistsBetween}(t_1, f, t_2) \wedge \neg \mathit{Declipped}(t_1, f, t_2)] \Rightarrow$
$\neg \mathit{HoldsAt}(f, t_2)$.

AXIOM EC18H
$[\mathit{ReleasedAt}(f, t_1) \wedge t_1 < t_2 \wedge \neg \mathit{Clipped}(t_1, f, t_2) \wedge \neg \mathit{Declipped}(t_1, f, t_2)] \Rightarrow$
$\mathit{ReleasedAt}(f, t_2)$.

AXIOM EC19H
$[\neg \mathit{ReleasedAt}(f, t_1) \wedge t_1 < t_2 \wedge \neg \mathit{ReleasedBetween}(t_1, f, t_2)] \Rightarrow$
$\neg \mathit{ReleasedAt}(f, t_2)$.

Axioms EC5H and EC6H are frame axioms for *HoldsAt*, and EC18H and EC19H are frame axioms for *ReleasedAt*.

The following definitional axiom is obtained from EC14H of Miller and Shanahan Section 3.7 [26] by changing *ReleasedBetween* to *ReleasedIn*, which will be used below, and $t_1 \leq t < t_2$ to $t_1 < t < t_2$ in order to conform to the version of the event calculus described in Miller and Shanahan Section 3.2 [26]:

AXIOM EC14H′
$\mathit{ReleasedIn}(t_1, f, t_2) \overset{\text{def}}{\equiv} \exists a, t[\mathit{Happens}(a, t) \wedge t_1 < t < t_2 \wedge \mathit{Releases}(a, f, t)]$.

A fluent can be in one of four states at a given timepoint: true and released, true and not released, false and released, or false and not released. The remaining axioms describe how the occurrence of an event affects the states of fluents. They are obtained from EC3H, EC4H, EC15H, and EC16H of Miller and Shanahan Section 3.7 [26] by changing *Clipped* to *StoppedIn*, *Declipped* to *StartedIn*, and *ReleasedBetween* to *ReleasedIn*, in order to conform to the version of the event calculus described in Miller and Shanahan Section 3.2 [26]:

AXIOM EC3H′
$[\mathit{Happens}(a, t_1) \wedge \mathit{Initiates}(a, f, t_1) \wedge t_1 < t_2 \wedge \neg \mathit{StoppedIn}(t_1, f, t_2) \wedge \neg \mathit{ReleasedIn}(t_1, f, t_2)] \Rightarrow$
$\mathit{HoldsAt}(f, t_2)$.

AXIOM EC4H′
$[\mathit{Happens}(a, t_1) \wedge \mathit{Terminates}(a, f, t_1) \wedge t_1 < t_2 \wedge \neg \mathit{StartedIn}(t_1, f, t_2) \wedge \neg \mathit{ReleasedIn}(t_1, f, t_2)] \Rightarrow$

$\neg \mathit{HoldsAt}(f, t_2)$.

AXIOM EC15H′
$[\mathit{Happens}(a, t_1) \wedge \mathit{Releases}(a, f, t_1) \wedge t_1 < t_2 \wedge \neg \mathit{StoppedIn}(t_1, f, t_2) \wedge \neg \mathit{StartedIn}(t_1, f, t_2)] \Rightarrow$
$\mathit{ReleasedAt}(f, t_2)$.

AXIOM EC16H′
$[\mathit{Happens}(a, t_1) \wedge [\mathit{Initiates}(a, f, t_1) \vee \mathit{Terminates}(a, f, t_1)] \wedge t_1 < t_2 \wedge \neg \mathit{ReleasedIn}(t_1, f, t_2)] \Rightarrow$
$\neg \mathit{ReleasedAt}(f, t_2)$.

## 2.3 Domain descriptions

Let EC be the conjunction of EC11F′, EC12F′, EC5H, EC6H, EC18H, EC19H, EC3H′, EC4H′, EC15H′, and EC16H′. Define $\mathit{CIRC}[\Phi; \rho_1, \ldots, \rho_n]$ as the circumscription [23, 21] of the predicate symbols $\rho_1, \ldots, \rho_n$ in the formula $\Phi$.

DEFINITION 2.1
An event calculus *domain description* is given by

$$\mathit{CIRC}[\Sigma; \mathit{Initiates}, \mathit{Terminates}, \mathit{Releases}] \wedge \mathit{CIRC}[\Delta; \mathit{Happens}] \wedge \Omega \wedge \Psi \wedge \Pi \wedge \Gamma \wedge \mathrm{EC}$$

where

- $\Sigma$ is a conjunction of *Initiates*, *Terminates*, and *Releases* formulas,
- $\Delta$ is a conjunction of *Happens* and temporal ordering formulas,
- $\Omega$ is the uniqueness-of-names axioms for all the event symbols conjoined with the uniqueness-of-names axioms for all the fluent symbols,
- $\Psi$ is a conjunction of state constraints,
- $\Pi$ is a conjunction of *Trajectory* and *AntiTrajectory* formulas, and
- $\Gamma$ is a conjunction of *HoldsAt* and *ReleasedAt* formulas.

EXAMPLE 2.2 (Domain description)
First we axiomatize some simple knowledge about falling objects. We use a state constraint that says that an object has a unique height:

$$\mathit{HoldsAt}(\mathit{Height}(o, h_1), t) \wedge \mathit{HoldsAt}(\mathit{Height}(o, h_2), t) \Rightarrow h_1 = h_2. \tag{2.1}$$

We add an effect axiom that states that if an object starts falling, then it will be falling:

$$\mathit{Initiates}(\mathit{StartFalling}(o), \mathit{Falling}(o), t). \tag{2.2}$$

We add a second effect axiom that states that if an object starts falling, then its height will be released from the commonsense law of inertia:

$$\mathit{Releases}(\mathit{StartFalling}(o), \mathit{Height}(o, h), t). \tag{2.3}$$

Next we use a gradual change axiom that states that if an object starts falling at time $t_1$ when its height is $h_1$, then its height at time $t_2$ will be $\mathit{Max}(0, h_1 - {t_2}^2)$:

$$\begin{gathered}\mathit{HoldsAt}(\mathit{Height}(o, h_1), t_1) \wedge h_2 = \mathit{Max}(0, h_1 - {t_2}^2) \Rightarrow \\ \mathit{Trajectory}(\mathit{Falling}(o), t_1, \mathit{Height}(o, h_2), t_2).\end{gathered} \tag{2.4}$$

Next we use a trigger axiom that states that if an object is falling and its height is zero, then it will hit the ground:

$$\begin{aligned} &\mathit{HoldsAt}(\mathit{Falling}(o), t) \land \mathit{HoldsAt}(\mathit{Height}(o, 0), t) \Rightarrow \\ &\qquad \mathit{Happens}(\mathit{HitsGround}(o), t). \end{aligned} \tag{2.5}$$

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

$$\begin{aligned} &\mathit{HoldsAt}(\mathit{Height}(o, h), t) \Rightarrow \\ &\mathit{Initiates}(\mathit{HitsGround}(o), \mathit{Height}(o, h), t). \end{aligned} \tag{2.6}$$

Finally we add an effect axiom that states that if an object hits the ground, then the object will no longer be falling:

$$\mathit{Terminates}(\mathit{HitsGround}(o), \mathit{Falling}(o), t). \tag{2.7}$$

Now we add formulas describing an initial situation and an event occurrence:

$$\neg \mathit{HoldsAt}(\mathit{Falling}(\mathit{Leaf}), 0) \tag{2.8}$$

$$\neg \mathit{ReleasedAt}(\mathit{Falling}(\mathit{Leaf}), 0) \tag{2.9}$$

$$\mathit{HoldsAt}(\mathit{Height}(\mathit{Leaf}, 9), 0) \tag{2.10}$$

$$\neg \mathit{ReleasedAt}(\mathit{Height}(\mathit{Leaf}, h), 0) \tag{2.11}$$

$$\mathit{Happens}(\mathit{StartFalling}(\mathit{Leaf}), 0). \tag{2.12}$$

We may now reason using this domain description as follows. Let $\Sigma$ be the conjunction of (2.2), (2.3), (2.6), and (2.7). Let $\Delta$ be the conjunction of (2.5) and (2.12). Let $\Omega$ be the uniqueness-of-names axioms for event and fluent symbols. Let $\Psi$ be (2.1). Let $\Pi$ be (2.4). Let $\Gamma$ be the conjunction of (2.8), (2.9), (2.10), and (2.11). We can then show the following:

$$\begin{aligned} &\mathit{CIRC}[\Sigma; \mathit{Initiates}, \mathit{Terminates}, \mathit{Releases}] \land \\ &\mathit{CIRC}[\Delta; \mathit{Happens}] \land \Omega \land \Psi \land \Pi \land \Gamma \land \mathrm{EC} \models \\ &\quad \mathit{HoldsAt}(\mathit{Falling}(\mathit{Leaf}), 1) \land \\ &\quad \mathit{HoldsAt}(\mathit{Height}(\mathit{Leaf}, 8), 1) \land \\ &\quad \mathit{HoldsAt}(\mathit{Height}(\mathit{Leaf}, 5), 2) \land \\ &\quad \mathit{HoldsAt}(\mathit{Height}(\mathit{Leaf}, 0), 3) \land \\ &\quad \mathit{Happens}(\mathit{HitsGround}(\mathit{Leaf}), 3) \land \\ &\quad \neg \mathit{HoldsAt}(\mathit{Falling}(\mathit{Leaf}), 4) \land \\ &\quad \mathit{HoldsAt}(\mathit{Height}(\mathit{Leaf}, 0), 4). \end{aligned}$$

## 3 The discrete event calculus

We desire an efficient satisfiability encoding for the event calculus. Observe that the standard event calculus axiomatization of the previous section involves triple quantification over timepoints. This is not ideal since it leads to an encoding size proportional to the cube of the number of timepoints.

In this section, we show that if one restricts the timepoint sort to the integers, then it becomes possible to eliminate triply quantified timepoints from each of the event calculus axioms and replace them with only singly quantified timepoints except for the two axioms that deal with gradual change. We present an alternative classical logic axiomatization of the event calculus, which we call the *discrete event calculus*, and prove that it is logically equivalent to the standard axiomatization if the timepoint sort is restricted to the integers.

It is worth pointing out that this equivalence does not require restriction to a finite universe, as in our propositional encoding of Section 4. It merely requires restriction of timepoints to the (infinite) set of integers.

### *3.1 Discrete event calculus (DEC) axiomatization*

The axioms of the discrete event calculus are as follows. We start with two definitional axioms identical to EC1 and EC2:

AXIOM DEC1
$\mathit{Clipped}(t_1, f, t_2) \overset{\text{def}}{\equiv} \exists a, t[\mathit{Happens}(a, t) \wedge t_1 \leq t < t_2 \wedge \mathit{Terminates}(a, f, t)]$.

AXIOM DEC2
$\mathit{Declipped}(t_1, f, t_2) \overset{\text{def}}{\equiv} \exists a, t[\mathit{Happens}(a, t) \wedge t_1 \leq t < t_2 \wedge \mathit{Initiates}(a, f, t)]$.

We then have axioms for gradual change identical to EC11F$'$ and EC12F$'$:

AXIOM DEC3
$[\mathit{Happens}(a, t_1) \wedge \mathit{Initiates}(a, f_1, t_1) \wedge 0 < t_2 \wedge$
$\mathit{Trajectory}(f_1, t_1, f_2, t_2) \wedge \neg\mathit{Clipped}(t_1, f_1, t_1 + t_2)] \Rightarrow$
$\mathit{HoldsAt}(f_2, t_1 + t_2)$.

AXIOM DEC4
$[\mathit{Happens}(a, t_1) \wedge \mathit{Terminates}(a, f_1, t_1) \wedge 0 < t_2 \wedge$
$\mathit{AntiTrajectory}(f_1, t_1, f_2, t_2) \wedge \neg\mathit{Declipped}(t_1, f_1, t_1 + t_2)] \Rightarrow$
$\mathit{HoldsAt}(f_2, t_1 + t_2)$.

We have new frame axioms that resemble explanation closure frame axioms [12, 35, 3, 33, 34], extended to allow fluents to be released from the commonsense law of inertia:

AXIOM DEC5
$[\mathit{HoldsAt}(f, t) \wedge \neg\mathit{ReleasedAt}(f, t + 1) \wedge \neg\exists a[\mathit{Happens}(a, t) \wedge \mathit{Terminates}(a, f, t)]] \Rightarrow$
$\mathit{HoldsAt}(f, t + 1)$.

