Event Calculus Planning Through Satisfiability

MURRAY SHANAHAN and MARK WITKOWSKI, *Department of Electrical and Electronic Engineering, Imperial College, Exhibition Road, London SW7 2BT, UK.*
E-mail: {*m.shanahan,m.witkowski*}*@imperial.ac.uk*

Abstract

Previous work on event calculus planning has viewed it as an abductive task, whose implementation through abductive logic programming reduces to partial order planning. By contrast, this paper presents a formal framework for tackling event calculus planning through satisfiability, in the style of Kautz and Selman. A provably correct conjunctive normal form encoding for event calculus planning problems is supplied, rendering them soluble by an off-the-shelf SAT solver.

Keywords: Event calculus, satisfiability planning.

1 Introduction

The event calculus is a well-known logic-based formalism for reasoning about actions for which a robust solution to the frame problem has been developed [21]. It can be used to represent a rich array of phenomena, including concurrent actions, actions with non-deterministic effects, actions with indirect effects (ramifications), and continuous change [21]. It has found application in a variety of areas, including natural language processing [13], intelligent agents [8], common sense reasoning [17], and robotics [20].

In all previous research known to the present authors, such as [4, 2, 16, 7] and [23], planning with the event calculus is considered as an abductive logic programming task. As shown in [23], the computation carried out by these abductive event calculus planners mirrors closely that carried out by partial order planners, such as UCPOP [19].[1] Since the performance of partial order planners compares poorly with that of more recent systems, such as Graphplan [1] and SATPLAN [11], it is natural to enquire whether a more up-to-date planner can be developed for the event calculus. This paper presents the formal foundations for one such planner, based on the idea of planning as satisfiability developed by Kautz and Selman [10, 11].

The main achievement of this paper is to define a provably correct encoding of a restricted class of event calculus planning problems into a propositional conjunctive normal form (CNF) suitable for submission to a SAT solver. The aim is not to develop a novel way of encoding planning problems in propositional CNF. Rather, the goal is to supply a correct mapping from the event calculus to a style of propositional CNF that has *already* been shown to be computationally effective in published work on planning as satisfiability. The difficulty here is due to the distinctive language of the event calculus which, not being state-based, is representationally very different from the kind of propositional CNF formula into which it needs to be translated.

[1]An exception is the event calculus planner described by Kakas *et al.* [9], which combines constraint solving technology with abduction to achieve a significantly better performance than traditional abductive event calculus planners. Moreover, as shown in [15], there is a close relationship between deductive and abductive planning for the event calculus.

2 The event calculus

The version of the event calculus employed here is based on many-sorted predicate calculus augmented with circumscription [21]. The event calculus has a number of variants [21, 15]. Our strategy here will be to present, in its full generality, a version of the calculus that has been put to practical use in several previously published papers (such as [20] and [17]), and then to restrict it for the purposes of the present paper.

The language of the calculus includes sorts for objects, fluents, actions (events), and time points. The time points are interpreted by the naturals, and the usual comparative predicates are taken for granted. The predicate symbols presented in Table 1 are also included in the language.

TABLE 1. The language of the event calculus

Formula	**Meaning**
Initiates(α,β,τ)	Fluent β holds after action α at time τ
Terminates(α,β,τ)	Fluent β does not hold after action α at time τ
Releases(α,β,τ)	Fluent β is not subject to the common sense law of inertia after action α at time τ
Initially$_P(\beta)$	Fluent β holds from time 0
Initially$_N(\beta)$	Fluent β does not hold from time 0
Happens(α,τ_1,τ_2)	Action α starts at time τ_1 and ends at time τ_2
HoldsAt(β,τ)	Fluent β holds at time τ
Clipped(τ_1,β,τ_2)	Fluent β is terminated between times τ_1 and τ_2
Declipped(τ_1,β,τ_2)	Fluent β is initiated between times τ_1 and τ_2

The language includes a countable set of constant symbols denoting time points. The language also includes a finite set O of object-valued constant symbols, a finite set F of fluent-valued function symbols, and a finite set A of action-valued function symbols. The following definition ensures that the set of ground fluent terms and the set of ground action terms are both finite.

DEFINITION 2.1
An *object term* is either a variable or a member of O. A *fluent term* is either a variable or has the form f(o$_1 \ldots$o$_n$) for some $n > 0$, where f $\in F$ and for all $0 \leq i \leq n, \mathrm{o}_i$ is an object term. An *action term* is either a variable or has the form a(o$_1 \ldots$o$_n$) for some $n > 0$, where a $\in A$ for all $0 \leq i \leq n, \mathrm{o}_i$ is an object term.

