Situation Calculus Specifications for Event Calculus Logic Programs

Rob Miller

Department of Computing,
Imperial College of Science, Technology and Medicine,
180 Queen's Gate, London SW7 2BZ, ENGLAND
email: rsm@doc. ic. ac. uk

WWW: http://laotzu.doc.ic.ac.uk/UserPages/staff/rsm/rsm.html

February 1995

To appear in:
Proceedings of the Third International Conference on Logic Programming and Non-monotonic Reasoning, June 26-28, 1995, Lexington, KY, USA.
pub. Springer Verlag

Abstract

A version of the Situation Calculus is presented which is able to deal with information about the actual occurrence of actions in time. Baker's solution to the frame problem using circumscription is adapted to enable default reasoning about action occurrences, as well as about the effects of actions. Two translations of Situation Calculus style theories into Event Calculus style logic programs are defined, and results are given on the soundness and completeness of the translations.

1 Introduction

This paper compares two formalisms and two associated default reasoning techniques for reasoning about action – the Situation Calculus [11], using a variant of Baker's circumscriptive solution to the frame problem [1], and the logic-programming based Event Calculus [8], in which default reasoning is realised through negation-as-failure. The version of the Situation Calculus used enables information about the occurrences of actions along a time line to be represented. A course of actions identified as actually occurring is referred to as a narrative, and this formalism is referred to as the Narrative Situation Calculus. Information about a narrative might be incomplete, so that default assumptions might be required. The circumscription policy incorporated in the Narrative Situation Calculus minimises action occurrences along the time-line. The original Event Calculus incorporates an analogous default assumption - that the only action occurrences are those provable from the theory.

The present paper shows that under certain circumstances the Narrative Situation Calculus may be regarded as a specification for Event Calculus style logic programs. The programs presented here are described as 'Event Calculus style' because of their use of $Initiates$ and $Terminates$ predicates to describe the effects of actions, because of the form of their persistence axioms, and because of the use of a time-line rather than the notion of a sequence or structure of situations. They differ from some other variants of the Event Calculus in that they do not assume complete knowledge of an initial state, and in that properties can hold (and persist) even if they have not been explicitly initiated by an action. Two classes of programs are discussed, both of which are "sound", for a wide class of domains, in that they only allow derivation of $Holds$ information which is semantically entailed by their circumscriptive specifications. Programs of the second type, although more complex, have an advantage over those of the first in that they are also "complete" even where information is missing about the state of affairs before any action occurs.

Notation: Many-sorted first order predicate calculus together with parallel and prioritized circumscription is used to describe the Narrative Situation Calculus. Variable names begin with a lower case letter. All variables in formulas are universally quantified with maximum scope unless otherwise indicated. To simplify descriptions of the implementations, logic programs are written in a subset of the same language, supplemented with the symbol *not* (negation-as-failure). Meta-variables are often written with Greek symbols, so that, for example, ζ might represent an arbitrary ground term of a particular sort. The parallel circumscription of predicates $\pi_1, \ldots, \pi_n$ in a theory T with $\xi_1, \ldots, \xi_k$ allowed to vary is written as

$$CIRC[T\ ;\ \pi_1, \ldots, \pi_n\ ;\ \xi_1, \ldots, \xi_k]$$

If $\rho_1, \ldots, \rho_m$ are also circumscribed, at a higher priority than $\pi_1, \ldots, \pi_n$, this is written as

$$\begin{gathered} CIRC[T\ ;\ \rho_1, \ldots, \rho_m\ ;\ \pi_1, \ldots, \pi_n, \xi_1, \ldots, \xi_k]\ \wedge \\ CIRC[T\ ;\ \pi_1, \ldots, \pi_n\ ;\ \xi_1, \ldots, \xi_k] \end{gathered}$$

Justification for this notation can be found, for example, in [9].

One other piece of notation for specifying uniqueness-of-names axioms will be useful. $UNA[\phi_1, \ldots, \phi_m]$ represents the set of axioms necessary to ensure inequality between different terms built up from the (possibly 0-ary) function symbols $\phi_1, \ldots, \phi_m$. It stands for the axioms

$$\phi_i(x_1, \ldots, x_k) \neq \phi_j(y_1, \ldots, y_n)$$

for $i < j$ where ϕ_i has arity k and ϕ_j has arity n, together with the following axiom for each ϕ_i of arity $k > 0$

$$\phi_i(x_1, \ldots, x_k) = \phi_i(y_1, \ldots, y_k) \rightarrow [x_1 = y_1, \ldots, x_k = y_k]$$

2 A Narrative Situation Calculus

In this section an overview is given of the Narrative Situation Calculus employed here as a specification language. This work is presented more fully in [13]. A class of many sorted first order languages is defined, and the types of sentence which can appear in particular domain descriptions are then described. Finally, the circumscription policy is discussed.

