Situation Calculus Specifications for Event Calculus Logic Programs

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

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. A translation of Situation Calculus style theories into Event Calculus style logic programs is defined, and results are given on its soundness and completeness.

1. Introduction

This paper compares two formalisms and two associated default reasoning techniques for reasoning about action — the Situation Calculus using a variant of Baker's circumscriptive solution to the frame problem [Baker, 1991], and the logic-programming based Event Calculus [Kowalski & Sergot, 1986], 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. The programs described 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. Where total information is available about the initial state of affairs, the programs are also "complete" in this same sense.

Many-sorted first order predicate calculus together with parallel and prioritised 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 italicised, so that, for example, "*F*" might represent an arbitrary ground term of a particular sort. The parallel circumscription of predicates $P_1,...,P_n$ in a sentence T with $V_1,...,V_k$ allowed to vary is written as

$$\mathrm{CIRC}[T\ ;\ P_1,...,P_n\ ;\ V_1,...,V_k]$$

If $R_1,...,R_m$ are also circumscribed, at a higher priority than $P_1,...,P_n$, this is written as

$$\mathrm{CIRC}[T\ ;\ R_1,...,R_m\ ;\ P_1,...,P_n,V_1,...,V_k] \\ \wedge\ \mathrm{CIRC}[T\ ;\ P_1,...,P_n\ ;\ V_1,...,V_k]$$

Justification for this notation can be found, for example, in [Lifschitz, 1995]. One other piece of notation for specifying uniqueness-of-names axioms will be useful. $\mathrm{UNA}[F_1,..,F_m]$ represents the set of axioms necessary to ensure inequality between different terms built up from the (possibly 0-ary) function symbols $F_1,..,F_m$. It stands for the axioms

$$F_i(x_1,......,x_k) \neq F_j(y_1,......,y_n)$$

for i<j where F_i has arity k and F_j has arity n, together with the following axiom for each F_i of arity k>0.

$$F_i(x_1,......,x_k)=F_i(y_1,......,y_k) \rightarrow [x_1=y_1 \wedge......\wedge x_k=y_k]$$

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 [Miller & Shanahan, 1994]. 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 {Narrative domain language}. A *Narrative domain language* is a first order language with equality of four sorts; a sort α of actions with sort variables {a,a1,a2,...}, a sort ϕ_g of generalised fluents[1] with sort variables {g,g1,g2,...}, a sort σ of situations with sort variables {s,s1,s2,...}, and a sort $\mathbb{R}$ of time-points with sort variables {t, t1, t2,......}. The sort ϕ_g has three sub-sorts; the sub-sort ϕ_+ of positive fluents with sort variables $\{f_+, f_+^1, f_+^2, ...\}$, the sub-sort ϕ_- of negative fluents with sort variables $\{f_-, f_-^1, f_-^2, ...\}$, and the sub-sort ϕ_f of fluents with sort variables {f,f1,f2,...}, such that

$$\phi_+ \cap \phi_- = \emptyset \qquad \phi_+ \cup \phi_- = \phi_f \qquad \phi_f \subset \phi_g$$

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: Result: $\alpha \times \sigma \rightarrow \sigma$, And: $\phi_g \times \phi_g \rightarrow \phi_g$, Neg: $\phi_+ \rightarrow \phi_-$, Sit: $\phi_g \rightarrow \sigma$, and State: $\mathbb{R} \rightarrow \sigma$, and six predicates (other than equality): Holds ranging over $\phi_g \times \sigma$, Ab ranging over $\alpha \times \phi_f \times \sigma$, Absit ranging over ϕ_g, < (infix) ranging over $\mathbb{R} \times \mathbb{R}$, $\leq$ (infix) ranging over $\mathbb{R} \times \mathbb{R}$, and Happens ranging over $\alpha \times \mathbb{R}$. □

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(*A*,*T*)" represents that an action *A* occurs at time *T*[2], and "State(*T*)" represents the situation at time *T*.