We have the following axioms, whose conjunction is denoted EC. [2]

HoldsAt(f,t) $\leftarrow$ Initially$_P$(f) $\wedge \neg$ Clipped(0,f,t) (EC1)

HoldsAt(f,t3) $\leftarrow$ (EC2)
Happens(a,t1,t2) $\wedge$ Initiates(a,f,t1) $\wedge$
t2 $<$ t3 $\wedge \neg$ Clipped(t1,f,t3)

[2] All variables are universally quantified with maximum scope unless otherwise indicated.

$$\text{Clipped(t1,f,t4)} \leftrightarrow \exists \text{ a,t2,t3 [Happens(a,t2,t3)} \wedge \text{t1} < \text{t3} \wedge \text{t2} < \text{t4} \wedge [\text{Terminates(a,f,t2)} \vee \text{Releases(a,f,t2)]]} \tag{EC3}$$

$$\neg \text{ HoldsAt(f,t)} \leftarrow \text{Initially}_N\text{(f)} \wedge \neg \text{ Declipped(0,f,t)} \tag{EC4}$$

$$\neg \text{ HoldsAt(f,t3)} \leftarrow \text{Happens(a,t1,t2)} \wedge \text{Terminates(a,f,t1)} \wedge \text{t2} < \text{t3} \wedge \neg \text{ Declipped(t1,f,t3)} \tag{EC5}$$

$$\text{Declipped(t1,f,t4)} \leftrightarrow \exists \text{ a,t2,t3 [Happens(a,t2,t3)} \wedge \text{t1} < \text{t3} \wedge \text{t2} < \text{t4} \wedge [\text{Initiates(a,f,t2)} \vee \text{Releases(a,f,t2)]]} \tag{EC6}$$

$$\text{Happens(a,t1,t2)} \rightarrow \text{t1} \leq \text{t2} \tag{EC7}$$

A two-argument version of Happens is defined as follows.

$$\text{Happens(a,t)} \equiv_{def} \text{Happens(a,t,t)}$$

The frame problem is overcome through circumscription, as will be shown shortly.

3 Planning as abduction

This section offers a formal abductive definition of event calculus planning. First, we pin down exactly what is meant by a 'domain description'.

DEFINITION 3.1
An *initiates formula* has the form,

$$\text{Initiates}(\alpha,\beta,\text{t}) \leftarrow \psi_1 \wedge ... \wedge \psi_n$$

where each ψ_i is either an inequality or has the form,

$$(\neg) \text{ HoldsAt}(\beta_i,\text{t}).$$

DEFINITION 3.2
A *terminates formula* has the form,

$$\text{Terminates}(\alpha,\beta,\text{t}) \leftarrow \psi_1 \wedge ... \wedge \psi_n$$

where each ψ_i is either an inequality or has the form,

$$(\neg) \text{ HoldsAt}(\beta_i,\text{t}).$$

DEFINITION 3.3
A *releases formula* has the form,

$$\text{Releases}(\alpha,\beta,\text{t}) \leftarrow \psi_1 \wedge ... \wedge \psi_n$$

where each ψ_i is either an inequality or has the form,

(¬) HoldsAt(β_i,t).

If n = 0, the right-hand side of an initiates, terminates or releases formula is omitted.

DEFINITION 3.4
An *effect axiom* is either an initiates formula, a terminates formula, or a releases formula.

In addition to describing the effects of actions, a domain description must describe the preconditions of those actions. In the event calculus, an action's preconditions can be captured in two distinct ways. The first option is to include them as HoldsAt conditions on the right-hand sides of effect axioms. Rather than preventing the execution of an action if its precondition doesn't hold, this method disables the action's effects. This approach is common in the event calculus literature. The second option, which is closer to the conventional planning approach, is to write axioms that rule out the execution of an action altogether unless its preconditions hold. This is the option taken up in the present paper.[3]

DEFINITION 3.5
A *precondition axiom* has the form,

Happens(α,t) $\rightarrow \psi_1 \wedge ... \wedge \psi_n$

where each ψ_i is of the form,

$(\neg)$ HoldsAt(β_i,t).

DEFINITION 3.6
A *domain description* is a finite conjunction of effect axioms and precondition axioms.

In general, the event calculus allows for much richer domain descriptions than fall within the class defined above, encompassing concurrent actions with cumulative or cancelling effects, actions with non-deterministic effects, actions with indirect effects (ramifications), compound actions, and continuous change [15, 21, 22]. Although the ultimate aim of this research is to devise fast planners that function with rich domain descriptions, the first step is to work on the sort of STRIPS-like domains that have been the focus of most work in the planning community.

Next we have the definitions of an initial situation, a goal, and a plan.

DEFINITION 3.7
An *initial situation* is a finite conjunction of formulae of the form,

Initially$_N(\beta)$
or,
Initially$_P(\beta)$

where β is a ground fluent term, in which each fluent name occurs at most once.

[3]This option is only available in the context of a restriction to completely known initial situations and actions with deterministic effects.

In the present paper, we will be concerned with completely known initial situations, which are defined as follows.[4]

DEFINITION 3.8
An initial situation Δ_0 is *complete* if for any ground fluent term β either

$\Delta_0 \models \text{Initially}_N(\beta)$ or $\Delta_0 \models \text{Initially}_P(\beta)$.

In order to avoid having to fully specify a complete initial situation, we have the following operator, which implements a closed world assumption.

DEFINITION 3.9
If Δ_0 is an initial situation, then CLO[Δ_0] is the conjunction of Δ_0 with all formulae of the form $\text{Initially}_N(\beta)$, where β is a ground fluent term, such that

$\Delta_0 \not\models \text{Initially}_P(\beta)$.