Definition 1 [Narrative domain language] A *Narrative domain language* is a first order language with equality of four sorts; a sort $\mathcal{A}$ of actions with sort variables $\{a, a_1, a_2, \ldots\}$, a sort $\mathcal{F}_g$ of generalised fluents[1] with sort variables $\{g, g_1, g_2, \ldots\}$, a sort $\mathcal{S}$ of situations with sort variables $\{s, s_1, s_2, \ldots\}$, and a sort $\mathcal{R}$ of time-points with sort variables $\{t, t_1, t_2, \ldots\}$. The sort $\mathcal{F}_g$ has three sub-sorts; the sub-sort $\mathcal{F}^+$ of positive fluents with sort variables $\{f^+, f_1^+, f_2^+, \ldots\}$, the sub-sort $\mathcal{F}^-$ of negative fluents with sort variables $\{f^-, f_1^-, f_2^-, \ldots\}$, and the sub-sort $\mathcal{F}_f$ of fluents with sort variables $\{f, f_1, f_2, \ldots\}$, such that

$$\mathcal{F}^+ \cap \mathcal{F}^- = \emptyset, \qquad \mathcal{F}^+ \cup \mathcal{F}^- = \mathcal{F}_f, \qquad \mathcal{F}_f \subset \mathcal{F}_g$$

It has time-point constant symbols corresponding to the real numbers, a finite number of action and positive fluent constants, a single situation constant $S0$, and no negative or generalised fluent constants. It has five functions:

$$\begin{array}{lll} Neg : \mathcal{F}^+ \mapsto \mathcal{F}^- & Sit : \mathcal{F}_g \mapsto \mathcal{S} & State : \mathcal{R} \mapsto \mathcal{S} \\ Result : \mathcal{A} \times \mathcal{S} \mapsto \mathcal{S} & And : \mathcal{F}_g \times \mathcal{F}_g \mapsto \mathcal{F}_g & \end{array}$$

and six predicates (other than equality): $Holds$ ranging over $\mathcal{F}_g \times \mathcal{S}$, Ab ranging over $\mathcal{A} \times \mathcal{F}_f \times \mathcal{S}$, $Absit$ ranging over $\mathcal{F}_g$, $<$ (infix) ranging over $\mathcal{R} \times \mathcal{R}$, $\leq$ (infix) ranging over $\mathcal{R} \times \mathcal{R}$, and $Happens$ ranging over $\mathcal{A} \times \mathcal{R}$. □

Only models are considered in which the predicates $<$ and $\leq$ are interpreted in the usual way as the "less-than" and "less-than-or-equal-to" relationships between real numbers. $Happens(\alpha, \tau)$ represents that an action α occurs at time τ[2], and $State(\tau)$ represents the situation at time τ.

Several domain independent axioms will always appear in Narrative Situation Calculus theories. The following five axioms are taken from [1].

$$Holds(And(g_1, g_2), s) \leftrightarrow [Holds(g_1, s) \wedge Holds(g_2, s)] \tag{B1}$$

$$Holds(Neg(f^+), s) \leftrightarrow \neg Holds(f^+, s) \tag{B2}$$

[1]Baker introduces generalised fluents in order to supply names to conjunctions of primitive fluents, so that for example the generalised fluent $And(Loaded, Neg(Alive))$ represents the joint property of being loaded and dead.

[2]Action occurrences are thus represented here as instantaneous. However the approach can easily be modified to represent actions with a duration (see [13]). The choice of the real numbers is also somewhat arbitrary – any ordered (or "non-converging" partially ordered) set of time-units would suffice (see [12] for further details).

$$Holds(g, Sit(g)) \leftarrow \neg Absit(g) \tag{B3}$$

$$Sit(g_1) = Sit(g_2) \rightarrow g_1 = g_2 \tag{B4}$$

$$[Holds(f, Result(a, s)) \leftrightarrow Holds(f, s)] \leftarrow \neg Ab(a, f, s) \tag{F1}$$

Axioms (B1)-(B4) are Baker's "existence-of-situations" axioms. Every combination of positive and negative fluents (negative fluent terms being those constructed with the Neg function) has at least one corresponding single "generalised fluent" term which can be constructed using the And function. Axioms (B1)-(B4), together with minimisation of $Absit$, ensure that, in each preferred model, for each consistent combination of fluents (characterised by some generalised fluent ϕ_g) there is at least one situation ($Sit(\phi_g)$) in which all of these fluents hold. Such situations are not characterised in the language by any actions which have led to them, but simply by the fluents that hold in them. Axiom (F1) is a frame axiom. Ab is minimised to represent the assumption that actions result only in changes demanded by the domain theory. Baker's minimisation policy avoids the "Yale Shooting Problem" by incorporating existence-of-situations axioms and by circumscribing Ab (at a lower priority than $Absit$) whilst allowing the $Result$ function to vary. Varying $Result$ ensures that for a given term $Result(\alpha, \sigma)$, the circumscription (and not the structure of the term) determines the set of possible situations to which it might refer. For any given model, inclusion in the language of terms of the form $Result(\alpha, \sigma)$ does not necessitate inclusion of extra situations in the universe of discourse, since there are no uniqueness-of-names axioms mentioning $Result$. Indeed, under the circumscription, the preferable models are generally those in which the denotation of each situation term of the form $Result(\alpha, \sigma)$ is the same as for some term of the form $Sit(\phi_g)$. (For further details consult [1].)