AXIOM DEC6
$[\neg \mathit{HoldsAt}(f,t) \wedge \neg \mathit{ReleasedAt}(f,t+1) \wedge \neg\exists a[\mathit{Happens}(a,t) \wedge \mathit{Initiates}(a,f,t)]] \Rightarrow$
$\neg \mathit{HoldsAt}(f,t+1).$

AXIOM DEC7
$[\mathit{ReleasedAt}(f,t) \wedge \neg\exists a[\mathit{Happens}(a,t) \wedge [\mathit{Initiates}(a,f,t) \vee \mathit{Terminates}(a,f,t)]]] \Rightarrow$
$\mathit{ReleasedAt}(f,t+1).$

AXIOM DEC8
$[\neg \mathit{ReleasedAt}(f,t) \wedge \neg\exists a[\mathit{Happens}(a,t) \wedge \mathit{Releases}(a,f,t)]] \Rightarrow$
$\neg \mathit{ReleasedAt}(f,t+1).$

We have axioms that describe how the occurrence of an event affects the states of fluents:

AXIOM DEC9
$[\mathit{Happens}(a,t) \wedge \mathit{Initiates}(a,f,t)] \Rightarrow \mathit{HoldsAt}(f,t+1).$

AXIOM DEC10
$[\mathit{Happens}(a,t) \wedge \mathit{Terminates}(a,f,t)] \Rightarrow \neg \mathit{HoldsAt}(f,t+1).$

AXIOM DEC11
$[\mathit{Happens}(a,t) \wedge \mathit{Releases}(a,f,t)] \Rightarrow \mathit{ReleasedAt}(f,t+1).$

AXIOM DEC12
$[\mathit{Happens}(a,t) \wedge [\mathit{Initiates}(a,f,t) \vee \mathit{Terminates}(a,f,t)]] \Rightarrow \neg \mathit{ReleasedAt}(f,t+1).$

Let DEC be the conjunction of DEC3 through DEC12.

## 3.2 Equivalence of EC and DEC

We now prove the equivalence of EC and DEC if the timepoint sort is restricted to the integers. Before doing so, a number of lemmas are required.

LEMMA 3.1
If the timepoint sort is restricted to the integers, then

DEC ⇒ EC5H.

PROOF. Suppose DEC. Let $\tau_1$ and $\tau_2$ be arbitrary integer timepoints and $\beta$ be an arbitrary fluent. We must show

$$[\mathit{HoldsAt}(\beta,\tau_1) \wedge \tau_1 < \tau_2 \wedge \mathit{PersistsBetween}(\tau_1,\beta,\tau_2) \wedge \neg \mathit{Clipped}(\tau_1,\beta,\tau_2)] \Rightarrow \mathit{HoldsAt}(\beta,\tau_2). \tag{3.1}$$

Case 1: $\tau_1 \geq \tau_2$. (3.1) is trivially satisfied.
Case 2: $\tau_1 < \tau_2$. We proceed by mathematical induction.
Base case: We show that (3.1) is true for $\tau_2 = \tau_1 + 1$. Suppose

$$\mathit{HoldsAt}(\beta,\tau_1) \wedge \mathit{PersistsBetween}(\tau_1,\beta,\tau_1+1) \wedge \neg \mathit{Clipped}(\tau_1,\beta,\tau_1+1).$$

From $\mathit{PersistsBetween}(\tau_1, \beta, \tau_1+1)$ and the definitional axiom EC17H′, we have $\neg\mathit{ReleasedAt}(\beta, \tau_1+1)$. From $\neg\mathit{Clipped}(\tau_1, \beta, \tau_1+1)$ and the definitional axiom EC1, we have $\neg\exists a[\mathit{Happens}(a, \tau_1) \wedge \mathit{Terminates}(a, \beta, \tau_1)]$. From this, $\mathit{HoldsAt}(\beta, \tau_1)$, $\neg\mathit{ReleasedAt}(\beta, \tau_1+1)$, and DEC5, we have $\mathit{HoldsAt}(\beta, \tau_1+1)$ as required.

Induction step: Suppose (3.1) is true for $\tau_2 = k$, $k > \tau_1$ (induction hypothesis):

$$[\mathit{HoldsAt}(\beta, \tau_1) \wedge \tau_1 < k \wedge \mathit{PersistsBetween}(\tau_1, \beta, k) \wedge \neg\mathit{Clipped}(\tau_1, \beta, k)] \Rightarrow \mathit{HoldsAt}(\beta, k). \tag{3.2}$$

We must show that (3.1) is true for $\tau_2 = k+1$. Suppose

$$\mathit{HoldsAt}(\beta, \tau_1) \wedge \tau_1 < k+1 \wedge \mathit{PersistsBetween}(\tau_1, \beta, k+1) \wedge \neg\mathit{Clipped}(\tau_1, \beta, k+1).$$

From $\mathit{PersistsBetween}(\tau_1, \beta, k+1)$ and EC17H′, we have $\mathit{PersistsBetween}(\tau_1, \beta, k)$. From $\neg\mathit{Clipped}(\tau_1, \beta, k+1)$ and EC1, we have $\neg\mathit{Clipped}(\tau_1, \beta, k)$. From this, $\mathit{HoldsAt}(\beta, \tau_1)$, $\tau_1 < k$, $\mathit{PersistsBetween}(\tau_1, \beta, k)$, and the induction hypothesis (3.2), we have $\mathit{HoldsAt}(\beta, k)$. From $\mathit{PersistsBetween}(\tau_1, \beta, k+1)$ and EC17H′, we have $\neg\mathit{ReleasedAt}(\beta, k+1)$. From $\neg\mathit{Clipped}(\tau_1, \beta, k+1)$ and EC1, we have $\neg\exists a[\mathit{Happens}(a, k) \wedge \mathit{Terminates}(a, \beta, k)]$. From this, $\mathit{HoldsAt}(\beta, k)$, $\neg\mathit{ReleasedAt}(\beta, k+1)$, and DEC5, we have $\mathit{HoldsAt}(\beta, k+1)$ as required. ∎

LEMMA 3.2
If the timepoint sort is restricted to the integers, then

DEC ⇒ EC6H.

PROOF. The proof is identical to that of Lemma 3.1, except that $\neg\mathit{HoldsAt}$ is substituted for *HoldsAt*, *Initiates* is substituted for *Terminates*, DEC6 is substituted for DEC5, *Declipped* is substituted for *Clipped*, EC2 is substituted for EC1, and EC6H is substituted for EC5H. ∎

LEMMA 3.3
If the timepoint sort is restricted to the integers, then

DEC ⇒ EC18H.

PROOF. Suppose DEC. Let $\tau_1$ and $\tau_2$ be arbitrary integer timepoints and $\beta$ be an arbitrary fluent. We must show

$$[\mathit{ReleasedAt}(\beta, \tau_1) \wedge \tau_1 < \tau_2 \wedge \neg\mathit{Clipped}(\tau_1, \beta, \tau_2) \wedge \neg\mathit{Declipped}(\tau_1, \beta, \tau_2)] \Rightarrow \mathit{ReleasedAt}(\beta, \tau_2). \tag{3.3}$$

Case 1: $\tau_1 \geq \tau_2$. (3.3) is trivially satisfied.
Case 2: $\tau_1 < \tau_2$. We proceed by mathematical induction.
Base case: We show that (3.3) is true for $\tau_2 = \tau_1 + 1$. Suppose

$$\mathit{ReleasedAt}(\beta, \tau_1) \wedge \tau_1 < \tau_1 + 1 \wedge \neg\mathit{Clipped}(\tau_1, \beta, \tau_1+1) \wedge \neg\mathit{Declipped}(\tau_1, \beta, \tau_1+1).$$

From $\neg\mathit{Declipped}(\tau_1, \beta, \tau_1+1)$ and EC2, we have $\neg\exists a[\mathit{Happens}(a, \tau_1) \wedge \mathit{Initiates}(a, \beta, \tau_1)]$. From $\neg\mathit{Clipped}(\tau_1, \beta, \tau_1+1)$ and EC1, we have $\neg\exists a[\mathit{Happens}(a, \tau_1) \wedge \mathit{Terminates}(a, \beta, \tau_1)]$. From $\mathit{ReleasedAt}(\beta, \tau_1)$, $\neg\exists a[\mathit{Happens}(a, \tau_1) \wedge \mathit{Initiates}(a, \beta, \tau_1)]$, $\neg\exists a[\mathit{Happens}(a, \tau_1) \wedge \mathit{Terminates}(a, \beta, \tau_1)]$, and DEC7, we have $\mathit{ReleasedAt}(\beta, \tau_1+1)$ as required.

Induction step: Suppose (3.3) is true for $\tau_2 = k$, $k > \tau_1$ (induction hypothesis):

$$[\mathit{ReleasedAt}(\beta, \tau_1) \wedge \tau_1 < k \wedge \neg \mathit{Clipped}(\tau_1, \beta, k) \wedge \neg \mathit{Declipped}(\tau_1, \beta, k)] \Rightarrow \mathit{ReleasedAt}(\beta, k). \tag{3.4}$$

We must show that (3.3) is true for $\tau_2 = k + 1$. Suppose

$$\mathit{ReleasedAt}(\beta, \tau_1) \wedge \tau_1 < k + 1 \wedge \neg \mathit{Clipped}(\tau_1, \beta, k + 1) \wedge \neg \mathit{Declipped}(\tau_1, \beta, k + 1).$$

From $\neg \mathit{Clipped}(\tau_1, \beta, k+1)$ and EC1, we have $\neg \mathit{Clipped}(\tau_1, \beta, k)$. From $\neg \mathit{Declipped}(\tau_1, \beta, k+1)$ and EC2, we have $\neg \mathit{Declipped}(\tau_1, \beta, k)$. From this, $\mathit{ReleasedAt}(\beta, \tau_1)$, $\tau_1 < k$, $\neg \mathit{Clipped}(\tau_1, \beta, k)$, and the induction hypothesis (3.4), we have $\mathit{ReleasedAt}(\beta, k)$. From $\neg \mathit{Declipped}(\tau_1, \beta, k + 1)$ and EC2, we have $\neg \exists a[\mathit{Happens}(a, k) \wedge \mathit{Initiates}(a, \beta, k)]$. From $\neg \mathit{Clipped}(\tau_1, \beta, k + 1)$ and EC1, we have $\neg \exists a[\mathit{Happens}(a, k) \wedge \mathit{Terminates}(a, \beta, k)]$. From $\mathit{ReleasedAt}(\beta, k)$, $\neg \exists a[\mathit{Happens}(a, k) \wedge \mathit{Initiates}(a, \beta, k)]$, $\neg \exists a[\mathit{Happens}(a, k) \wedge \mathit{Terminates}(a, \beta, k)]$, and DEC7, we have $\mathit{ReleasedAt}(\beta, k+1)$ as required. ∎

The following lemma is used in Theorem 3.9 as well as Lemmas 3.5 and 3.6:

LEMMA 3.4
If the timepoint sort is restricted to the integers, then

DEC ⇒ EC19H.

PROOF. Suppose DEC. Let $\tau_1$ and $\tau_2$ be arbitrary integer timepoints and $\beta$ be an arbitrary fluent. We must show

$$[\neg \mathit{ReleasedAt}(\beta, \tau_1) \wedge \tau_1 < \tau_2 \wedge \neg \mathit{ReleasedBetween}(\tau_1, \beta, \tau_2)] \Rightarrow \neg \mathit{ReleasedAt}(\beta, \tau_2). \tag{3.5}$$

Case 1: $\tau_1 \geq \tau_2$. (3.5) is trivially satisfied.
Case 2: $\tau_1 < \tau_2$. We proceed by mathematical induction.
Base case: We show that (3.5) is true for $\tau_2 = \tau_1 + 1$. Suppose

$$\neg \mathit{ReleasedAt}(\beta, \tau_1) \wedge \tau_1 < \tau_1 + 1 \wedge \neg \mathit{ReleasedBetween}(\tau_1, \beta, \tau_1 + 1).$$

From $\neg \mathit{ReleasedBetween}(\tau_1, \beta, \tau_1+1)$ and EC14H, we have $\neg \exists a[\mathit{Happens}(a, \tau_1) \wedge \mathit{Releases}(a, \beta, \tau_1)]$. From this, $\neg \mathit{ReleasedAt}(\beta, \tau_1)$, and DEC8, we have $\neg \mathit{ReleasedAt}(\beta, \tau_1 + 1)$ as required.