The event calculus definition of a goal state is straightforward.

DEFINITION 3.10
A *goal* is a finite conjunction of formulae of the form,

$(\neg)$ HoldsAt(β,τ)

where β is a ground fluent term, and τ is a time point constant.

Finally, a plan, in event calculus terms, is a narrative of actions.

DEFINITION 3.11
A *narrative* is a finite conjunction of formulae of the form,

Happens(α,τ)
or,
$\tau_1 < \tau_2$

where α is a ground action term, and τ, τ_1 and τ_2 are time point constants.

Note that this definition permits partially ordered plans, as well as concurrent actions.

In general, a set of uniqueness-of-names axioms is required for actions and fluents. In what follows, the term UNA[$f_1, f_2, \ldots, f_k$] abbreviates the conjunction of all formulae of the form,

$f_i(x_1 , x_2, \ldots x_m) \neq f_j(y_1, y_2, \ldots, y_n)$

where $i < j < k$, with all formulae of the form,

$f_i(x_1, x_2, \ldots x_n) = f_i(y_1, y_2, \ldots, y_n) \rightarrow [x_1 = y_1 \wedge x_2 = y_2 \wedge \ldots \wedge x_n = y_n]$

where $i < k$.

Now we are ready to characterize event calculus planning.

[4]Only named fluents are considered here. Models can include unnamed fluents, but they have no logical influence on the encoding or its correctness.

DEFINITION 3.12
Let Γ be a goal, let Σ be a domain description, let Δ_0 be an initial situation, and let Ω be the conjunction of a pair of uniqueness-of-names axioms for the actions and fluents mentioned in Σ. A *plan* for Γ is a narrative Δ such that,

$$\text{CIRC}[\Sigma;\ \text{Initiates, Terminates, Releases}] \wedge \text{CIRC}[\Delta;\ \text{Happens}] \wedge \Delta_0 \wedge \text{EC} \wedge \Omega \models \Gamma$$

and,

$$\text{CIRC}[\Sigma;\ \text{Initiates, Terminates, Releases}] \wedge \text{CIRC}[\Delta;\ \text{Happens}] \wedge \Delta_0 \wedge \text{EC} \wedge \Omega$$

is consistent.

By minimizing Initiates, Terminates and Releases we assume that actions have no unexpected effects, and by minimizing Happens we assume that there are no unexpected event occurrences. In all the cases under consideration in this paper, Σ and Δ can be straightforwardly reduced to predicate completions using theorems from Lifschitz [14].

In the following example, Blocks World planning is formulated in the event calculus.

EXAMPLE 3.13
Consider an event calculus language that includes the object constants A, B, C, and Table. It also includes a single action, Move(x,y,z), and two fluents, On(x,y) and Clear(x). Let Σ be the conjunction of the following effect axioms and precondition axiom.

Initiates(Move(x,y,z),On(x,z),t)

Terminates(Move(x,y,z),On(x,y),t)

Initiates(Move(x,y,z),Clear(y),t)

Terminates(Move(x,y,z),Clear(z),t) $\leftarrow$ z $\neq$ Table

Happens(Move(x,y,z),t) $\rightarrow$
  HoldsAt(On(x,y),t) $\wedge$
    HoldsAt(Clear(x),t) $\wedge$ HoldsAt(Clear(z),t)

Let Ω be the conjunction of the following uniqueness-of-names axioms. [5]

UNA[Move]
UNA[On,Clear]
UNA[A,B,C,Table]

Let Δ_0 be the conjunction of the following Initially$_P$ formulae.

Initially$_P$(On(A,B))
Initially$_P$(On(B,C))
Initially$_P$(On(C,Table))

[5] Note that these axioms entail, for example, Move(A,B) $\neq$ Move(A,C).

$\text{Initially}_P(\text{Clear(A)})$
$\text{Initially}_P(\text{Clear(Table)})$

Let the goal Γ be the conjunction of the following HoldsAt formulae.

HoldsAt(On(B,A),T)
HoldsAt(On(A,C),T)
HoldsAt(On(C,Table),T)

Now let Δ be the conjunction of the following Happens and temporal ordering formulae.

Happens(Move(A,B,Table),T1)
Happens(Move(B,C,Table),T2)
Happens(Move(A,Table,C),T3)
Happens(Move(B,Table,A),T4)
$T1 < T2 \qquad T2 < T3$
$T3 < T4 \qquad T4 < T$

Now, we have,

$$\text{CIRC}[\Sigma; \text{Initiates, Terminates, Releases}] \wedge$$
$$\text{CIRC}[\Delta; \text{Happens}] \wedge \text{CLO}[\Delta_0] \wedge \text{EC} \wedge \Omega \models \Gamma.$$

In other words, Δ is a plan.

Now we know exactly what event calculus planning is. But how do we compute plans? The rest of the paper shows how event calculus planning problems can be mapped into the ‘planning as satisfiability’ framework of Kautz and Selman [10], which has resulted in planners that outperform the best of the previous generation of partial order planners [11, 12].