Two more domain independent axioms are included in the Narrative Situation Calculus, concerning properties of narratives and time-points:

$$State(t) = S0 \leftarrow \neg\exists a_1, t_1[Happens(a_1, t_1) \ \wedge \ t_1 < t] \tag{N1}$$

$$State(t) = Result(a_1, State(t_1)) \leftarrow \quad \text{(N2)}$$
$$[Happens(a_1, t_1) \ \wedge \ t_1 < t \ \wedge$$
$$\neg\exists a_2, t_2[Happens(a_2, t_2) \wedge [a_1 \neq a_2 \vee t_1 \neq t_2] \wedge t_1 \leq t_2 \wedge t_2 < t]]$$

Axiom (N1) relates all time points before the first action occurrence to the initial situation $S0$, and Axiom (N2) says that if action α_1 happens at τ_1, τ_1 is before τ and no other action happens between τ_1 and τ, then the situation at time τ is equal to $Result(\alpha_1, State(\tau_1))$.

Several types of axioms are either required or allowed in Narrative Situation Calculus theories[3]. The following definitions specify the form of such sentences.

Definition 2 [Initial conditions description] A formula is an *initial conditions description* if it is of the form

$$Holds(\phi, S0) \qquad \text{or} \qquad Holds(Neg(\phi), S0)$$

where ϕ is a positive fluent constant. □

Definition 3 [Action description] A formula is an *action description* if it is of one of the following two forms

$$Holds(\phi, Result(\alpha, s))$$

$$Holds(\phi, Result(\alpha, s)) \leftarrow [Holds(\phi_1, s) \wedge \ldots \wedge Holds(\phi_n, s)]$$

where α is an action constant, $\phi, \phi_1, \ldots, \phi_n$ are (positive or negative) ground fluent terms, and for each i and j, $1 \leq i, j \leq n$, $\phi_i \neq Neg(\phi_j)$. □

Definition 4 [Narrative domain description] Given a narrative domain language with positive fluent constants $\phi_1, \ldots, \phi_n$ and with action constants $\alpha_1, \ldots, \alpha_m$, a formula N is a *narrative domain description* if it is a conjunction of action descriptions, initial conditions descriptions, occurrence descriptions, the frame axiom (F1), existence-of-situations axioms (B1)-(B4), axioms (N1) and (N2), uniqueness-of-names axioms

[3] In this paper theories do not include domain constraints, since no translation of domain constraints into logic programs will be given. The Narrative Situation Calculus in [13] includes such constraints.

$$UNA[\phi_1, \ldots, \phi_n, And, Neg] \qquad UNA[\alpha_1, \ldots, \alpha_m]$$

and a domain closure axiom for fluents

$$f = \phi_1 \vee \ldots \vee f = \phi_n \vee f = Neg(\phi_1) \vee \ldots \vee f = Neg(\phi_n)$$

□

Although domain constraints have not been explicitly included in narrative domain descriptions, care must be taken, since domain constraints might be derived from pairs of action descriptions, together with Axiom (B2). For example, from the two action descriptions

$$Holds(F_1, Result(A, s)) \leftarrow Holds(F_2, s)$$

$$Holds(Neg(F_1), Result(A, s)) \leftarrow Holds(F_3, s)$$

the sentence $\neg[Holds(F_2, s) \wedge Holds(F_3, s)]$ can be derived. In fact, domain constraints are entailed only from pairs of action descriptions of this form (see [12]). Hence the following definition is included, of narrative domain descriptions with no implicit domain constraints.

Definition 5 [Fluent independence] A narrative domain description N is *fluent independent* if for every pair of action descriptions in N of the form

$$Holds(\phi, Result(\alpha, s)) \leftarrow [Holds(\phi_1, s) \wedge \ldots \wedge Holds(\phi_m, s)]$$

$$Holds(Neg(\phi), Result(\alpha, s)) \leftarrow [Holds(\phi_{m+1}, s) \wedge \ldots \wedge Holds(\phi_n, s)]$$

there is some i, $1 \leq i \leq m$, and some j, $m+1 \leq j \leq n$, such that $\phi_i = Neg(\phi_j)$ or $\phi_j = Neg(\phi_i)$. □

The following definition is also useful. It identifies narrative domain descriptions in which information about what holds in the initial situation is complete.