Induction step: Suppose (3.5) is true for $\tau_2 = k$, $k > \tau_1$ (induction hypothesis):

$$[\neg \mathit{ReleasedAt}(\beta, \tau_1) \wedge \tau_1 < k \wedge \neg \mathit{ReleasedBetween}(\tau_1, \beta, k)] \Rightarrow \neg \mathit{ReleasedAt}(\beta, k). \tag{3.6}$$

We must show that (3.5) is true for $\tau_2 = k + 1$. Suppose

$$\neg \mathit{ReleasedAt}(\beta, \tau_1) \wedge \tau_1 < k + 1 \wedge \neg \mathit{ReleasedBetween}(\tau_1, \beta, k + 1).$$

From $\neg \mathit{ReleasedBetween}(\tau_1, \beta, k + 1)$ and EC14H, we have $\neg \mathit{ReleasedBetween}(\tau_1, \beta, k)$. From this, $\neg \mathit{ReleasedAt}(\beta, \tau_1)$, $\tau_1 < k$, and the induction hypothesis (3.6), we have $\neg \mathit{ReleasedAt}(\beta, k)$. From $\neg \mathit{ReleasedBetween}(\tau_1, \beta, k+1)$ and EC14H, we have $\neg \exists a[\mathit{Happens}(a, k) \wedge \mathit{Releases}(a, \beta, k)]$. From this, $\neg \mathit{ReleasedAt}(\beta, k)$, and DEC8, we have $\neg \mathit{ReleasedAt}(\beta, k + 1)$ as required. ∎

LEMMA 3.5
If the timepoint sort is restricted to the integers, then

$$\text{DEC} \Rightarrow \text{EC3H}'.$$

PROOF. Suppose DEC. Let $\tau_1$ and $\tau_2$ be arbitrary integer timepoints, $\alpha$ be an arbitrary event, and $\beta$ be an arbitrary fluent. We must show

$$\begin{aligned}[&\mathit{Happens}(\alpha,\tau_1) \wedge \mathit{Initiates}(\alpha,\beta,\tau_1) \wedge \tau_1 < \tau_2 \wedge \\ &\neg\mathit{StoppedIn}(\tau_1,\beta,\tau_2) \wedge \neg\mathit{ReleasedIn}(\tau_1,\beta,\tau_2)] \Rightarrow \\ &\qquad \mathit{HoldsAt}(\beta,\tau_2).\end{aligned} \tag{3.7}$$

Case 1: $\tau_1 \geq \tau_2$. (3.7) is trivially satisfied.
Case 2: $\tau_1 < \tau_2$. We proceed by mathematical induction.
Base case: We show that (3.7) is true for $\tau_2 = \tau_1 + 1$. Suppose

$$\begin{aligned}&\mathit{Happens}(\alpha,\tau_1) \wedge \mathit{Initiates}(\alpha,\beta,\tau_1) \wedge \tau_1 < \tau_1 + 1 \wedge \\ &\neg\mathit{StoppedIn}(\tau_1,\beta,\tau_1+1) \wedge \neg\mathit{ReleasedIn}(\tau_1,\beta,\tau_1+1).\end{aligned}$$

From this and DEC9, we have $\mathit{HoldsAt}(\beta,\tau_1+1)$, as required.

Induction step: Suppose (3.7) is true for $\tau_2 = k$, $k > \tau_1$ (induction hypothesis):

$$\begin{aligned}[&\mathit{Happens}(\alpha,\tau_1) \wedge \mathit{Initiates}(\alpha,\beta,\tau_1) \wedge \tau_1 < k \wedge \\ &\neg\mathit{StoppedIn}(\tau_1,\beta,k) \wedge \neg\mathit{ReleasedIn}(\tau_1,\beta,k)] \Rightarrow \\ &\qquad \mathit{HoldsAt}(\beta,k).\end{aligned} \tag{3.8}$$

We must show that (3.7) is true for $\tau_2 = k + 1$. Suppose

$$\begin{aligned}&\mathit{Happens}(\alpha,\tau_1) \wedge \mathit{Initiates}(\alpha,\beta,\tau_1) \wedge \tau_1 < k + 1 \wedge \\ &\neg\mathit{StoppedIn}(\tau_1,\beta,k+1) \wedge \neg\mathit{ReleasedIn}(\tau_1,\beta,k+1).\end{aligned}$$

From $\mathit{Happens}(\alpha,\tau_1)$, $\mathit{Initiates}(\alpha,\beta,\tau_1)$, and DEC9, we have $\mathit{HoldsAt}(\beta,\tau_1+1)$. From $\neg\mathit{StoppedIn}(\tau_1,\beta,k+1)$ and EC9B, we have $\neg\mathit{StoppedIn}(\tau_1,\beta,k)$. From $\neg\mathit{ReleasedIn}(\tau_1,\beta,k+1)$ and EC14H′, we have $\neg\mathit{ReleasedIn}(\tau_1,\beta,k)$. From $\mathit{Happens}(\alpha,\tau_1)$, $\mathit{Initiates}(\alpha,\beta,\tau_1)$, $\tau_1 < k$, $\neg\mathit{StoppedIn}(\tau_1,\beta,k)$, $\neg\mathit{ReleasedIn}(\tau_1,\beta,k)$, and the induction hypothesis (3.8), we have $\mathit{HoldsAt}(\beta,k)$. From $\mathit{Happens}(\alpha,\tau_1)$, $\mathit{Initiates}(\alpha,\beta,\tau_1)$, and DEC12, we have $\neg\mathit{ReleasedAt}(\beta,\tau_1+1)$. From $\neg\mathit{ReleasedIn}(\tau_1,\beta,k+1)$, EC14H′, and EC14H, we have $\neg\mathit{ReleasedBetween}(\tau_1+1,\beta,k+1)$. From Lemma 3.4 and DEC, we have EC19H. From $\neg\mathit{ReleasedAt}(\beta,\tau_1+1)$, $\tau_1+1 < k+1$, $\neg\mathit{ReleasedBetween}(\tau_1+1,\beta,k+1)$, and EC19H, we have $\neg\mathit{ReleasedAt}(\beta,k+1)$. From $\neg\mathit{StoppedIn}(\tau_1,\beta,k+1)$ and EC9B, we have $\neg\exists a[\mathit{Happens}(a,k) \wedge \mathit{Terminates}(a,\beta,k)]$. From this, $\mathit{HoldsAt}(\beta,k)$, $\neg\mathit{ReleasedAt}(\beta,k+1)$, and DEC5, we have $\mathit{HoldsAt}(\beta,k+1)$ as required. ∎

LEMMA 3.6
If the timepoint sort is restricted to the integers, then

$$\text{DEC} \Rightarrow \text{EC4H}'.$$

PROOF. The proof is identical to that of Lemma 3.5, except that $\neg$*HoldsAt* is substituted for *HoldsAt*, *Terminates* is substituted for *Initiates*, DEC10 is substituted for DEC9, *StartedIn* is substituted for *StoppedIn*, EC10B is substituted for EC9B, and DEC6 is substituted for DEC5. ∎

LEMMA 3.7
If the timepoint sort is restricted to the integers, then

$$\text{DEC} \Rightarrow \text{EC15H}'.$$

PROOF. Suppose DEC. Let $\tau_1$ and $\tau_2$ be arbitrary integer timepoints, $\alpha$ be an arbitrary event, and $\beta$ be an arbitrary fluent. We must show

$$\begin{aligned}[&\mathit{Happens}(\alpha, \tau_1) \wedge \mathit{Releases}(\alpha, \beta, \tau_1) \wedge \tau_1 < \tau_2 \wedge \\ &\neg \mathit{StoppedIn}(\tau_1, \beta, \tau_2) \wedge \neg \mathit{StartedIn}(\tau_1, \beta, \tau_2)] \Rightarrow \\ &\qquad \mathit{ReleasedAt}(\beta, \tau_2).\end{aligned} \tag{3.9}$$

Case 1: $\tau_1 \geq \tau_2$. (3.9) is trivially satisfied.
Case 2: $\tau_1 < \tau_2$. We proceed by mathematical induction.
Base case: We show that (3.9) is true for $\tau_2 = \tau_1 + 1$. Suppose

$$\begin{aligned}&\mathit{Happens}(\alpha, \tau_1) \wedge \mathit{Releases}(\alpha, \beta, \tau_1) \wedge \tau_1 < \tau_1 + 1 \wedge \\ &\neg \mathit{StoppedIn}(\tau_1, \beta, \tau_1 + 1) \wedge \neg \mathit{StartedIn}(\tau_1, \beta, \tau_1 + 1).\end{aligned}$$

From $\mathit{Happens}(\alpha, \tau_1)$, $\mathit{Releases}(\alpha, \beta, \tau_1)$, and DEC11, we have $\mathit{ReleasedAt}(\beta, \tau_1 + 1)$, as required.

Induction step: Suppose (3.9) is true for $\tau_2 = k$, $k > \tau_1$ (induction hypothesis):

$$\begin{aligned}[&\mathit{Happens}(\alpha, \tau_1) \wedge \mathit{Releases}(\alpha, \beta, \tau_1) \wedge \tau_1 < k \wedge \\ &\neg \mathit{StoppedIn}(\tau_1, \beta, k) \wedge \neg \mathit{StartedIn}(\tau_1, \beta, k)] \Rightarrow \\ &\qquad \mathit{ReleasedAt}(\beta, k).\end{aligned} \tag{3.10}$$

We must show that (3.9) is true for $\tau_2 = k + 1$. Suppose

$$\begin{aligned}&\mathit{Happens}(\alpha, \tau_1) \wedge \mathit{Releases}(\alpha, \beta, \tau_1) \wedge \tau_1 < k + 1 \wedge \\ &\neg \mathit{StoppedIn}(\tau_1, \beta, k + 1) \wedge \neg \mathit{StartedIn}(\tau_1, \beta, k + 1).\end{aligned}$$

From $\mathit{Happens}(\alpha, \tau_1)$, $\mathit{Releases}(\alpha, \beta, \tau_1)$, and DEC11, we have $\mathit{ReleasedAt}(\beta, \tau_1 + 1)$. From $\neg \mathit{StoppedIn}(\tau_1, \beta, k + 1)$ and EC9B, we have $\neg \mathit{StoppedIn}(\tau_1, \beta, k)$. From $\neg \mathit{StartedIn}(\tau_1, \beta, k + 1)$ and EC10B, we have $\neg \mathit{StartedIn}(\tau_1, \beta, k)$. From $\mathit{Happens}(\alpha, \tau_1)$, $\mathit{Releases}(\alpha, \beta, \tau_1)$, $\tau_1 < k$, $\neg \mathit{StoppedIn}(\tau_1, \beta, k)$, $\neg \mathit{StartedIn}(\tau_1, \beta, k)$, and the induction hypothesis (3.10), we have $\mathit{ReleasedAt}(\beta, k)$. From $\neg \mathit{StoppedIn}(\tau_1, \beta, k + 1)$ and EC9B, we have $\neg \exists a[\mathit{Happens}(a, k) \wedge \mathit{Terminates}(a, \beta, k)]$. From $\neg \mathit{StartedIn}(\tau_1, \beta, k + 1)$ and EC10B, we have $\neg \exists a[\mathit{Happens}(a, k) \wedge \mathit{Initiates}(a, \beta, k)]$. From this, $\mathit{ReleasedAt}(\beta, k)$, $\neg \exists a[\mathit{Happens}(a, k) \wedge \mathit{Terminates}(a, \beta, k)]$, and DEC7, we have $\mathit{ReleasedAt}(\beta, k + 1)$ as required. ∎

LEMMA 3.8
If the timepoint sort is restricted to the integers, then

$$\text{DEC} \Rightarrow \text{EC16H}'.$$

PROOF. Suppose DEC. Let $\tau_1$ and $\tau_2$ be arbitrary integer timepoints, $\alpha$ be an arbitrary event, and $\beta$ be an arbitrary fluent. We must show

$$\begin{aligned}[&\mathit{Happens}(\alpha, \tau_1) \wedge [\mathit{Initiates}(\alpha, \beta, \tau_1) \vee \mathit{Terminates}(\alpha, \beta, \tau_1)] \wedge \\ &\qquad \tau_1 < \tau_2 \wedge \neg \mathit{ReleasedIn}(\tau_1, \beta, \tau_2)] \Rightarrow \\ &\qquad\qquad \neg \mathit{ReleasedAt}(\beta, \tau_2).\end{aligned} \tag{3.11}$$

Case 1: $\tau_1 \geq \tau_2$. (3.11) is trivially satisfied.
Case 2: $\tau_1 < \tau_2$. We proceed by mathematical induction.
Base case: We show that (3.11) is true for $\tau_2 = \tau_1 + 1$. Suppose

$$\begin{aligned} &\mathit{Happens}(\alpha, \tau_1) \wedge [\mathit{Initiates}(\alpha, \beta, \tau_1) \vee \mathit{Terminates}(\alpha, \beta, \tau_1)] \wedge \\ &\qquad \tau_1 < \tau_1 + 1 \wedge \neg \mathit{ReleasedIn}(\tau_1, \beta, \tau_1 + 1). \end{aligned}$$

From $\mathit{Happens}(\alpha, \tau_1)$, $[\mathit{Initiates}(\alpha, \beta, \tau_1) \vee \mathit{Terminates}(\alpha, \beta, \tau_1)]$, and DEC12, we have $\neg \mathit{ReleasedAt}(\beta, \tau_1 + 1)$, as required.