4 From event calculus to CNF

As shown by Kautz and Selman [10, 11], as long as the initial situation is completely known, and all actions have completely known, deterministic effects, a planning problem can be encoded as a conjunctive normal form formula in such a way that any model of the formula corresponds to a plan. By conjoining together the set of all ground instances of this formula, a large propositional CNF formula is obtained that can be submitted to an off-the-shelf SAT solver. Surprisingly, given size of these CNF formulae and the NP-completeness of the SAT problem, Kautz and Selman [11] have demonstrated that certain planning problems can be solved more quickly this way than by other fast planners, such as Graphplan [1], which in turn have been shown to outperform partial order planners such as UCPOP [19]. Although Kautz and Selman’s 1996 [11] results used hand-crafted CNF encodings, more recent experiments have shown that similarly impressive results can be obtained using automatic encoding [12].

This section defines a CNF encoding of event calculus planning problems, as defined in the previous section, in the spirit of Kautz and Selman. The encoding of this section is only defined for a restricted class of problems, namely those with a complete initial situation and a ‘simple’ domain description, which excludes Releases formulae and forbids effect axioms from mentioning the HoldsAt predicate. The exclusion of Releases formulae means that all fluents are subject to inertia at all times, and the restriction on other effect axioms rules out actions with conditional or context-dependent effects.

DEFINITION 4.1
A *simple domain description* is a domain description in which all effect axioms are initiates and terminates formulae whose right-hand-sides do not mention the predicate HoldsAt.

Our attention will be confined to totally ordered plans, reflecting the fact that the CNF encoding to be defined, like those of Kautz and Selman, doesn't support the generation of partially ordered plans.

DEFINITION 4.2
A narrative Δ is *totally ordered* if,

- for every distinct pair of time point constants τ_1 and τ_2, either $\Delta \models \tau_1 < \tau_2$ or $\Delta \models \tau_2 < \tau_1$, and
- for every time point constant τ, there is at most one ground action term α such that $\Delta \models$ Happens(α,τ).

Since the set of time point constants occurring in a totally ordered narrative is isomorphic to a subset of the naturals, standard decimal whole numbers will be used as time point constants from now on, with the usual comparative predicates interpreted accordingly. The time of occurrence of the kth action in a totally ordered narrative will be denoted by the number k. In the following definitions, '+' is used as a symbol in the meta-language. So 'τ+1' means the time constant denoting τ's successor.

The next task is to define a series of functions that encode the various components of an event calculus planning problem into propositional CNF. First, we define a function T$_I$ that deals with the initial situation.

DEFINITION 4.3
If $\Delta_0 = \phi_1 \wedge ... \wedge \phi_n$ is an initial situation, then T$_I$(Δ_0) is,

$$\sigma_1 \wedge ... \wedge \sigma_n$$

where,

$$\sigma_i = \begin{cases} \text{HoldsAt}(\beta_i,0) \text{ if } \phi_i = \text{Initially}_P(\beta_i) \\ \\ \neg\, \text{HoldsAt}(\beta_i,0) \text{ otherwise.} \end{cases}$$

The remaining functions depend on the ideas of substitution and instantiation.

DEFINITION 4.4
A *substitution* is a set of equalities of the form $\xi = \psi$, where ξ is a variable and ψ is a ground term, in which no variable occurs more than once. If the set of variables occurring in a formula ϕ is a subset of the set of variables occurring in a substitution σ, then σ is a *substitution for* ϕ. The *instantiation* of a term or formula ϕ with a substitution σ for ϕ is the result of replacing every variable ξ in ϕ with the corresponding term ψ such that $\xi = \psi \in \sigma$.

Next we define a function T$_C$ for encoding a single effects axiom.

DEFINITION 4.5
Let τ be a time point constant. For any initiates formula Π of the form,

Initiates(α,β,t) $\leftarrow$ Φ

T$_C$(Π,τ) is the conjunction of every instantiation of the formula,

$\neg$ Happens(α,τ) $\vee$ HoldsAt(β,τ+1)

with a substitution σ for that formula which is consistent with the inequalities in Φ.

Similarly, for any terminates formula Π of the form,

Terminates(α,β,t) $\leftarrow$ Φ

T$_C$(Π,τ) is the conjunction of every instantiation of the formula,

$\neg$ Happens(α,τ) $\vee$ $\neg$ HoldsAt(β,τ+1)

with a substitution σ for that formula which is consistent with the inequalities in Φ.

Finally, for any precondition axiom Π of the form,

Happens(α,t) $\rightarrow$ ψ_1 $\wedge$ ... $\wedge$ ψ_n

T$_C$(Π,τ) is the conjunction of every ground instance of the formula,

($\neg$ Happens(α,τ) $\vee$ ψ_1) $\wedge$ ... $\wedge$ ($\neg$ Happens(α,τ) $\vee$ ψ_n)

EXAMPLE 4.6
If Π is the initiates formula,

Initiates(Move(x,y,z),On(x,z),t)

then, T$_C$(Π,1) is the conjunction of every ground instance of the formula,

$\neg$ Happens(Move(x,y,z),1) $\vee$ HoldsAt(On(x,z),2).