Definition 6 [Initially specified narrative domain description] A narrative domain description N is *initially specified* if for every positive fluent constant ϕ in the language, either $Holds(\phi, S0)$ or $Holds(Neg(\phi), S0)$ is an initial conditions description in N. □

Two examples of fluent independent narrative domain descriptions are given below. The first is initially specified and the second is not. Only the domain-dependent axioms are given. Example 1 is a version of the Yale Shooting Problem, including a simple narrative in which a $Sneeze$[4] action occurs at time 1 followed by a $Shoot$ action at time 3. The full narrative domain description is referred to as N_{YSP}.

Example 1 [The Yale Shooting Problem, N_{YSP}] Fluent constants: $Alive$, $Loaded$, Action constants: $Sneeze$, $Shoot$, Domain-specific axioms:

$$Holds(Neg(Alive), Result(Shoot, s)) \leftarrow Holds(Loaded, s) \tag{Y1}$$

$$Holds(Loaded, S0) \tag{Y2}$$

$$Holds(Alive, S0) \tag{Y3}$$

$$UNA[Alive, Loaded, And, Neg] \tag{Y4}$$

$$UNA[Sneeze, Shoot] \tag{Y5}$$

$$f = Alive \vee f = Loaded \vee f = Neg(Alive) \vee f = Neg(Loaded) \tag{Y6}$$

$$Happens(Sneeze, 1) \tag{Y7}$$

$$Happens(Shoot, 3) \tag{Y8}$$

□

Because it is not initially specified, Example 2 below is useful in illustrating that logic program translations cannot be used to derive $Holds$ literals not warranted by their specifications (i.e. that they are "sound"). It concerns an electric gate, connected to a button which will open the gate when pressed, provided the system is connected to an electric supply. The gate is initially closed. There is no information as to whether the system is initially

[4]The $Sneeze$ action takes place of the $Wait$ action in the original formulation. In the Narrative Situation Calculus, "waits" are more naturally represented by the absence of an action occurrence within some time interval.

connected to an electrical supply, hence it is not possible to deduce that the gate will be either open or closed after the button has been pressed. The full narrative domain description is referred to as N_{GATE}.

Example 2 $[N_{GATE}]$ Fluent constants: *Open*, *Connected*, Action constant: *Press*, Domain-specific axioms:

$$Holds(Open, Result(Press, s)) \leftarrow Holds(Connected, s) \tag{G1}$$

$$Holds(Neg(Open), S0) \tag{G2}$$

$$UNA[Open, Connected, And, Neg] \tag{G3}$$

$$\begin{aligned} f = Open \lor f = Connected \lor f = Neg(Open) \\ \lor f = Neg(Connected) \end{aligned} \tag{G4}$$

$$Happens(Press, 1) \tag{G5}$$

$\square$

For a given domain description D, Baker's original circumscription policy is

$$CIRC[D \; ; \; Absit \; ; \; Ab, Result, Holds, S0] \; \wedge \; CIRC[D \; ; \; Ab \; ; \; Result, S0]$$

The above policy will be referred to as $CIRC_b$. As regards the Yale Shooting Problem, Baker shows that

$$CIRC_b[N_{YSP}] \models Holds(Neg(Alive), Result(Shoot, Result(Sneeze, S0)))$$

The Narrative Situation Calculus introduces an extended circumscription policy representing the assumption that the only action occurrences are those explicitly described in the narrative description. The separation of sentences in theories into those which describe actions' effects and those which refer to the narrative allows this to be achieved in a natural way simply by circumscribing *Happens* in parallel with *Ab*, while varying *State* along with *Result*. As before, *Absit* is circumscribed at a higher priority so as to ensure the existence of all consistent situations. Thus, given a narrative domain description N, the extended circumscription policy is

$$CIRC[N\ ;\ Absit\ ;\ Ab, Happens, Result, Holds, S0, State]\ \wedge$$
$$CIRC[N\ ;\ Ab, Happens\ ;\ Result, S0, State]$$

This circumscription policy is referred to as $CIRC_n$.

Three theorems are useful at this point. Full proofs of these can be found in [12]. The first two theorems show that minimisation of *Happens* does not interfere with the minimisation used to solve the frame problem. The third theorem shows that, unsurprisingly, circumscribing *Happens* has the same effect as forming its completion.