Induction step: Suppose (3.11) is true for $\tau_2 = k$, $k > \tau_1$ (induction hypothesis):

$$\begin{aligned} &[\mathit{Happens}(\alpha, \tau_1) \wedge [\mathit{Initiates}(\alpha, \beta, \tau_1) \vee \mathit{Terminates}(\alpha, \beta, \tau_1)] \wedge \\ &\qquad \tau_1 < k \wedge \neg \mathit{ReleasedIn}(\tau_1, \beta, k)] \Rightarrow \\ &\qquad\qquad \neg \mathit{ReleasedAt}(\beta, k). \end{aligned} \tag{3.12}$$

We must show that (3.11) is true for $\tau_2 = k + 1$. Suppose

$$\begin{aligned} &\mathit{Happens}(\alpha, \tau_1) \wedge [\mathit{Initiates}(\alpha, \beta, \tau_1) \vee \mathit{Terminates}(\alpha, \beta, \tau_1)] \wedge \\ &\qquad \tau_1 < k + 1 \wedge \neg \mathit{ReleasedIn}(\tau_1, \beta, k + 1). \end{aligned}$$

From $\mathit{Happens}(\alpha, \tau_1)$, $\mathit{Initiates}(\alpha, \beta, \tau_1) \vee \mathit{Terminates}(\alpha, \beta, \tau_1)$, and DEC12, we have $\neg \mathit{ReleasedAt}(\beta, \tau_1 + 1)$. From $\neg \mathit{ReleasedIn}(\tau_1, \beta, k+1)$ and EC14H′, we have $\neg \mathit{ReleasedIn}(\tau_1, \beta, k)$. From this, $\mathit{Happens}(\alpha, \tau_1)$, $\mathit{Initiates}(\alpha, \beta, \tau_1) \vee \mathit{Terminates}(\alpha, \beta, \tau_1)$, $\tau_1 < k$, and the induction hypothesis (3.12), we have $\neg \mathit{ReleasedAt}(\beta, k)$. From $\neg \mathit{ReleasedIn}(\tau_1, \beta, k + 1)$ and EC14H′, we have $\neg \exists a [\mathit{Happens}(a, k) \wedge \mathit{Releases}(a, \beta, k)]$. From this, $\neg \mathit{ReleasedAt}(\beta, k)$, and DEC8, we have $\neg \mathit{ReleasedAt}(\beta, k + 1)$ as required. ∎

Now we proceed to the equivalence theorem:

THEOREM 3.9
If the timepoint sort is restricted to the integers, then

$\mathit{EC} \Leftrightarrow \mathit{DEC}$.

PROOF. We prove the two directions separately.
($\mathit{EC} \Rightarrow \mathit{DEC}$)
Suppose EC. DEC3 is identical to EC11F′ and DEC4 is identical to EC12F′. DEC5 follows from EC5H by universal instantiation, substituting $t_1 + 1$ for $t_2$. Similarly, DEC6, DEC7, DEC8, DEC9, DEC10, DEC11, and DEC12 follow from EC6H, EC18H, EC19H, EC3H′, EC4H′, EC15H′, and EC16H′, respectively, by universal instantiation, substituting $t_1 + 1$ for $t_2$.

($\mathit{EC} \Leftarrow \mathit{DEC}$)
Suppose DEC. EC11F′ is identical to DEC3 and EC12F′ is identical to DEC4. EC5H, EC6H, EC18H, EC19H, EC3H′, EC4H′, EC15H′, and EC16H′ follow from Lemmas 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, and 3.8, respectively. ∎

## 4 The encoding method

In order to perform event calculus reasoning through satisfiability, we must construct an efficient satisfiability encoding of a domain description

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

In this section, we describe our method for constructing such an encoding and prove a form of equivalence between a domain description and its encoding. Our basic method performs model finding. After describing the method in the context of model finding, we discuss how the method is also used to solve deduction and abduction problems.

## *4.1 Restriction to a finite universe*

In order to use a satisfiability solver [8], we must transform event calculus problems into the propositional calculus. A satisfiability solver takes as input a set of Boolean variables and a propositional formula over those variables and produces as output zero or more models or satisfying truth assignments—truth assignments for the variables such that the formula is true. Satisfiability solvers take a propositional formula in conjunctive normal form: a conjunction of clauses, where each clause is a disjunction of literals, where each literal is a variable or a negated variable. A complete satisfiability solver produces all satisfying truth assignments.

Following Shanahan and Witkowski [44], we restrict the predicate calculus to a finite universe [16, 13]. We restrict the event calculus to finite sets of variables, constants, function symbols, predicate symbols, sorts, events, fluents, timepoints, and domain objects. We restrict the timepoint sort to a finite set of integers $\{0, 1, 2, \ldots, n\}$ for some $n \geq 0$.

Note that we may now ignore $\Omega$ since the propositional calculus already incorporates the unique names assumption.

## *4.2 Definitions*

We start with some definitions.

DEFINITION 4.1
A *comparison* is a formula of the form $t_1 < t_2$, $t_1 \leq t_2$, $t_1 = t_2$, $t_1 \geq t_2$, $t_1 > t_2$, or $t_1 \neq t_2$, where $t_1$ and $t_2$ are terms.

DEFINITION 4.2
If $t$ is a variable, then a *condition over* $t$ is defined as follows: (1) A comparison is a condition over $t$. (2) If $f$ is a term, then *HoldsAt*$(f, t)$ and $\neg$*HoldsAt*$(f, t)$ are conditions over $t$. (3) If $c_1$ and $c_2$ are conditions over $t$, then $c_1 \wedge c_2$ and $c_1 \vee c_2$ are conditions over $t$. (4) If $v$ is a variable and $c$ is a condition over $t$, then $\exists v\ c$ is a condition over $t$.

DEFINITION 4.3
If $\pi$ is the predicate symbol *Initiates*, *Terminates*, or *Releases*, then a $\pi$ *effect axiom* is a formula of the form $\forall a, f, t[\Theta(a, f, t) \Rightarrow \pi(a, f, t)]$, where $\Theta(a, f, t)$ is a condition over $t$ with only $a$, $f$, and $t$ free.

DEFINITION 4.4
A $\pi$ *effect description* is a collection of $\pi$ effect axioms written as a single, logically equivalent $\pi$ effect axiom of the form $\forall a, f, t[\Theta(a, f, t) \Rightarrow \pi(a, f, t)]$, where $\Theta(a, f, t)$ is a condition over $t$ with only $a$, $f$, and $t$ free.

Let $\Sigma_{init}$ be the *Initiates* effect description $\forall a, f, t[\Theta_{init}(a, f, t) \Rightarrow$ *Initiates*$(a, f, t)]$.
Let $\Sigma_{term}$ be the *Terminates* effect description $\forall a, f, t[\Theta_{term}(a, f, t) \Rightarrow$ *Terminates*$(a, f, t)]$.
Let $\Sigma_{rel}$ be the *Releases* effect description $\forall a, f, t[\Theta_{rel}(a, f, t) \Rightarrow$ *Releases*$(a, f, t)]$.

DEFINITION 4.5
A *trigger axiom* is a formula of the form $\forall a, t[\Upsilon(a, t) \Rightarrow$ *Happens*$(a, t)]$, where $\Upsilon(a, t)$ is a condition over $t$ with only $a$ and $t$ free.

DEFINITION 4.6
A *trigger description* is a collection of trigger axioms written as a single, logically equivalent trigger axiom of the form $\forall a, t[\Upsilon(a,t) \Rightarrow \mathit{Happens}(a,t)]$, where $\Upsilon(a,t)$ is a condition over $t$ with only $a$ and $t$ free.

DEFINITION 4.7
An *event occurrence* is a formula of the form $\mathit{Happens}(a,t)$, where $a$ is an event ground term and $t$ is a timepoint ground term.

DEFINITION 4.8
An *event occurrence description* is a collection of event occurrences written as a single, logically equivalent trigger axiom of the form $\forall a, t[\Upsilon(a,t) \Rightarrow \mathit{Happens}(a,t)]$, where $\Upsilon(a,t)$ is a condition over $t$ with only $a$ and $t$ free.

DEFINITION 4.9
An *event description* is a trigger description and an event occurrence description written as a single, logically equivalent trigger axiom of the form $\forall a, t[\Upsilon(a,t) \Rightarrow \mathit{Happens}(a,t)]$, where $\Upsilon(a,t)$ is a condition over $t$ with only $a$ and $t$ free.

Let $\Delta$ be an event description.

DEFINITION 4.10
A *state constraint* is a formula of the form (1) $c_1 \Rightarrow c_2$ or (2) $c_1 \Leftrightarrow c_2$, where $c_1$ and $c_2$ are conditions over some variable $t$.

Let $\Psi$ be a conjunction of state constraints.

DEFINITION 4.11
If $\pi$ is the predicate symbol *Trajectory* or *AntiTrajectory*, then a $\pi$ *gradual change axiom* is a formula of the form $\forall f_1, t_1, f_2, t_2[\Xi(f_1, t_1, f_2, t_2) \Rightarrow \pi(f_1, t_1, f_2, t_2)]$, where $\Xi(f_1, t_1, f_2, t_2)$ is a condition over $t_1$ with only $f_1$, $t_1$, $f_2$, and $t_2$ free.

DEFINITION 4.12
A $\pi$ *gradual change description* is a collection of $\pi$ gradual change axioms written as a single, logically equivalent trajectory axiom of the form $\forall f_1, t_1, f_2, t_2[\Xi(f_1, t_1, f_2, t_2) \Rightarrow \pi(f_1, t_1, f_2, t_2)]$, where $\Xi(f_1, t_1, f_2, t_2)$ is a condition over $t_1$ with only $f_1$, $t_1$, $f_2$, and $t_2$ free.

Let $\Pi_{traj}$ be the *Trajectory* gradual change description $\forall f_1, t_1, f_2, t_2[\Xi_{traj}(f_1, t_1, f_2, t_2) \Rightarrow \mathit{Trajectory}(f_1, t_1, f_2, t_2)]$.

Let $\Pi_{anti}$ be the *AntiTrajectory* gradual change description $\forall f_1, t_1, f_2, t_2[\Xi_{anti}(f_1, t_1, f_2, t_2) \Rightarrow \mathit{AntiTrajectory}(f_1, t_1, f_2, t_2)]$.

DEFINITION 4.13
A *state description* is a conjunction of formulas of the form $\mathit{HoldsAt}(f,t)$, $\neg\mathit{HoldsAt}(f,t)$, $\mathit{ReleasedAt}(f,t)$, or $\neg\mathit{ReleasedAt}(f,t)$, where $f$ is a fluent ground term and $t$ is a timepoint ground term.

Let $\Gamma$ be a state description.

## 4.3 Computing circumscription

Our encoding method requires computation of circumscription of effect and event descriptions. We perform these computations using two theorems of Lifschitz [21]. The first theorem provides a rule for computing circumscription using predicate completion:

THEOREM 4.14
Let $\rho$ be an $n$-ary predicate symbol and $\Gamma(x_1,\ldots,x_n)$ be a formula with only $x_1,\ldots,x_n$ free. If $\Gamma(x_1,\ldots,x_n)$ does not mention $\rho$, then the circumscription *CIRC*$[\forall x_1,\ldots,x_n[\Gamma(x_1,\ldots,x_n) \Rightarrow \rho(x_1,\ldots,x_n)];\rho]$ is equivalent to $\forall x_1,\ldots,x_n[\Gamma(x_1,\ldots,x_n) \Leftrightarrow \rho(x_1,\ldots,x_n)]$.