If Π is the terminates formula,

Terminates(Move(x,y,z),Clear(z),t) $\leftarrow$ z $\neq$ Table

then, T$_C$(Π,1) is the conjunction of every ground instance, excluding those in which z is bound to Table, of the formula,

$\neg$ Happens(Move(x,y,z),1) $\vee$ $\neg$ HoldsAt(On(x,z),2).

If Π is the precondition axiom,

Happens(Move(x,y,z),t) $\rightarrow$
  HoldsAt(On(x,y),t) $\wedge$ HoldsAt(Clear(x),t) $\wedge$ HoldsAt(Clear(z),t)

then, $T_C(\Pi,1)$ is the conjunction of every ground instance of the formula,

(¬ Happens(Move(x,y,z),1) ∨ HoldsAt(On(x,y),1)) ∧
  (¬ Happens(Move(x,y,z),1) ∨ HoldsAt(Clear(x),1)) ∧
    (¬ Happens(Move(x,y,z),1) ∨ HoldsAt(Clear(z),1))

Building on the definition of T_C, the function T_D encodes all the effect axioms.

DEFINITION 4.7
If τ is a time point constant and $\Sigma = \phi_1 \wedge ... \wedge \phi_n$ is a simple domain description, then $T_D(\Sigma,\tau)$ is,

$T_C(\phi_1,\tau) \wedge ... \wedge T_C(\phi_n,\tau)$.

The formulae in $T_D(\Sigma,\tau)$ only take care of changes in fluents that are brought about by actions. We also need a solution to the frame problem. That is to say, we need to represent the non-effects of actions. Following the recommendation of Ernst *et al.* [3], the mapping below employs explanation closure axioms in the style of Haas [6], as these lead to a more compact encoding than classical McCarthy and Hayes style frame axioms.

DEFINITION 4.8
If τ is a time point constant, Σ is a simple domain description, and β is a ground fluent term, then $E^-(\beta,\tau,\Sigma)$ is the conjunction of every formula of the form,

HoldsAt(β,τ) ∨ ¬ HoldsAt(β,τ+1) ∨ Happens(α_1,τ) ∨ ... ∨ Happens(α_n,τ)

where $\{\alpha_1\ ...\ \alpha_n\}$ is the set of all instantiations of an action term α with a substitution σ for α such that Σ includes the formula Initiates(α,β,t) ← Π and σ is consistent with the inequalities in Π. Similarly, $E^+(\beta,\tau,\Sigma)$ is the conjunction of every formula of the form,

¬ HoldsAt(β,τ) ∨ HoldsAt(β,τ+1) ∨ Happens(α_1,τ) ∨ ... ∨ Happens(α_n,τ)

where $\{\alpha_1\ ...\ \alpha_n\}$ is the set of all instantiations of an action term α with a substitution σ for α such that Σ includes the formula Terminates(α,β,t) ← Π and σ is consistent with the inequalities in Π.

EXAMPLE 4.9
Suppose we have the same language and domain description Σ as in Example 3.13. Then E^-(On(A,B),1,Σ) is the formula,

HoldsAt(On(A,B),1) ∨ ¬ HoldsAt(On(A,B),2) ∨
  Happens(Move(A,Table,B),1) ∨ Happens(Move(A,A,B),1) ∨
    Happens(Move(A,B,B),1) ∨ Happens(Move(A,C,B),1).

(Note that this formula takes into account two impossible actions.)

E^+(On(A,B),1,Σ) is the formula,

¬ HoldsAt(On(A,B),1) ∨ HoldsAt(On(A,B),2) ∨
  Happens(Move(A,B,Table),1) ∨ Happens(Move(A,B,A),1) ∨
    Happens(Move(A,B,B),1) ∨ Happens(Move(A,B,C),1).

(Note that this formula takes into account one impossible action, and one action that initiates the same fluent that it terminates.)

Based on the mappings E^- and E^+, the function T_F encodes all the required explanation closure axioms.

DEFINITION 4.10
If τ is a time point constant and Σ is a simple domain description, then $T_F(\Sigma,\tau)$ is the conjunction of all formulae $E^+(\beta,\tau,\Sigma) \wedge E^-(\beta,\tau,\Sigma)$ such that β is a ground instance of a fluent term mentioned in Σ.

Finally, we need to exclude the possibility of multiple actions occurring at the same time. This is done via the function T_X.

DEFINITION 4.11
Let T_X be the conjunction of every ground instance of the formula,

$$\neg\, \text{Happens}(\alpha_1,\tau) \vee \neg\, \text{Happens}(\alpha_2,\tau)$$

such that $\alpha_1 \neq \alpha_2$.

We're now in a position to define the CNF encoding of an event calculus planning problem.

DEFINITION 4.12
If τ is a time point constant and Γ is a goal, then $T_G(\Gamma,\tau)$ is identical to Γ except that every time point constant occurring in Γ is replaced by τ.