Theorem 1 Let N be a narrative domain description. Then
$CIRC_n[N] \models CIRC_b[N]$ □

Theorem 2 Let N be a narrative domain description, and let Φ be a sentence which does not contain the predicate symbol *Happens* and does not contain the function symbol *State*. Then $CIRC_n[N] \models \Phi$ if and only if $CIRC_b[N] \models \Phi$ □

Theorem 3 Let N be a narrative domain description with k occurrence descriptions. If $k < 0$ and the set of occurrence descriptions is
$\{Happens(\alpha_i, \tau_i) \mid 1 \leq i \leq k\}$ then

$$CIRC_n[N] \models Happens(a, t) \leftrightarrow \bigvee_{i=1}^{k} [a = \alpha_i \wedge t = \tau_i]$$

and if $k = 0$ then $CIRC_n[N] \models \neg \exists a, t[Happens(a, t)]$ □

These theorems together with Axioms (N1) and (N2) allow the deduction of what fluents hold at different time points, i.e. they facilitate temporal projection. For example, in the Yale Shooting Problem it can be shown that

$$CIRC_n[N_{YSP}] \models Holds(Neg(Alive), State(5))$$

A derivation of this is given in [12].

3 A Translation into Logic Programs

For the purposes of deriving information about what holds along the time-line, the narrative domain descriptions of the previous section can be translated into Event Calculus style logic programs which do not contain situation terms or arguments. In this section, logic programs are defined which use the following predicate symbols: $<$, $HoldsAt$, $Initially$, $Initiates$, $Terminates$, $ClippedBetween$, $ClippedBefore$, $Happens$, and $\leq$[5]. Given a narrative domain description N, the aim is to define a logic program $EC[N]$ which facilitates temporal projection and which is "sound" in the following sense; for any ground fluent term ϕ and real number τ, the positive literal $HoldsAt(\phi, \tau)$ is derivable from $EC[N]$ only if $CIRC_n[N] \models Holds(\phi, State(\tau))$. The following definition of *fluent converses* will be useful.

Definition 7 [Converse of a fluent] Let ϕ be a ground fluent term. Then the *converse* of ϕ, written $\overline{\phi}$, is $Neg(\phi)$ if ϕ is a positive fluent constant, or ϕ' if ϕ is of the form $Neg(\phi')$ for some positive fluent constant ϕ'. $\Box$

Occurrence descriptions are included directly in logic programs as conditionless clauses. The following definitions show how initial conditions descriptions and action descriptions are translated into domain-specific $Initially$, $Initiates$ and $Terminates$ clauses.

Definition 8 [$IN[IC]$] Let ϕ be a ground fluent term and let IC be an initial conditions description of the form $Holds(\phi, S0)$. The program clause $IN[IC]$ is defined as $Initially(\phi)$. $\Box$

Definition 9 [$INIT[AD]$ and $TERM[AD]$] Let $\phi, \phi_1, \ldots, \phi_n$ be ground fluent terms, α an action constant, s a situation variable and t a time-point variable. Let AD be an action description of the form

$$Holds(\phi, Result(\alpha, s)) \leftarrow [Holds(\phi_1, s) \wedge \ldots \wedge Holds(\phi_n, s)]$$

The program clause $INIT[AD]$ is defined as

$$Initiates(\alpha, \phi, t) \leftarrow [HoldsAt(\phi_1, t) \wedge \ldots \wedge HoldsAt(\phi_n, t)]$$

[5] It is assumed that $\leq$ and $<$ are defined in such a way that for all $\tau_1, \tau_2 \in \mathcal{R}$, goals of the form "$\tau_1 \leq \tau_2$" and "$\tau_1 < \tau_2$" succeed if and only if $\tau_1 \leq \tau_2$ and $\tau_1 < \tau_2$ respectively.

and the program clause $TERM[AD]$ is defined as

$$Terminates(\alpha, \overline{\phi}, t) \leftarrow [not\ HoldsAt(\overline{\phi_1}, t) \wedge \ldots \wedge not\ HoldsAt(\overline{\phi_n}, t)]$$

□

In the definition above, although $Terminates$ clauses have a similar structure to $Initiates$ clauses, their bodies incorporate negated (not) literals with fluent converses as their first arguments. As will be seen from the examples below, this difference becomes important for domains where information about the initial situation is incomplete. Whereas $Initiates$ clauses describe the immediate effects of actions, $Terminates$ clauses play the role of domain-specific "abnormality" clauses which help determine those fluents which do not persist through an action occurrence. Where information about what holds at a time-point τ is incomplete, the sub-goal $not\ HoldsAt(\overline{\phi_i}, t)$ might succeed even when the sub-goal $HoldsAt(\phi_i, t)$ fails. Were it not for this use of negative literals, the definition of $Terminates$ would in general be incomplete, so that the default persistence of fluents would in general be incorrect. Similar techniques are used for example in [4] and [2]. (In fact, in the special case where domains are initially specified these domain-specific $Terminates$ clauses may be replaced by a general definition, expressing that an action terminates a fluent if and only if it initiates its converse.) The following definition gives a complete translation of a narrative domain description into a logic program.