PROOF. See the proof of Proposition 2 by Lifschitz [21]. ∎

The second theorem provides a rule for computing circumscription of several predicates.

DEFINITION 4.15
A formula $\Gamma$ is *positive relative to a predicate symbol* $\rho$ if all mentions of $\rho$ in $\Gamma$ are in the range of an even number of negations in an equivalent formula obtained by eliminating $\Rightarrow$ and $\Leftrightarrow$ from $\Gamma$.

THEOREM 4.16
Let $\rho_1,\ldots,\rho_n$ be predicate symbols and $\Gamma$ be a formula. If $\Gamma$ is positive relative to every $\rho_i$, then *CIRC*$[\Gamma;\rho_1,\ldots,\rho_n]$ is equivalent to $\bigwedge_{i=1}^{n}$ *CIRC*$[\Gamma;\rho_i]$.

PROOF. See the proof of Proposition 14 by Lifschitz [21]. ∎

## 4.4 Description of the encoding method

We now describe our method for encoding a problem given by $\Sigma_{init}$, $\Sigma_{term}$, $\Sigma_{rel}$, $\Delta$, $\Psi$, $\Pi_{traj}$, $\Pi_{anti}$, and $\Gamma$.

First, we use the axiomatization DEC of Section 3 instead of the axiomatization EC of Section 2 in order to reduce triply quantified time to singly quantified time in most axioms.

Second, observe that EC and DEC contain atoms involving *Initiates*, *Terminates*, *Releases*, *Trajectory* and *AntiTrajectory*, which may lead to a large number of ground atoms. For example, *Initiates*$(a,f,t)$ gives rise to $A \cdot F \cdot T$ ground atoms, where $A$ is the number of events, $F$ is the number of fluents, and $T$ is the number of timepoints. Therefore, in order to eliminate such atoms, we expand DEC by performing the following substitutions:

$$\mathit{Initiates}(a,f,t) \Longrightarrow \Theta_{init}(a,f,t)$$

$$\mathit{Terminates}(a,f,t) \Longrightarrow \Theta_{term}(a,f,t)$$

$$\mathit{Releases}(a,f,t) \Longrightarrow \Theta_{rel}(a,f,t)$$

$$\mathit{Trajectory}(f_1,t_1,f_2,t_2) \Longrightarrow \Xi_{traj}(f_1,t_1,f_2,t_2)$$

$$\mathit{AntiTrajectory}(f_1,t_1,f_2,t_2) \Longrightarrow \Xi_{anti}(f_1,t_1,f_2,t_2).$$

For example, if $\Sigma_{init}$ is

$$[a = \mathit{Hold}(p,o) \wedge f = \mathit{Holding}(p,o)] \Rightarrow \mathit{Initiates}(a,f,t)$$

then we replace DEC9 with

$$[\mathit{Happens}(a,t) \wedge [a = \mathit{Hold}(p,o) \wedge f = \mathit{Holding}(p,o)]] \Rightarrow \mathit{HoldsAt}(f,t+1).$$

Third, we compute *CIRC*$[\Delta;$ *Happens*$]$ using Theorems 4.14 and 4.16.

Fourth, we conjoin $\Psi$, $\Gamma$, the expanded DEC, and *CIRC*$[\Delta;$ *Happens*$]$.

Fifth, we instantiate quantifiers by replacing $\forall x\ \Phi(x)$ with $\bigwedge_i \Phi(x_i)$ and $\exists x\ \Phi(x)$ with $\bigvee_i \Phi(x_i)$, where $x_i$ are the constants of the sort of $x$. This gives a propositional calculus formula.

Sixth, we simplify the formula using standard techniques [31, pp. 35–36].

Seventh, we convert the formula to conjunctive normal form using standard techniques [6, pp. 17–18].

Finally, we construct a one-to-one and onto map $B$ that maps the ground atoms of the formula to Boolean variables. We construct an inverse map $B^{-1}$ from $B$. We construct a formula to pass to the satisfiability solver by replacing each ground atom $u$ in the formula with $B(u)$.

In order to perform model finding, we feed the formula to a satisfiability solver. We decode satisfying truth assignments produced by the solver by applying $B^{-1}$. Model finding is useful in many applications such as determining what can possibly occur given a formalization [1, 10].

Our implementation employs two additional optimizations in order to reduce the size of the encoding further. First, note that converting to conjunctive normal form using standard techniques may result in a combinatorial explosion. We convert to a compact conjunctive normal form using the technique of renaming subformulas [32, 11]. Second, instead of using a single sort for all domain objects, we allow the use of a number of domain-specific sorts.

In order to reduce the encoding time further, our implementation simplifies the expanded DEC in order to eliminate the quantification over events and fluents. For example, we simplify

$$[\mathit{Happens}(a,t) \wedge [a = \mathit{Hold}(p,o) \wedge f = \mathit{Holding}(p,o)]] \Rightarrow \mathit{HoldsAt}(f, t+1)$$

to

$$\mathit{Happens}(\mathit{Hold}(p,o),t) \Rightarrow \mathit{HoldsAt}(\mathit{Holding}(p,o), t+1).$$

EXAMPLE 4.17 (Encoding of a domain description)
Consider the following domain description. We have an *Initiates* effect description that states that if a person holds an object, then the person will be holding the object:

$$[a = \mathit{Hold}(p,o) \wedge f = \mathit{Holding}(p,o)] \Rightarrow \mathit{Initiates}(a,f,t). \tag{4.1}$$

We have a state description that says that at timepoint $0$, person *P1* is not holding object *O1* and $\mathit{Holding}(\mathit{P1},\mathit{O1})$ is not released from the commonsense law of inertia:

$$\neg\mathit{HoldsAt}(\mathit{Holding}(\mathit{P1},\mathit{O1}),0) \wedge \neg\mathit{ReleasedAt}(\mathit{Holding}(\mathit{P1},\mathit{O1}),0). \tag{4.2}$$

We have an event description that states that at timepoint $0$, person *P1* holds object *O1*:

$$[a = \mathit{Hold}(\mathit{P1},\mathit{O1}) \wedge t = 0] \Rightarrow \mathit{Happens}(a,t). \tag{4.3}$$

Suppose that $0$ and $1$ are the only constants of the timepoint sort, $P1$ is the only constant of the person sort, and $O1$ is the only constant of the object sort. The conjunctive normal form encoding of this domain description then consists of 10 clauses. We have the clauses for $(4.2)$:

$$\neg\mathit{HoldsAt}(\mathit{Holding}(\mathit{P1},\mathit{O1}),0)$$
$$\neg\mathit{ReleasedAt}(\mathit{Holding}(\mathit{P1},\mathit{O1}),0).$$

We have the following clauses, which result from the expansion of DEC5, DEC6, DEC7, DEC8, DEC9, and DEC12, respectively, given (4.1):

$$\mathit{ReleasedAt}(\mathit{Holding}(\mathit{P1},\mathit{O1}),1) \vee \mathit{HoldsAt}(\mathit{Holding}(\mathit{P1},\mathit{O1}),1) \vee \\ \neg\mathit{HoldsAt}(\mathit{Holding}(\mathit{P1},\mathit{O1}),0)$$

$$\begin{aligned}&\mathit{Happens}(\mathit{Hold}(P1,O1),0) \lor \mathit{HoldsAt}(\mathit{Holding}(P1,O1),0) \lor\\ &\mathit{ReleasedAt}(\mathit{Holding}(P1,O1),1) \lor \neg\mathit{HoldsAt}(\mathit{Holding}(P1,O1),1)\end{aligned}$$

$$\begin{aligned}&\mathit{Happens}(\mathit{Hold}(P1,O1),0) \lor \mathit{ReleasedAt}(\mathit{Holding}(P1,O1),1) \lor\\ &\neg\mathit{ReleasedAt}(\mathit{Holding}(P1,O1),0)\end{aligned}$$

$$\mathit{ReleasedAt}(\mathit{Holding}(P1,O1),0) \lor \neg\mathit{ReleasedAt}(\mathit{Holding}(P1,O1),1)$$

$$\mathit{HoldsAt}(\mathit{Holding}(P1,O1),1) \lor \neg\mathit{Happens}(\mathit{Hold}(P1,O1),0)$$

$$\neg\mathit{Happens}(\mathit{Hold}(P1,O1),0) \lor \neg\mathit{ReleasedAt}(\mathit{Holding}(P1,O1),1).$$

Note that axioms DEC10 and DEC11 are trivially satisfied since no *Terminates* or *Releases* formulas are in the domain description. We have the following clauses, which are equivalent to the circumscription of *Happens* in (4.3):

$$\begin{aligned}&\neg\mathit{Happens}(\mathit{Hold}(P1,O1),1)\\ &\mathit{Happens}(\mathit{Hold}(P1,O1),0).\end{aligned}$$

We construct a map from ground atoms to Boolean variables:

| | |
|---|---|
| *Happens*(*Hold*(*P1*,*O1*),0) | → 1 |
| *HoldsAt*(*Holding*(*P1*,*O1*),0) | → 2 |
| *ReleasedAt*(*Holding*(*P1*,*O1*),0) | → 3 |
| *Happens*(*Hold*(*P1*,*O1*),1) | → 4 |
| *ReleasedAt*(*Holding*(*P1*,*O1*),1) | → 5 |
| *HoldsAt*(*Holding*(*P1*,*O1*),1) | → 6. |

We convert the clauses into the standard DIMACS format for satisfiability problems [5]:

```
p cnf 6 10
-2 0
-3 0
5 6 -2 0
1 2 5 -6 0
1 5 -3 0
3 -5 0
6 -1 0
-1 -5 0
-4 0
1 0
```

The first line specifies the number of variables and clauses. The remaining lines are the clauses. Each line consists of a sequence of numbers. A negated variable $v$ is represented by $-v$; a non-negated variable $v$ is represented by $v$. Each line is terminated with the number $0$.

We invoke a satisfiability solver on the problem, which produces one model as output:

```
1 -2 -3 -4 -5 6
```

By applying the inverse of the above map, we get:

$$\begin{aligned}
&\mathit{Happens}(\mathit{Hold}(\mathit{P1}, \mathit{O1}), 0)\\
&\neg \mathit{HoldsAt}(\mathit{Holding}(\mathit{P1}, \mathit{O1}), 0)\\
&\neg \mathit{ReleasedAt}(\mathit{Holding}(\mathit{P1}, \mathit{O1}), 0)\\
&\neg \mathit{Happens}(\mathit{Hold}(\mathit{P1}, \mathit{O1}), 1)\\
&\neg \mathit{ReleasedAt}(\mathit{Holding}(\mathit{P1}, \mathit{O1}), 1)\\
&\mathit{HoldsAt}(\mathit{Holding}(\mathit{P1}, \mathit{O1}), 1).
\end{aligned}$$

## *4.5 Equivalence*

We now prove a form of equivalence between a domain description and the encoding of the domain description produced by our method. We start with some definitions and a lemma.

DEFINITION 4.18
An *intermediate atom* is an atom of the form *Initiates*$(\alpha, \beta, \tau)$, *Terminates*$(\alpha, \beta, \tau)$, *Releases*$(\alpha, \beta, \tau)$, *Trajectory*$(\beta_1, \tau_1, \beta_2, \tau_2)$, or *AntiTrajectory*$(\beta_1, \tau_1, \beta_2, \tau_2)$, where $\alpha$ is an event, $\beta$, $\beta_1$, and $\beta_2$ are fluents, and $\tau$, $\tau_1$, and $\tau_2$ are timepoints.