DEFINITION 4.13
If Δ_0 is an initial situation, Σ is a domain description, and Γ is a goal, then $T(\Delta_0,\Sigma,\Gamma,n)$ is,

$$T_I(\Delta_0) \wedge \bigwedge_{i=1}^{n} T_D(\Sigma, i) \wedge \bigwedge_{i=1}^{n} T_F(\Sigma, i) \wedge T_X \wedge T_G(\Gamma, n{+}1).$$

5 The correctness of the encoding

This section presents a theorem showing that the set of solutions to an event calculus planning problem is isomorphic to the set of solutions to its CNF encoding. First, we require a lemma ensuring that, according to the event calculus axioms, the completeness of the initial situation carries over to all subsequent time points.

LEMMA 5.1
Let Δ_0 be a complete initial situation, let Σ be a simple domain description, let Ω be the conjunction of a pair of uniqueness-of-names axioms for the actions and fluents mentioned in Σ, and let Δ be a totally ordered narrative. For any ground fluent term β and time point constant τ, either

$$\begin{aligned}&\text{CIRC}[\Sigma\,;\ \text{Initiates, Terminates, Releases}] \wedge \\ &\quad \text{CIRC}[\Delta;\ \text{Happens}] \wedge \Delta_0 \wedge \text{EC} \wedge \Omega \models \text{HoldsAt}(\beta,\tau)\end{aligned}$$

or,

$$\begin{aligned}&\text{CIRC}[\Sigma\,;\ \text{Initiates, Terminates, Releases}] \wedge \\ &\quad \text{CIRC}[\Delta\,;\ \text{Happens}] \wedge \Delta_0 \wedge \text{EC} \wedge \Omega \models \neg\, \text{HoldsAt}(\beta,\tau).\end{aligned}$$

PROOF. The proof is by induction over the time points. It follows from the completeness of Δ_0 that the lemma holds for $\tau = 0$. Now suppose the lemma holds for $\tau = k$. Then we can show that the lemma holds for $\tau = k + 1$ as follows. Consider any ground fluent term β. There are two cases to consider. Case 1: we have HoldsAt(β, k). From EC, this is either because β held at time point 0 and has not been terminated between 0 and k, or because some event occurred before k that initiated β. Now, if an event occurs at time k that terminates β, then we will have $\neg$ HoldsAt($\beta, k + 1$), from (EC5). But if no event occurs at time k that terminates β, then we will have HoldsAt($\beta, k + 1$), from (EC2). Either way, the lemma holds at time $k + 1$. Case 2: We have not HoldsAt(β, k). The argument is analogous to the first case. ∎

Next, we have a lemma showing that, given any totally ordered narrative, the CNF encoding agrees with the original event calculus formulation on what fluents hold when.

LEMMA 5.2
Let Δ_0 be a complete initial situation, let Σ be a simple domain description, let Ω be the conjunction of a pair of uniqueness-of-names axioms for the actions and fluents mentioned in Σ, and let Δ be a totally ordered narrative. For any fluent β and time point $\tau \leq$ n,

$$\text{CIRC}[\Sigma;\ \text{Initiates, Terminates, Releases}] \wedge$$
$$\text{CIRC}[\Delta;\ \text{Happens}] \wedge \Delta_0 \wedge \text{EC} \wedge \Omega \models \text{HoldsAt}(\beta,\tau)$$

if and only if,

$$\text{T}_I(\Delta_0) \wedge \bigwedge_{i=1}^{n} \text{T}_D(\Sigma, \text{i}) \wedge \bigwedge_{i=1}^{n} \text{T}_F(\Sigma, \text{i}) \wedge \text{CIRC}[\Delta;\text{Happens}] \wedge \Omega \models \text{HoldsAt}(\beta,\tau).$$

PROOF. First, we note that the event calculus formulation is inconsistent if and only if the CNF formulation is inconsistent. (In the event calculus formulation, inconsistency arises if and only if some fluent β is both inititated and terminated at the same time point τ. Definition 4.5 ensures that both HoldsAt(β,τ) and $\neg$ HoldsAt(β,τ) follow from the transformation if and only if this is the case.) So we can confine our attention to the consistent cases. The proof is by induction over the set of time points. First we show that the lemma holds for $\tau = 0$. Consider any ground fluent term β that holds at time 0 according to the event calculus formulation. Then, given the event calculus axioms, we must have Initially$_P$(β), in which case $\text{T}_I(\Delta_0) \models$ HoldsAt(β,0), so β also holds at time 0 according to the CNF formulation. An analogous argument holds for ground fluent terms that do not hold at time 0. The completeness of the initial situation ensures that this covers all ground fluent terms. Now suppose the lemma holds for $\tau = k$, where $k < n$. Then we can show that the lemma holds for $\tau = k + 1$ as follows. Consider any ground fluent term β. There are four cases to consider, given Lemma 5.1. Case 1: β doesn't hold at k but does hold at $k + 1$ according to the event calculus formulation. Given the event calculus axioms, it follows that we have $\Sigma \models$ Initiates(α, β, k) for some action α such that $\Delta \models$ Happens(α, k). In this case we also have, $\text{T}_D(\Sigma, k) \models$ Happens(α, k) $\rightarrow$ HoldsAt($\beta, k + 1$), from which it follows that β doesn't hold at k but does hold at $k + 1$ according to the CNF formulation. Case 2: β holds at k but doesn't hold at $k + 1$ according to the event calculus formulation. An analogous argument to that for the first case shows that β also holds at k but doesn't hold at $k + 1$ according to the CNF formulation. Case 3: β holds at both k and $k + 1$ according to the event calculus formulation. It follows that, for every action α such that $\Sigma \models$ Terminates(α,β, k), we have $\Delta \not\models$ Happens(α, k). But if $\Sigma \models$ Terminates(α,β, k), we also have $\text{T}_F(\Sigma$,i$) \models \neg$HoldsAt(β, k) $\vee$ HoldsAt($\beta, k + 1$) $\vee$ Happens(α, k). Since $\Delta \not\models$ Happens(α, k), it follows from this that β holds at both k and $k + 1$ according to the CNF formulation. Case 4: β doesn't hold at k and doesn't hold at $k + 1$ according to the event calculus formulation. An analogous argument to that for the third case shows that β doesn't hold at k and doesn't hold at $k + 1$ according to the CNF formulation. ∎