Definition 10 $[EC[N]]$ Given a narrative domain description N with action descriptions $AD_1, \ldots, AD_n$, initial condition descriptions $ID_1, \ldots, ID_m$ and occurrence descriptions $OD_1, \ldots, OD_k$ then the logic program $EC[N]$ is defined as

$$\{INIT[AD_1], \ldots, INIT[AD_n], TERM[AD_1], \ldots, TERM[AD_n], \\ IN[ID_1], \ldots, IN[ID_m], OD_1, \ldots, OD_k\}$$

together with the following domain-independent clauses

$$HoldsAt(f, t) \leftarrow [Initially(f) \wedge not\ ClippedBefore(f, t)] \qquad \text{(EC1)}$$

$$HoldsAt(f, t3) \leftarrow [Happens(a, t1) \wedge t1 < t3 \wedge Initiates(a, f, t1) \wedge not\ ClippedBetween(t1, f, t3)] \quad \text{(EC2)}$$

$$ClippedBefore(f, t) \leftarrow [Happens(a, t1) \wedge t1 < t \wedge Terminates(a, f, t1)] \quad \text{(EC3)}$$

$$ClippedBetween(t1, f, t3) \leftarrow [Happens(a, t2) \wedge t1 \leq t2 \wedge t2 < t3 \wedge Terminates(a, f, t2)] \quad \text{(EC4)}$$

□

Clauses (EC1) and (EC2) above are persistence axioms. Clause (EC1) states that a fluent ϕ holds at a time τ if it is initially true and it has not been cancelled ("clipped") before τ by some action. Clause (EC2) states that a fluent holds at a time τ_3 if it is initiated by an action occurrence at some previous time τ_1, and is not clipped in the meantime. Clauses (EC3) and (EC4) give the definitions for $ClippedBefore$ and $ClippedBetween$ in terms of $Happens$ and $Terminates$. Notice that there is no direct representation of Axiom (B2) in EC programs. No clause such as $HoldsAt(Neg(f), t) \leftarrow not\ HoldsAt(f, t)$ is included, since this would clearly cause unsoundness in cases where narrative domain descriptions are not initially specified (such as Example 2). In the method presented here, a distinction should be made between $EC[N] \not\vdash_{SLDNF} HoldsAt(\phi, \tau)$ which should be interpreted as "it is not provable that ϕ holds at time τ", and $EC[N] \vdash_{SLDNF} HoldsAt(Neg(\phi), \tau)$ which should be interpreted as "ϕ does not hold at time τ".

As regards the Yale Shooting Problem (Example 1), the domain dependent clauses in $EC[N_{YSP}]$ are

$$Initiates(Shoot, Neg(Alive), t) \leftarrow HoldsAt(Loaded, t) \quad (INIT[\text{(Y1)}])$$

$$Terminates(Shoot, Alive, t) \leftarrow not\ HoldsAt(Neg(Loaded), t) \quad (TERM[\text{(Y1)}])$$

$$Initially(Loaded) \quad (IN[\text{(Y2)}])$$

$$Initially(Alive) \quad (IN[\text{(Y3)}])$$

$$Happens(Sneeze, 1) \tag{Y7}$$

$$Happens(Shoot, 3) \tag{Y8}$$

so that $EC[N_{YSP}] \vdash_{SLDNF} HoldsAt(Neg(Alive), 5)$.

In Example 2 (of the electric gate), the initial situation was not fully specified. $EC[N_{GATE}]$ consists of the clauses (EC1)-(EC4) together with the clauses

$$Initiates(Press, Open, t) \leftarrow HoldsAt(Connected, t) \qquad (INIT[(\mathrm{G1})])$$

$$Terminates(Press, Neg(Open), t) \leftarrow \\ \quad not\ HoldsAt(Neg(Connected), t) \qquad (TERM[(\mathrm{G1})])$$

$$Initially(Neg(Open)) \qquad (IN[(\mathrm{G2})])$$

$$Happens(Press, 1) \tag{G5}$$

Example 2 illustrates the need for the use of negation-as-failure in the bodies of $Terminates$ clauses. Had $TERM[(\mathrm{G1})]$ simply been

$$Terminates(Press, Neg(Open), t) \leftarrow HoldsAt(Connected, t)$$

then the query $HoldsAt(Neg(Open), 2)$ would succeed, so that the program would be "unsound" in the sense described above. In fact, Theorems 4 and 5 below show that the translation method given is unsound only under two circumstances – (i) if the narrative domain description is not fluent independent, and (ii) if two or more different actions in the narrative occur simultaneously. If there are no simultaneous action occurrences and the narrative domain description is both fluent independent and initially specified, then the translations are both sound and complete.