DEFINITION 4.19
Let $\Theta_{init}$, $\Theta_{term}$, $\Theta_{rel}$, $\Xi_{traj}$, and $\Xi_{anti}$ be as defined above. If $\Lambda$ is a formula, then *Expand*$[\Lambda; \Theta_{init}; \Theta_{term}; \Theta_{rel}; \Xi_{traj}; \Xi_{anti}]$ is defined as follows:

1. $\Theta_{init}(\alpha, \beta, \tau)$, if $\Lambda$ is of the form *Initiates*$(\alpha, \beta, \tau)$,
2. $\Theta_{term}(\alpha, \beta, \tau)$, if $\Lambda$ is of the form *Terminates*$(\alpha, \beta, \tau)$,
3. $\Theta_{rel}(\alpha, \beta, \tau)$, if $\Lambda$ is of the form *Releases*$(\alpha, \beta, \tau)$,
4. $\Xi_{traj}(\beta_1, \tau_1, \beta_2, \tau_2)$, if $\Lambda$ is of the form *Trajectory*$(\beta_1, \tau_1, \beta_2, \tau_2)$,
5. $\Xi_{anti}(\beta_1, \tau_1, \beta_2, \tau_2)$, if $\Lambda$ is of the form *AntiTrajectory*$(\beta_1, \tau_1, \beta_2, \tau_2)$, and
6. $\Lambda$ with each intermediate atom $\lambda$ replaced with *Expand*$[\lambda; \Theta_{init}; \Theta_{term}; \Theta_{rel}; \Xi_{traj}; \Xi_{anti}]$, otherwise.

DEFINITION 4.20
If $\Sigma_{init}$, $\Sigma_{term}$, $\Sigma_{rel}$, $\Theta_{init}$, $\Theta_{term}$, $\Theta_{rel}$, $\Delta$, $\Psi$, $\Pi_{traj}$, $\Pi_{anti}$, $\Xi_{traj}$, $\Xi_{anti}$, and $\Gamma$ are as defined above, then *Encode* is defined as follows:

*Encode*$[\mathrm{DEC}; \Sigma_{init}; \Sigma_{term}; \Sigma_{rel}; \Delta; \Psi; \Pi_{traj}; \Pi_{anti}; \Gamma] =$
*Expand*$[\mathrm{DEC}; \Theta_{init}; \Theta_{term}; \Theta_{rel}; \Xi_{traj}; \Xi_{anti}] \wedge$ *CIRC*$[\Delta;$ *Happens*$] \wedge \Psi \wedge \Gamma$.

LEMMA 4.21
Let $x_1, \ldots, x_n, y_1, \ldots, y_k$ be distinct atoms. Let $X_1, \ldots, X_n, Y_1, \ldots, Y_k$ be propositional formulas not mentioning any of the atoms $x_1, \ldots, x_n, y_1, \ldots, y_k$. Let $Z$ be a propositional formula in conjunctive normal form mentioning all of the atoms $x_1, \ldots, x_n, y_1, \ldots, y_k$. Let $F = [\bigwedge_{i=1}^{n} X_i \Leftrightarrow x_i] \wedge [\bigwedge_{i=1}^{k} Y_i \Rightarrow y_i] \wedge Z$. Let $G$ be $Z$ with each occurrence of $x_i$ replaced by $X_i$ and each occurrence of $y_i$ replaced by $Y_i$. Let $A_G$ be the set of atoms mentioned in $G$. Let $\mathcal{T}_G$ be a truth assignment $\mathcal{T}_G : A_G \rightarrow \{T, F\}$. Let $\overline{\mathcal{T}}_G$ be the extension of $\mathcal{T}_G$ to propositional formulas mentioning the atoms $A_G$. Let $A_F = A_G \cup \{x_1, \ldots, x_n, y_1, \ldots, y_k\}$. Let $\mathcal{T}_F$ be the truth assignment $\mathcal{T}_F : A_F \rightarrow \{T, F\}$ defined as follows:

$$\mathcal{T}_F(u) = \begin{cases} \overline{\mathcal{T}}_G(X_i) & \text{if } u = x_i \text{ for some } i \\ \overline{\mathcal{T}}_G(Y_i) & \text{if } u = y_i \text{ for some } i \\ \overline{\mathcal{T}}_G(u) & \text{otherwise.} \end{cases}$$

Let $\overline{\mathcal{T}}_F$ be the extension of $\mathcal{T}_F$ to propositional formulas mentioning the atoms $A_F$. Then it is the case that

$\overline{\mathcal{T}}_F(F) = T$ if and only if $\overline{\mathcal{T}}_G(G) = T$.

PROOF. Let $Z_1, \ldots, Z_m$ be the conjuncts of $Z$ and $G_1, \ldots, G_m$ be the corresponding conjuncts of $G$. We prove each direction separately.
($\Rightarrow$) Suppose $\overline{\mathcal{T}}_F(F) = T$. We must show that for every $i \,\epsilon\, \{1, \ldots, m\}$, $\overline{\mathcal{T}}_G(G_i) = T$. Let $i$ be an arbitrary element of $\{1, \ldots, m\}$. From $\overline{\mathcal{T}}_F(Z) = T$, which follows from $\overline{\mathcal{T}}_F(F) = T$, it follows that $\overline{\mathcal{T}}_F(Z_i) = T$. Let $L_1, \ldots, L_p$ be the disjuncts of $Z_i$ and $M_1, \ldots, M_p$ be the corresponding disjuncts of $G_i$. From the definition of $\overline{\mathcal{T}}_F$, it follows that $\bigwedge_{j=1}^{p} \overline{\mathcal{T}}_F(L_j) = \overline{\mathcal{T}}_G(M_j)$. From this and $\overline{\mathcal{T}}_F(\bigvee_{j=1}^{p} L_j) = T$, it follows that $\overline{\mathcal{T}}_G(\bigvee_{j=1}^{p} M_j) = T$, as required.
($\Leftarrow$) Suppose $\overline{\mathcal{T}}_G(G) = T$. We must show that (1) for every $i \,\epsilon\, \{1, \ldots, n\}$, $\overline{\mathcal{T}}_F(X_i \Leftrightarrow x_i) = T$, (2) for every $i \,\epsilon\, \{1, \ldots, k\}$, $\overline{\mathcal{T}}_F(Y_i \Rightarrow y_i) = T$, and (3) for every $i \,\epsilon\, \{1, \ldots, m\}$, $\overline{\mathcal{T}}_F(Z_i) = T$.
(1) Let $i$ be an arbitrary element of $\{1, \ldots, n\}$. From the definition of $\overline{\mathcal{T}}_F$, it follows that $\overline{\mathcal{T}}_F(X_i)$ and $\overline{\mathcal{T}}_F(x_i)$ have the same truth value.
(2) Let $i$ be an arbitrary element of $\{1, \ldots, k\}$. From the definition of $\overline{\mathcal{T}}_F$, it follows that $\overline{\mathcal{T}}_F(Y_i) = F$ or $\overline{\mathcal{T}}_F(y_i) = T$.
(3) Let $i$ be an arbitrary element of $\{1, \ldots, m\}$. From $\overline{\mathcal{T}}_G(G) = T$, it follows that $\overline{\mathcal{T}}_G(G_i) = T$. Let $M_1, \ldots, M_p$ be the disjuncts of $G_i$ and $L_1, \ldots, L_p$ be the corresponding disjuncts of $Z_i$. From the definition of $\overline{\mathcal{T}}_F$, it follows that $\bigwedge_{j=1}^{p} \overline{\mathcal{T}}_G(M_j) = \overline{\mathcal{T}}_F(L_j)$. From this and $\overline{\mathcal{T}}_G(\bigvee_{j=1}^{p} M_j) = T$, it follows that $\overline{\mathcal{T}}_F(\bigvee_{j=1}^{p} L_j) = T$, as required. ∎

DEFINITION 4.22
A truth assignment $\mathcal{T} : U \to \{T, F\}$ with a set $V$ removed is defined as a truth assignment $\mathcal{T}' : [U - V] \to \{T, F\}$ such that for every $\lambda \,\epsilon\, U - V$, $\mathcal{T}'(\lambda) = \mathcal{T}(\lambda)$.

DEFINITION 4.23
The *grounding* of a formula $\Lambda$ is a formula obtained from $\Lambda$ by instantiating quantifiers and simplifying.

We now proceed to the equivalence theorem:

THEOREM 4.24
Restrict the logic to a finite universe as specified above. If $\Sigma_{init}$, $\Sigma_{term}$, $\Sigma_{rel}$, $\Theta_{init}$, $\Theta_{term}$, $\Theta_{rel}$, $\Delta$, $\Psi$, $\Pi_{traj}$, $\Pi_{anti}$, $\Xi_{traj}$, $\Xi_{anti}$, and $\Gamma$ are as defined above, then the satisfying truth assignments with intermediate atoms removed of:

$$E = \mathit{CIRC}[\Sigma_{init} \wedge \Sigma_{term} \wedge \Sigma_{rel}; \mathit{Initiates}, \mathit{Terminates}, \mathit{Releases}] \wedge$$
$$\mathit{CIRC}[\Delta; \mathit{Happens}] \wedge \Psi \wedge \Pi_{traj} \wedge \Pi_{anti} \wedge \Gamma \wedge \text{EC}$$

are the same as the satisfying truth assignments with intermediate atoms removed of:

$$D = \mathit{Encode}[\text{DEC}; \Sigma_{init}; \Sigma_{term}; \Sigma_{rel}; \Delta; \Psi; \Pi_{traj}; \Pi_{anti}; \Gamma].$$

PROOF. Let $x_1, \ldots, x_n$ be all the atoms in the grounding of $E$ of the form *Initiates*$(\alpha, \beta, \tau)$, *Terminates*$(\alpha, \beta, \tau)$, and *Releases*$(\alpha, \beta, \tau)$. Let $y_1, \ldots, y_k$ be all the atoms in the grounding of $E$ of the form *Trajectory*$(\beta_1, \tau_1, \beta_2, \tau_2)$ and *AntiTrajectory*$(\beta_1, \tau_1, \beta_2, \tau_2)$. Let $X_i = \mathit{Expand}[x_i; \Theta_{init}; \Theta_{term}; \Theta_{rel}; \Xi_{traj}; \Xi_{anti}]$. Let $Y_i = \mathit{Expand}[y_i; \Theta_{init}; \Theta_{term}; \Theta_{rel}; \Xi_{traj}; \Xi_{anti}]$. Let $Z$ be the grounding of $\mathit{CIRC}[\Delta; \mathit{Happens}] \wedge \Psi \wedge \Gamma \wedge \mathrm{DEC}$ written in conjunctive normal form. Let $F = [\bigwedge_{i=1}^{n} X_i \Leftrightarrow x_i] \wedge [\bigwedge_{i=1}^{k} Y_i \Rightarrow y_i] \wedge Z$. Let $G = \mathit{Expand}[Z; \Theta_{init}; \Theta_{term}; \Theta_{rel}; \Xi_{traj}; \Xi_{anti}]$. Let $A_G$ be the set of atoms mentioned in $G$. Let $\mathcal{T}_G$ be a truth assignment $\mathcal{T}_G : A_G \rightarrow \{T, F\}$. Let $\overline{\mathcal{T}}_G$ be the extension of $\mathcal{T}_G$ to propositional formulas mentioning the atoms $A_G$. Let $A_F = A_G \cup \{x_1, \ldots, x_n, y_1, \ldots, y_k\}$. Let $\mathcal{T}_F$ be the truth assignment $\mathcal{T}_F : A_F \rightarrow \{T, F\}$ defined as follows:

$$\mathcal{T}_F(u) = \begin{cases} \overline{\mathcal{T}}_G(X_i) & \text{if } u = x_i \text{ for some } i \\ \overline{\mathcal{T}}_G(Y_i) & \text{if } u = y_i \text{ for some } i \\ \overline{\mathcal{T}}_G(u) & \text{otherwise.} \end{cases}$$

Let $\overline{\mathcal{T}}_F$ be the extension of $\mathcal{T}_F$ to propositional formulas mentioning the atoms $A_F$. From Theorems 4.14 and 4.16, and the definition of *Expand*, we have

$$\mathit{CIRC}[\Sigma_{init} \wedge \Sigma_{term} \wedge \Sigma_{rel}; \mathit{Initiates}, \mathit{Terminates}, \mathit{Releases}] \Leftrightarrow [\bigwedge_{i=1}^{n} X_i \Leftrightarrow x_i]. \tag{4.4}$$

From the definition of *Expand*, we have

$$[\Pi_{traj} \wedge \Pi_{anti}] \Leftrightarrow [\bigwedge_{i=1}^{k} Y_i \Rightarrow y_i]. \tag{4.5}$$