Next we prove the correspondence between models of the CNF encoding and abductively defined event calculus plans. This correspondence relies on the completeness of the initial situation and the fact that all actions in a simple domain description have deterministic effects. First we define some useful semantic concepts for propositional calculus.

DEFINITION 5.3
Let U be a set of propositional variables. A *truth assignment* for U is a function from U to the set $\{$True,False$\}$. Two truth assignments V_1 and V_2 *agree* if there is no propositional variable υ such that $\mathrm{V}_1(\upsilon) \neq \mathrm{V}_2(\upsilon)$.

DEFINITION 5.4
Let C be a CNF formula, and let U be the set of all propositional variables occurring in C. The formula C is *satisfiable given the truth assignment* V_1 if there exists a truth assignment V_2 for U that agrees with V_1 and which satisfies C.

Next we define a function that maps from event calculus narratives to propositional truth assignments.

DEFINITION 5.5
If C is a CNF formula and Δ is a totally ordered narrative, then the truth assignment $\mathrm{V}_{\Delta,C}$ is defined as follows. For every propositional variable of the form Happens(α,τ) that occurs in C,

$$\mathrm{V}_{\Delta,C}(\text{Happens}(\alpha,\tau)) = \begin{cases} \text{True if } \Delta \models \text{Happens}(\alpha,\tau) \\ \\ \text{False otherwise.} \end{cases}$$

The next lemma provides a bridge between satisfiability and entailment for encoded planning problems, in the context of simple domain descriptions and totally ordered narratives. It shows that an encoding plus a narrative is satisfiable if and only if the encoding minus the goal plus the narrative entails the goal.

LEMMA 5.6
For any complete initial situation Δ_0, any simple domain description Σ, any goal Γ, and any totally ordered narrative Δ of length n,[6] C = $\mathrm{T}(\Delta_0,\Sigma, \Gamma, n)$ is satisfiable given the truth assignment $\mathrm{V}_{\Delta,C}$ if and only if C† is consistent and C† $\models \mathrm{T}_G(\Gamma, n + 1)$ where C† is,

$$\bigwedge_{i=1}^{n} \mathrm{T}_D(\Sigma,\mathrm{i}) \wedge \bigwedge_{i=1}^{n} \mathrm{T}_F(\Sigma,\mathrm{i}) \wedge \text{CIRC}[\Delta;\text{Happens}] \wedge \mathrm{T}_I(\Delta_0) \wedge \Omega.$$

PROOF. Given the consistency of C†, the if half of the lemma follows directly. The only-if half can be shown as follows. Since the initial situation is complete, the form of $\mathrm{T}_D(\Sigma, i)$ and $\mathrm{T}_F(\Sigma, i)$ guarantees that, for any fluent β and any time point $\tau \leq n$, either C† $\models$ HoldsAt(β,τ) or C† $\models \neg$ HoldsAt(β,τ). (This can be shown by induction over time points. The details are omitted.) In other words, all models of C† satisfy exactly the same set of HoldsAt formulae. Now, note that any model V of C that agrees with the truth assignment $\mathrm{V}_{\Delta,C}$ is also a model of C†. Therefore, since V satisfies $\mathrm{T}_G(\Gamma, n + 1)$, we have C† $\models \mathrm{T}_G(\Gamma, n + 1)$. ∎

[6]The length of a narrative or plan is the number of actions it contains.

Finally, we have the main result.

THEOREM 5.7
For any complete initial situation Δ_0, any simple domain description Σ, and any goal Γ, Δ is a plan for Γ of length n if and only if C = T($\Delta_0, \Sigma, \Gamma, n$) is satisfiable given the truth assignment $V_{\Delta,C}$.

PROOF. The theorem follows directly from Lemmas 5.1, 5.2, and 5.6 ∎

Theorem 5.7 guarantees, for example, the soundness and completeness of any planner that CNF-encodes an event calculus planning problem using the transformation defined, and then submits it to a sound and complete SAT solver.