Theorem 4 Let N be a fluent independent narrative domain description, let τ be a real number, and let ϕ be a ground fluent term. Suppose that for all $\tau' \leq \tau$ there is at most one occurrence description in N of the form $Happens(\alpha, \tau')$. Then

$$CIRC_n[N] \models Holds(\phi, State(t)) \quad \text{if} \quad EC[N] \vdash_{SLDNF} HoldsAt(\phi, t)$$

□

Theorem 5 Let N be an initially specified fluent independent narrative domain description, let τ be a real number, and let ϕ be a ground fluent term. Suppose that for all $\tau' \leq \tau$ there is at most one occurrence description in N of the form $Happens(\alpha, \tau')$. Then

$$CIRC_n[N] \models Holds(\phi, State(t)) \quad \text{if and only if} \\ EC[N] \vdash_{SLDNF} HoldsAt(\phi, t)$$

□

Space limitations do not permit full proofs to be given here; these can be found in [12]. However, the following remarks summarise the arguments. Theorems 4 and 5 rely on several intermediate propositions. In the proofs, an intermediate translation of circumscribed theories into Situation Calculus style logic programs is given, and soundness and/or completeness is shown for these programs. They are of independent interest because equivalents of Theorems 4 and 5 above hold for these programs even where simultaneous actions occur. To show this, several properties of the circumscriptions are first proved. In particular, it is shown that the circumscriptive formulation is, in Lin and Shoham's terms, epistemologically complete [10]. That is, given any situation in which all fluent values are known, then all fluent values will also be known in the situation resulting from a single action. Propositions are also proved which express limits to the possible extension of the predicate Ab, even in the general case where theories are not initially specified, in terms of syntactic properties of the theories. These results are used to show that, in this general case, it is possible to partially compute what holds in any situation named by sequence of actions by consideration of each action in turn, and that the nesting of the *not* operator in the Situation Calculus programs (similar to the nesting of *not* in the Event Calculus programs resulting from the $Terminates$ literals in the bodies of clauses (EC3) and (EC4)) provides precisely the strength of default persistence required.

The intermediate logic programs which are of Situation Calculus style contain a frame axiom which is a clausal counterpart to Axiom (F1). They also contain domain-specific Ab clauses analogous to the $Terminates$ clauses in the Event Calculus style programs. The essential difference between the two types of program is as follows. In the Situation Calculus implementations, persistence of properties along the time-line is derived indirectly by proving persistence one step at a time through a series of situations, which is in turn shown to correspond to the narrative in question. In the Event Calculus programs, default persistence of properties through a narrative is expressed more directly (clauses (EC1)–(EC4)). To show the correspondence between the Event Calculus style and Situation Calculus style programs (in the case where no simultaneous actions occur within the narrative), it is necessary to show that these two expressions of persistence are equivalent, and this can be done using induction on the number of occurrence descriptions within the narrative.

Where domains are not initially specified, program translations are "incomplete" because, in the specifications, Axiom (B2) allows the derivation of disjunctions of ground $Holds$ literals. To take a trivial example, given a language including fluent constants F_1 and F_2 and action constant A, it is possible to derive $Holds(F_1, S0) \vee Holds(Neg(F_1), S0)$, so that the action descriptions and occurence description

$$Holds(F_2, Result(A, S0)) \leftarrow Holds(F_1, S0)$$

$$Holds(F_2, Result(A, S0)) \leftarrow Holds(Neg(F_1), S0)$$

$$Happens(A, 1)$$

entail $Holds(F_2, State(2))$. The corresponding "EC" logic program translation will not yield an SLDNF derivation of $HoldsAt(F_2, 2)$, since programs do not include a direct representation of Axiom (B2). However, since domains include only a finite number of fluents, it is not hard to design meta-level procedures or program enhancements which provide completeness in the more general case by testing "EC" programs with each possible initial situation consistent with the information in the specification. A more detailed discussion of this can be found in [12], but for the purposes of stating an appropriate theorem, a simple definition of such a program enhancement is given below:

Definition 11 $[EC^+[N]]$ Let N be a narrative domain description written in a narrative domain language with positive fluent constants $\phi_1, \ldots, \phi_n$. Let l be a variable not used in $EC[N]$. Then the program $EC^+[N]$ is defined as $EC[N]$ with the variable l added as an extra first argument in all occurrences of $HoldsAt$, $Initiates$, $Terminates$, $ClippedBetween$, $ClippedBefore$ and $Initially$ literals, together with the following clauses:

$Initially(l, f) \leftarrow On(f, l)$

$Fluent(\phi_1) \ \ldots \ Fluent(\phi_n)$

$MholdsAt(f, t) \leftarrow Forall(PossibleFluentCombination(l), HoldsAt(l, f, t))$

$PossibleFluentCombination(l) \leftarrow$
$\quad [Setof(f, (Fluent(f) \wedge \mathit{not}\ Initially([], f) \wedge \mathit{not}\ Initially([], Neg(f))), p)$
$\quad \wedge\ Permutation(p, l)]$

$Permutation([], [])$

$Permutation([f|r1], [f|r2]) \leftarrow Permutation(r1, r2)$

$Permutation([f|r1], [Neg(f)|r2]) \leftarrow Permutation(r1, r2)$

□

The following theorem, the proof of which is given in [12], states the sense in which EC^+ programs[6] are sound and complete.