From Theorem 3.9 we have $EC \Leftrightarrow DEC$. From (4.4), (4.5), $Z \Leftrightarrow \mathit{CIRC}[\Delta; \mathit{Happens}] \wedge \Psi \wedge \Gamma \wedge \mathrm{DEC}$, and $EC \Leftrightarrow DEC$, we have $E \Leftrightarrow F$. From the definition of *Encode*, we have $D \Leftrightarrow \mathit{Expand}[\mathrm{DEC}; \Theta_{init}; \Theta_{term}; \Theta_{rel}; \Xi_{traj}; \Xi_{anti}] \wedge \mathit{CIRC}[\Delta; \mathit{Happens}] \wedge \Psi \wedge \Gamma$. From the definition of *Expand* and since $\mathit{CIRC}[\Delta; \mathit{Happens}] \wedge \Psi \wedge \Gamma$ does not mention any intermediate atoms, we have $D \Leftrightarrow \mathit{Expand}[\mathrm{DEC} \wedge \mathit{CIRC}[\Delta; \mathit{Happens}] \wedge \Psi \wedge \Gamma; \Theta_{init}; \Theta_{term}; \Theta_{rel}; \Xi_{traj}; \Xi_{anti}]$. From $Z \Leftrightarrow \mathit{CIRC}[\Delta; \mathit{Happens}] \wedge \Psi \wedge \Gamma \wedge \mathrm{DEC}$ we have $D \Leftrightarrow \mathit{Expand}[Z; \Theta_{init}; \Theta_{term}; \Theta_{rel}; \Xi_{traj}; \Xi_{anti}]$. From this, we have $D \Leftrightarrow G$. From Lemma 4.21, $E \Leftrightarrow F$, and $D \Leftrightarrow G$, we have $\overline{\mathcal{T}}_F(E) = T$ if and only if $\overline{\mathcal{T}}_G(D) = T$. From this, it follows that the satisfying truth assignments with intermediate atoms removed of E are the same as the satisfying truth assignments with intermediate atoms removed of D, as required. ■

## 4.6 Deduction

We may use our encoding method to perform deduction:

THEOREM 4.25
Restrict the logic to a finite universe as specified above. Let $\Sigma_{init}$, $\Sigma_{term}$, $\Sigma_{rel}$, $\Delta$, $\Psi$, $\Pi_{traj}$, $\Pi_{anti}$, and $\Gamma$ be as defined above. If $\Gamma'$ is a state description, then

$$\begin{aligned}&\mathit{CIRC}[\Sigma_{init} \wedge \Sigma_{term} \wedge \Sigma_{rel}; \mathit{Initiates}, \mathit{Terminates}, \mathit{Releases}] \wedge \\ &\mathit{CIRC}[\Delta; \mathit{Happens}] \wedge \Psi \wedge \Pi_{traj} \wedge \Pi_{anti} \wedge \Gamma \wedge \mathrm{EC} \models \Gamma'\end{aligned}$$

if and only if

$$\mathit{Encode}[\mathrm{DEC}; \Sigma_{init}; \Sigma_{term}; \Sigma_{rel}; \Delta; \Psi; \Pi_{traj}; \Pi_{anti}; \Gamma] \models \Gamma'.$$

PROOF. This follows from Theorem 4.24, since none of the atoms of $\Gamma'$ are intermediate. ∎

There are two ways to determine whether the encoding entails $\Gamma'$. (1) We may run a complete satisfiability solver on our encoding and the negation of $B(\Gamma')$. The encoding entails $\Gamma'$ iff the solver does not find any satisfying truth assignments. (2) We may run a complete satisfiability solver on our encoding, producing a set of satisfying truth assignments. The encoding entails $\Gamma'$ iff for every satisfying truth assignment, for every conjunct $\lambda$ of $\Gamma'$, $B(\lambda)$ is assigned to $T$. This method has the benefit of filling in additional information (model finding).

## 4.7 Abduction

We may also use our encoding method to perform event calculus abduction [36, 4, 42]. We start with some definitions:

DEFINITION 4.26
A *goal* is a state description.

DEFINITION 4.27
Let $\Sigma_{init}$, $\Sigma_{term}$, $\Sigma_{rel}$, $\Psi$, $\Pi_{traj}$, $\Pi_{anti}$, and $\Gamma$ be as defined above. Let $\Gamma'$ be a goal, $\Delta_{occ}$ be an event occurrence description, and $\Delta_{trig}$ be a trigger description. $\Delta_{occ}$ is a *plan* for $\Gamma'$ if and only if

$$\begin{aligned}&\mathit{CIRC}[\Sigma_{init} \wedge \Sigma_{term} \wedge \Sigma_{rel}; \mathit{Initiates}, \mathit{Terminates}, \mathit{Releases}] \wedge \\ &\mathit{CIRC}[\Delta_{trig} \wedge \Delta_{occ}; \mathit{Happens}] \wedge \Psi \wedge \Pi_{traj} \wedge \Pi_{anti} \wedge \Gamma \wedge \mathrm{EC} \models \Gamma'.\end{aligned}$$

THEOREM 4.28
Restrict the logic to a finite universe as specified above. Let $\Sigma_{init}$, $\Sigma_{term}$, $\Sigma_{rel}$, $\Psi$, $\Pi_{traj}$, $\Pi_{anti}$, and $\Gamma$ be as defined above. Let $\Gamma'$ be a goal and $\Delta_{trig}$ be a trigger description. The following algorithm finds all plans for $\Gamma'$:

1. Create an empty list of plans.
2. For each satisfying truth assignment $\mathcal{T}$ of
   *Encode*[DEC; $\Sigma_{init}$; $\Sigma_{term}$; $\Sigma_{rel}$; $\Delta_{trig}$; $\Psi$; $\Pi_{traj}$; $\Pi_{anti}$; $\Gamma \wedge \Gamma'$].
   (a) Let $\Delta_{occ}$ be an event occurrence description constructed from the set of all ground atoms $\lambda$ of the form *Happens*$(\alpha, \tau)$ such that $\mathcal{T}(\lambda) = T$.
   (b) If *Encode*[DEC; $\Sigma_{init}$; $\Sigma_{term}$; $\Sigma_{rel}$; $\Delta_{trig} \wedge \Delta_{occ}$; $\Psi$; $\Pi_{traj}$; $\Pi_{anti}$; $\Gamma$] $\models \Gamma'$, then add $\Delta_{occ}$ to the list of plans.

PROOF. This follows from Theorem 4.24, since none of the atoms of $\Gamma'$ are intermediate and for every $\Delta_{occ}$ none of the atoms of $\Delta_{occ}$ are intermediate. ∎

Thus we find all plans for $\Gamma'$ as follows. We first run a complete satisfiability solver on our encoding augmented with $\Gamma'$. For each satisfying truth assignment, we form a candidate plan $\Delta_{occ}$ consisting of a set of *Happens* atoms. We then run the complete solver on the encoding augmented with $\Delta_{occ}$ and the negation of $\Gamma'$. $\Delta_{occ}$ is a plan for $\Gamma'$ iff the solver does not find any satisfying truth assignments.

## 5 Evaluation

In this section, we evaluate our method. First, we compare the method to that of Shanahan and Witkowski [44]. Second, we evaluate our method and that of Shanahan and Witkowski on a set of

14 event calculus problems. Third, we compare the performance of our method to that of the causal calculator [10], a tool for reasoning about action and change using the language of causal theories [22].

## 5.1 Comparison with Shanahan and Witkowski's method

Table 1 compares the coverage of our method for satisfiability-based event calculus reasoning and that of Shanahan and Witkowski [44]. Causal constraints deal with the instantaneous propagation of interacting indirect effects, as in idealized electronic circuits [45]. Our method handles problems involving causal constraints provided that four new predicates and four new axioms are added to the formulation of the event calculus, as described by Shanahan [41].

Our method handles problems involving concurrent events with cumulative or canceling effects [39, pp. 301–304] provided that the problems are formulated in the style of Miller and Shanahan [26, pp. 460–461]. Since our method supports effect axioms with conditions, fluents that are released from the commonsense law of inertia, and incompletely specified initial situations, our method supports the use of determining fluents to enable events with nondeterministic effects [40].

Neither method supports disjunctive event axioms [39, pp. 342–345] and neither method supports compound event axioms [40]. The circumscription of *Happens* in such axioms cannot be computed using Theorem 4.14 since *Happens* is mentioned in $\Gamma(x_1, \ldots, x_n)$.

Event precondition axioms are formulas of the form

$$\mathit{Happens}(a, t) \Rightarrow \text{condition over } t.$$

Event precondition axioms may be used in our method by incorporating them into $\Psi$, with the caveat that if the initial situation is not completely specified, then *Happens*$(a, t)$ becomes a plan for the condition over $t$ (see the discussion of Miller and Shanahan [26, p. 465]). Fluent precondition axioms are the same as effect axioms with conditions.

## 5.2 Evaluation on event calculus benchmark problems

We have implemented our method within a tool for satisfiability-based reasoning in the event calculus [30]. The entire implementation consists of about 10,000 lines of code, with the critical portions (about 4,000 lines) written in the C language for maximum runtime efficiency. The tool invokes the Relsat 2.0 complete satisfiability solver [2].

We conducted an evaluation of our encoding method and the method of Shanahan and Witkowski [44] on a set of 14 benchmark reasoning problems that have been described for the event calculus by Shanahan [39, 40]. Table 2 provides the results of the evaluation. For each problem, the presence of a mark indicates that the method is able to handle, and in the case of our encoding method was successfully able to solve, the problem. Our encoding method was able to solve 11 of the 14 problems; the previous method handles only one due to its limited coverage of the event calculus. Our encoding method was not able to handle the problems involving disjunctive event axioms, compound events, and effect constraints because it does not support those features of the event calculus. In order to run using our method, SUPERMARKETTROLLEY was reformulated using the method of Miller and Shanahan [26, pp. 460–461]. The problems were solved in less than one second.

TABLE 1. Coverage of event calculus satisfiability encoding methods (S&W = Shanahan and Witkowski [44])

| **Feature of the event calculus** | **S&W** | **Us** |
|---|---|---|
| causal constraints | | $\surd$ |
| compound event axioms | | |
| concurrent events | | $\surd$ |
| continuous time | | |
| determining fluents for nondeterminism | | $\surd$ |
| discrete time | $\surd$ | $\surd$ |
| disjunctive event axioms | | |
| effect axioms without conditions | $\surd$ | $\surd$ |
| effect axioms with conditions | | $\surd$ |
| effect constraints | | |
| event precondition axioms | $\surd$ | $\surd$ |
| fluent precondition axioms | | $\surd$ |
| gradual change axioms | | $\surd$ |
| incompletely specified initial situations | | $\surd$ |
| release from the commonsense law of inertia | | $\surd$ |
| state constraints | | $\surd$ |
| three-argument *Happens* | | |
| trigger axioms | | $\surd$ |

### *5.3 Evaluation on zoo world problems*

We conducted a performance comparison of our tool and the causal calculator (CCALC) [10]. We performed the comparison using a collection of zoo world problems proposed by Erik Sandewall and formalized in the language of CCALC [1]. We translated the CCALC formalization of the zoo world into the event calculus, and used our tool to solve the same set of zoo world test problems solved by CCALC. The CCALC formalization consists of 62 causal laws and our event calculus translation consists of 78 axioms.

Table 3 provides the results of the comparison. The performance of our tool on the test problems is comparable to that of CCALC. The columns of this table are: (1) the number of variables in the satisfiability problem, (2) the number of clauses in the problem, (3) the time taken to encode the problem, and (4) the time taken by the Relsat 2.0 satisfiability solver to solve the problem. Encoding and solution times are elapsed wall-clock time in seconds on a machine with a 1.8 GHz Intel Pentium 4 processor and 512 megabytes of RAM. The CCALC encoding time is the sum of the grounding and completion times. The CCALC runs were performed with CCALC 2.0 beta 8.3 and SWI-Prolog 5.0.10.