6 Concluding remarks

Although the focus of this work is logical foundations not implementation, a working prototype event calculus planner has been implemented based on the CNF encoding presented here, in combination with Kautz and Selman's (sound but not complete) WalkSat solver. In addition, the planner employs the action splitting technique described by Ernst *et al.* [3] to reduce the size of the encoding. The system was implemented in C and deployed on a 933 MHz Pentium III processor. The planner was run 100 times on the 9-block Blocks World problem `large_a` from the BlackBox test suite [12], and it found an optimal solution in an average time of around 9 seconds. This performance isn't competitive with recent high-speed domain-independent planners. But, as expected, it offers a significant improvement on traditional abductive event calculus planners, such as that presented in [23], which fail to solve Blocks World problems with five or more blocks in measurable time. The door is obviously open to using recent and future advances in satisfiability planning to obtain a better performance.

The implemented planner has been extended to cope with actions with conditional effects, although the formal basis for this is beyond the scope of the present paper. In future work, it is hoped to further extend the CNF encoding to encompass a richer class of event calculus planning problems, in a similar vein to [5]. For example, actions with non-deterministic effects can be handled by a form of generate-and-test algorithm, first generating plans that work for some outcome of the non-deterministic actions, and then testing to see whether they work for all possible outcomes. Mueller has already made considerable progress in this direction [18].

Acknowledgments

This work was supported by EPSRC grants GR/N13104 'Cognitive Robotics II' and GR/S02891/01 'Spatial Reasoning and Perception in a Humanoid Robot'. Thanks to Erik Mueller and the two anonymous referees.

References

[1] A.L. Blum and M.L. Furst. Fast planning through planning graph analysis, *Artificial Intelligence*, **90**, 281–300, 1997.

[2] M. Denecker, L. Missiaen, and M. Bruynooghe. Temporal reasoning with abductive event calculus, *Proceedings ECAI 92*, pp. 384–388, 1992.

[3] M. D. Ernst, T. D. Millstein, and D. S. Weld. Automatic SAT-compilation of planning problems, *Proceedings IJCAI 97*, pp. 1169–1176, 1997.

[4] K. Eshghi. Abductive planning with event calculus, *Proceedings of the Fifth International Conference on Logic Programming*, pp. 562–579, 1998.

[5] E. Giunchiglia. Planning as satisfiability with expressive action languages: concurrency, constraints and nondeterminism, *Proceedings KR 2000*, pp. 657–666, 2000.

[6] A. R. Haas. The case for domain-specific frame axioms, *Proceedings of the 1987 Workshop on the Frame Problem*, pp. 343–348, 1987.

[7] C. G. Jung, K. Fischer, and A. Burt. Multi-agent planning using an abductive event calculus, DFKI Report RR-96-04, DFKI, Germany, 1996.

[8] C. G. Jung. Situated abstraction planning by abductive temporal reasoning, *Proceedings ECAI 98*, pp. 383–387, 1998.

[9] A. C. Kakas, A. Michael, and C. Mourlas. ACLP: abductive constraint logic programming, *The Journal of Logic Programming*, **44**, 129–177, 2000.

[10] H. Kautz and B. Selman. Planning as satisfiability, *Proceedings ECAI 92*, pp. 359–363, 1992.

[11] H. Kautz and B. Selman. Pushing the envelope: planning, propositional logic and stochastic search, *Proceedings AAAI 96*, pp. 1194–1201, 1996.

[12] H. Kautz and B. Selman. Unifying SAT-based and graph-based planning, *Proceedings IJCAI 99*, pp. 318–325, 1999.

[13] F. Lévy and J. J. Quantz. Representing beliefs in a situated event calculus, *Proceedings ECAI 98*, pp. 547–551, 1998.

[14] V. Lifschitz. Circumscription. In *The 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. 297–352. Oxford University Press, 1994.

[15] R. S. Miller and M. P. Shanahan. Some alternative formulations of the event calculus. In *Computational Logic: Logic Programming and Beyond: Essays in Honour of Robert A. Kowalski*, pp. 452–490. Vol 2408 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, 2002.

[16] L. Missiaen, M. Bruynooghe and M. Denecker. CHICA, a planning system based on event calculus, *The Journal of Logic and Computation*, **5**, 579–602, 1995.

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

[18] E. T. Mueller. Event calculus reasoning through satisfiability, *The Journal of Logic and Computation*, **14**, 703–730, 2004.

[19] J. S. Penberthy and D. S. Weld. UCPOP: a sound, complete, partial order planner for ADL, *Proceedings KR 92*, pp. 103–114, 1992.

[20] M. P. Shanahan. Robotics and the common sense informatic situation, *Proceedings ECAI 96*, pp. 684–688, 1996.

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

[22] M. P. Shanahan. The ramification problem in the event calculus, *Proceedings IJCAI 99*, pp. 140–146, 1999.

[23] M. P. Shanahan. An abductive event calculus planner, *The Journal of Logic Programming*, **44**, 207–239, 2000.

Received 19 June 2003