Theorem 6 Let N be a fluent independent narrative domain description, let τ be a real number, and let ϕ be a ground fluent term. Suppose that for all $\tau' \leq \tau$ there is at most one occurrence description in N of the form $Happens(\alpha, \tau')$. Then

$$CIRC_n[N] \models Holds(\phi, State(t)) \quad \text{if and only if} \quad EC^+[N] \vdash_{SLDNF} MholdsAt(\phi, t)$$

□

[6]Strictly speaking, it is not possible to use SLDNF with such programs as they contain calls to the "second order" primitives $Forall$ and $Setof$ included here for readability. It is assumed that these primitives are appropriately re-interpreted. For definitions and practical details the reader may consult [16].

4 Related Work

This paper contributes to a growing body of research into the correspondence between different mechanisms for reasoning about actions, and on the use of logic programming in this respect. A recent result of Kowalski and Sadri [7] is closely related to the topic of this paper. This shows that, in the context of a class of logic programs for reasoning about actions and narratives, then all other aspects of programs being equal, the Situation Calculus type and Event Calculus type persistence clauses are interchangeable. Pinto and Reiter [14] also show how reasoning about a narrative may be accomplished with Situation Calculus style logic programs. The present paper addresses Pinto and Reiter's criticism of the original Event Calculus – that it "does not characterize a class of sound programs". Both Kowalski and Sadri and Pinto and Reiter use the Clark completion to give a "semantics" to programs, which can be regarded as their "specification" in the sense used here. Shanahan [15] shows how a circumscription policy related to Baker's may be used with Event Calculus style first order theories to model default reasoning.

As regards the non-narrative aspects of reasoning about action, various results enable a network of correspondences between formalisms to be built up. For example, Kartha [6] shows a correspondence between Baker's formalism and the Language $\mathcal{A}$ introduced in [5]. Furthermore, Gelfond and Lifschitz [5], Dung [4], Baral and Gelfond [2], and Denecker and De Schreye [3] have each shown how the Language $\mathcal{A}$ can be used as a specification for various logic programming formulations. Like the work in this paper, most results are restricted to cases where theories are "fluent independent".

References

[1] A. B. Baker, *Nonmonotonic Reasoning in the Framework of the Situation Calculus*, Artificial Intelligence, vol 49, page 5, 1991.

[2] Chitta Baral and Michael Gelfond, *Representing Concurrent Actions in Extended Logic Programming*, Proceedings IJCAI 1993, Morgan Kaufmann, page 866, 1993.

[3] Marc Denecker and Danny De Schreye, *Representing Incomplete Knowledge in Abductive Logic Programming*, Proceedings of the International Symposium on Logic Programming, 1993.

[4] Phan Minh Dung, *Representing Actions in Logic Programming and its Applications in Database Updates*, Proceedings of the Tenth International Conference on Logic Programming, ed David S. Warren, MIT Press, pages 222-238, 1993.

[5] Michael Gelfond and Vladimir Lifschitz, *Representing Actions in Extended Logic Programming*, Proceedings of the Joint International Conference and Symposium on Logic Programming, ed. Krzysztof Apt, MIT Press, page 560, 1992.

[6] G. Neelakantan Kartha, *Soundness and Completeness Theorems for Three Formalizations of Action*, Proceedings IJCAI 1993, page 724, 1993.

[7] R. A. Kowalski and F. Sadri, *The Situation Calculus and Event Calculus Compared*, Proceedings of the International Logic Programming Symposium, 1994

[8] R. A. Kowalski and M. J. Sergot, *A Logic-Based Calculus of Events*, New Generation Computing, vol 4, page 267, 1986.

[9] V. Lifschitz, *Circumscription*, in Handbook of Logic in A.I., ed.s D. Gabbay *et al.*, OUP, pages 297–352, 1995

[10] Fangzhen Lin and Yoav Shoham, *Provably Correct Theories of Action*, Proceedings AAAI 1991, MIT Press, page 349, 1991.

[11] J. McCarthy and P. J. Hayes, *Some Philosophical Problems from the Standpoint of Artificial Intelligence*, in Machine Intelligence 4, ed.s D. Michie and B. Meltzer, Edinburgh University Press, 1969.

[12] R. S. Miller, *Narratives in the Context of Temporal Reasoning*, Imperial College Research Report DoC 94/3, available from Department of Computing, Imperial College, 180 Queen's Gate, London SW7 2BZ, 1994.

[13] R. S. Miller and M. P. Shanahan, *Narratives in the Situation Calculus*, in Journal of Logic and Computation, Special Issue on Actions and Processes, vol 4 no 5, Oxford University Press, 1994.

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

[15] M. P. Shanahan, *A Circumscriptive Calculus of Events*, to appear in Artificial Intelligence, 1994.

[16] Leon Sterling and Ehud Shapiro, *The Art of Prolog*, MIT Press, 1986.