## 6 Conclusion

We have described a method for encoding reasoning problems of a discrete version of the classical logic event calculus in propositional conjunctive normal form, enabling the problems to be solved efficiently by off-the-shelf complete satisfiability solvers. The method has been implemented as a

TABLE 2. Event calculus benchmark problems solved by satisfiability encoding methods (S&W = Shanahan and Witkowski [44])

| **Problem** | **Reasoning type** | **Notable features** | **S&W** | **Us** |
|---|---|---|---|---|
| BUSRIDE<br>[39, pp. 359–361] | abduction | abduction<br>disjunctive event axiom<br>state constraint | | |
| CHESSBOARD<br>[40] | model finding | determining fluent<br>state constraint | | $\surd$ |
| COINTOSS<br>[40] | model finding | determining fluent | | $\surd$ |
| COMMUTER<br>[40] | deduction | compound event | | |
| DEADORALIVE<br>[39, p. 324] | deduction | state constraint | | $\surd$ |
| HAPPY<br>[40] | deduction | state constraint | | $\surd$ |
| KITCHENSINK<br>[39, pp. 326–329] | deduction | release from inertia<br>gradual change<br>trigger axiom | | $\surd$ |
| RUSSIANTURKEY<br>[40] | model finding | release from inertia | | $\surd$ |
| STOLENCAR<br>[39, p. 359] | abduction | abduction | $\surd$ | $\surd$ |
| STUFFYROOM<br>[39, pp. 288–289] | deduction | state constraint | | $\surd$ |
| SUPERMARKETTROLLEY<br>[39, pp. 302–304] | deduction | concurrent event | | $\surd$ |
| THIELSCHERCIRCUIT<br>[40] | deduction | causal constraint | | $\surd$ |
| WALKINGTURKEY<br>[40] | deduction | effect constraint | | |
| YALE<br>[39, pp. 322–323] | deduction | effect axiom with condition | | $\surd$ |

tool for event calculus reasoning about action and change. The tool successfully solves 11 of 14 benchmark commonsense reasoning problems described for the event calculus and has performance comparable to the causal calculator in the zoo world domain.

Several tools now exist for reasoning about action and change. The most similar ones to ours are the causal calculator, VITAL [7, 19], and $\mathcal{E}$-RES [14, 15]. $\mathcal{E}$-RES is inspired by the event calculus and a mapping between the event calculus and $\mathcal{E}$ has been described [26].

The advantages of our tool are its efficiency and ease of use due to the familiarity of the classical logic event calculus, which is a straightforward extension of first-order logic. The disadvantages of our tool are that it does not support compound events, continuous time, disjunctive event axioms, and effect constraints.

TABLE 3. Comparison with CCALC (c) on zoo problems (wall times in seconds)

| **Problem** | **Variables** | **Clauses** | **Encode** | **Solve** |
|---|---|---|---|---|
| ZOOTEST1 | 3,609 | 23,355 | 29.43 | 0.92 |
| c | 2,693 | 31,881 | 14.83 | 12.24 |
| ZOOTEST2 | 1,072 | 5,587 | 2.44 | 0.14 |
| c | 1,116 | 8,870 | 3.67 | 0.17 |
| ZOOTEST3 | 1,989 | 12,370 | 10.70 | 0.44 |
| c | 1,726 | 17,895 | 15.08 | 0.61 |
| ZOOTEST4.1 | 3,609 | 23,352 | 29.38 | 1.06 |
| c | 2,770 | 32,193 | 14.98 | 4.14 |
| ZOOTEST4.2 | 4,443 | 28,993 | 43.81 | 1.67 |
| c | 3,292 | 39,354 | 14.90 | 11.54 |
| ZOOTEST5.1 | 1,812 | 12,692 | 23.97 | 0.77 |
| c | 1,483 | 18,120 | 44.69 | 5.04 |
| ZOOTEST5.2 | 1,812 | 12,694 | 23.90 | 0.68 |
| c | 1,483 | 18,122 | 44.69 | 5.06 |
| ZOOTEST6 | 1,179 | 6,877 | 4.88 | 0.21 |
| c | 1,127 | 10,428 | 14.85 | 1.10 |

We have begun to use the tool to develop applications. We are developing a commonsense knowledge base, or library of reusable event calculus representations of commonsense knowledge, for use with the tool. We are applying the tool and the commonsense knowledge base to the problem of making inferences and filling in missing information in story understanding [25, 28, 29]. Our approach consists of (1) using a semantic parser to build a semantic parse of a story text, (2) feeding the semantic parse to the tool, which produces one or more models of the story, and (3) using the models to answer questions about the story.

In addition to building applications and extending the method to support more features of the event calculus, another area for future work is to parallelize the method to run on a computing grid so that much larger problems can be solved even more quickly.

## References

[1] V. Akman, S. T. Erdogan, J. Lee, V. Lifschitz, and H. Turner. Representing the zoo world and the traffic world in the language of the causal calculator. *Artificial Intelligence*, **153**, 105–140, 2004.

[2] R. J. Bayardo Jr. and R. C. Schrag. Using CSP look-back techniques to solve real world SAT instances. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference*, pp. 203–208, AAAI Press, Menlo Park, CA, 1997.

[3] E. Davis. *Representations of Commonsense Knowledge*. Morgan Kaufmann, San Mateo, CA, 1990.

[4] M. Denecker, L. Missiaen, and M. Bruynooghe. Temporal reasoning with abductive event calculus. In *Proceedings of the Tenth European Conference on Artificial Intelligence*, B. Neumann, ed., pp. 384–388. John Wiley, Chichester, 1992.

[5] DIMACS. Satisfiability suggested format. Technical report, Center for Discrete Mathematics and Theoretical Computer Science, 1993.

[6] K. Doets. *From Logic to Logic Programming*. MIT Press, Cambridge, MA, 1994.

[7] P. Doherty, J. Gustafsson, L. Karlsson, and J. Kvarnström. TAL: Temporal Action Logics language specification and tutorial. *Linköping Electronic Articles in Computer and Information Science*, 3(015), 1998.

[8] D. Du, J. Gu, and P. M. Pardalos, eds. *Satisfiability Problem: Theory and Applications*, volume 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, Boston, MA, 1997.

[9] K. M. Ford and Z. W. Pylyshyn, eds. *The Robot's Dilemma Revisited: The Frame Problem in Artificial Intelligence*. Ablex, Norwood, NJ, 1996.

[10] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic causal theories. *Artificial Intelligence*, **153**, 49–104, 2004.

[11] E. Giunchiglia and R. Sebastiani. Applying the Davis-Putnam procedure to non-clausal formulas. In *Proceedings of the Sixth Congress of the Italian Association for Artificial Intelligence*, Bologna, 1999.

[12] A. R. Haas. The case for domain-specific frame axioms. In *The Frame Problem in Artificial Intelligence: Proceedings of the 1987 Workshop*, Frank M. Brown, ed., pp. 343–348. Morgan Kaufmann, Los Altos, CA, 1987.

[13] D. Jackson. Automating first-order relational logic. In *Proceedings of the Eighth International Symposium on the Foundations of Software Engineering*, pp. 130–139. ACM, New York, 2000.

[14] A. Kakas, R. Miller, and F. Toni. An argumentation framework for reasoning about actions and change. In *Proceedings of the Fifth International Conference on Logic Programming and Nonmonotonic Reasoning*, M. Gelfond, N. Leone, and G. Pfeifer, eds. Volume 1730 of *Lecture Notes in Computer Science*, pp. 78–91. Springer-Verlag, Heidelberg, 1999.

[15] A. Kakas, Rob Miller, and Francesca Toni. E-RES - A system for reasoning about actions, events and observations. In *Proceedings of the Eighth International Workshop on Non-Monotonic Reasoning*, 2000.

[16] H. Kautz and B. Selman. Planning as satisfiability. In *Proceedings of the Tenth European Conference on Artificial Intelligence*, B. Neumann, ed., pp. 359–363. John Wiley, Chichester, UK, 1992.

[17] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Annual Conference on Innovative Applications of Artificial Intelligence*, pp. 1194–1201, Menlo Park, CA, 1996. AAAI Press.

[18] R. A. Kowalski and M. J. Sergot. A logic-based calculus of events. *New Generation Computing*, **4**, 67–95, 1986.

[19] J. Kvarnström. VITAL: Visualization and implementation of temporal action logic. Technical report, Department of Computer and Information Science, Linköping University, 2001.

[20] F. Lévy and J. J. Quantz. Representing beliefs in a situated event calculus. In *Proceedings of the Thirteenth European Conference on Artificial Intelligence*, H. Prade, ed., pp. 547–551. John Wiley, Chichester, 1998.

[21] V. Lifschitz. Circumscription. In *Handbook of Logic in Artificial Intelligence and Logic Programming, volume 3: Nonmonotonic Reasoning and Uncertain Reasoning*, D. M. Gabbay, C. J. Hogger, and J. A. Robinson, eds, pp. 298–352. Oxford University Press, Oxford, 1994.

[22] N. McCain and H. Turner. Causal theories of action and change. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference*, pp. 460–465, Menlo Park, CA, 1997. AAAI Press.

[23] J. McCarthy. Circumscription—A form of non-monotonic reasoning. *Artificial Intelligence*, **13**, 27–39, 1980.

[24] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*, D. Michie and B. Meltzer, eds., pp. 463–502. Edinburgh University Press, Edinburgh, Scotland, 1969.

[25] J. McCarthy, M. Minsky, A. Sloman, L. Gong, T. Lau, L. Morgenstern, E. T. Mueller, D. Riecken, M. Singh, and P. Singh. An architecture of diversity for commonsense reasoning. *IBM Systems Journal*, **41**, 530–539, 2002.

[26] R. Miller and M. Shanahan. Some alternative formulations of the event calculus. In *Computational Logic: Logic Programming and Beyond: Essays in Honour of Robert A. Kowalski, Part II*, A. C. Kakas and F. Sadri, eds. Volume 2408 of *Lecture Notes in Computer Science*, pp. 452–490. Springer-Verlag, Heidelberg, 2002.

[27] L. Morgenstern. Mid-sized axiomatizations of commonsense problems: A case study in egg cracking. *Studia Logica*, **67**, 333–384, 2001.

[28] E. T. Mueller. Story understanding. In *Encyclopedia of Cognitive Science*, L. Nadel, ed., volume 4, pp. 238–246. Nature Publishing Group, London, 2002.

[29] E. T. Mueller. Story understanding through multi-representation model construction. In *Text Meaning: Proceedings of the HLT-NAACL 2003 Workshop*, G. Hirst and S. Nirenburg, eds, pp. 46–53, East Stroudsburg, PA, 2003. Association for Computational Linguistics.

[30] E. T. Mueller. A tool for satisfiability-based commonsense reasoning in the event calculus. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*. AAAI Press, Menlo Park, CA, 2004.

[31] A. Nerode and R. A. Shore. *Logic for Applications*. Springer-Verlag, New York, second edition, 1997.

[32] D. A. Plaisted and S. Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, **2**, 293–304, 1986.

[33] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, V. Lifschitz, ed., pp. 359–380. Academic Press, San Diego, 1991.

[34] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, Cambridge, MA, 2001.

[35] L. K. Schubert. Monotonic solution of the frame problem in the situation calculus: An efficient method for worlds with fully specified actions. In *Knowledge Representation and Defeasible Reasoning*, H. E. Kyburg Jr., R. P. Loui, and G. N. Carlson, eds., pp. 23–67. Kluwer, Dordrecht, 1990.

[36] M. Shanahan. Prediction is deduction but explanation is abduction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, N. S. Sridharan, ed., pp. 1055–1060. Morgan Kaufmann, San Mateo, CA, 1989.

[37] M. Shanahan. A circumscriptive calculus of events. *Artificial Intelligence*, **77**, 249–284, 1995.

[38] M. Shanahan. Robotics and the common sense informatic situation. In *Proceedings of the Twelfth European Conference on Artificial Intelligence*, W. Wahlster, ed., pp. 684–688. John Wiley, Chichester, 1996.

[39] M. Shanahan. *Solving the Frame Problem*. MIT Press, Cambridge, MA, 1997.

[40] M. Shanahan. The event calculus explained. In *Artificial Intelligence Today*, M. J. Wooldridge and M. M. Veloso, eds. volume 1600 of *Lecture Notes in Computer Science*, pp. 409–430. Springer-Verlag, Heidelberg, 1999.

[41] M. Shanahan. The ramification problem in the event calculus. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pp. 140–146. Morgan Kaufmann, San Mateo, CA, 1999.

[42] M. Shanahan. An abductive event calculus planner. *Journal of Logic Programming*, **44**, 207–240, 2000.

[43] M. Shanahan. An attempt to formalise a non-trivial benchmark problem in common sense reasoning. *Artificial Intelligence*, **153**, 141–165, 2004.

[44] M. Shanahan and M. Witkowski. Event calculus planning through satisfiability. *Journal of Logic and Computation*, **14**, 731–746, 2004.

[45] M. Thielscher. Ramification and causality. *Artificial Intelligence*, **89**, 317–364, 1997.

Received 17 June 2003