Several domain independent axioms will always appear in Narrative Situation Calculus theories. The following five axioms are taken from [Baker, 1991].

$$\text{Holds(And(g1,g2),s)} \equiv [\text{Holds(g1,s)} \wedge \text{Holds(g2,s)}] \quad \text{(B1)}$$

$$\text{Holds(Neg}(f_+),s) \equiv \neg\text{Holds}(f_+,s) \quad \text{(B2)}$$

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

$$\text{Sit(g1)=Sit(g2)} \rightarrow \text{g1=g2} \quad \text{(B4)}$$

$$[\text{Holds(f,Result(a,s))} \equiv \text{Holds(f,s)}] \leftarrow \neg\text{Ab(a,f,s)} \quad \text{(F1)}$$

Axioms (B1)–(B4) are Baker's "existence-of-situations" axioms. Along with the minimisation of Absit, these ensure that all consistent combinations of fluents are accounted for in the overall theory. To solve the frame problem, Ab is also minimised (at a lower priority) allowing Result to vary. For details the reader should consult [Baker, 1991] or [Miller, 1994].

[1]Generalised fluents supply names to conjunctions of primitive fluents. For example the generalised fluent "And(Loaded,Neg(Alive))" represents the joint property of being loaded and dead.

[2]This approach is modified to represent (possibly overlapping) actions with a duration in [Miller & Shanhan, 1994].

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

$$\text{State(t)=S0} \leftarrow \neg\exists \text{a1,t1[Happens(a1,t1)} \wedge \text{t1<t]} \quad \text{(N1)}$$

$$\begin{aligned}\text{State(t)=Result(a1,State(t1))} &\leftarrow \quad \text{(N2)}\\ &[\text{Happens(a1,t1)} \wedge \text{t1<t} \wedge\\ &\neg\exists\text{a2,t2[Happens(a2,t2)} \wedge [\text{a1}\neq\text{a2} \vee \text{t1}\neq\text{t2}]\\ &\quad \wedge \text{t1} \leq \text{t2} \wedge \text{t2} < \text{t}]]\end{aligned}$$

Axiom (N1) relates all time points before the first action occurrence to the initial situation S0, and Axiom (N2) says that if action *A1* happens at *T1*, *T1* is before *T* and no other action happens between *T1* and *T*, then the situation at is equal to Result(*A1*,State(*T1*)).

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 {Initial conditions description}. A formula is an *initial conditions description* if it is of the form

Holds(*F*,S0) or Holds(Neg(*F*),S0)

where *F* is a positive fluent constant. □

Definition {Action description}. A formula is an *action description* if it is of the form

Holds(*F*,Result(*A*,s))
or Holds(Neg(*F*),Result(*A*,s))
or Holds(*F*,Result(*A*,s)) ← [Holds(F_1,s) ∧ ... ∧ Holds(F_n,s)]
or Holds(Neg(*F*),Result(*A*,s)) ← [Holds(F_1,s) ∧ ... ∧ Holds(F_n,s)]

where *A* is an action constant, *F*, F_1,..,F_n are ground fluent terms, and for each i and j, $1 \leq i,j \leq n$, $F_i \neq \text{Neg}(F_j)$. □

Definition {Occurrence description}. A formula is an *occurrence description* if it is of the form

Happens(*A*,*T*)

where *A* is an action constant and *T* is a real number. □

Definition {Narrative domain description}. Given a narrative domain language with positive fluent constants F_1,....,F_n and action constants A_1,....,A_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-

[3]Unlike in [Miller & Shanhan, 1994], domain constraints between fluents are not considered here, since no translation of these into logic programs will be given.

of-situations axioms (B1)–(B4), axioms (N1) and (N2), a uniqueness-of-names axiom

$$UNA[F_1,....,F_n,And,Neg] \wedge UNA[A_1,....,A_m]$$

and a domain closure axiom for fluents

$$f=F_1 \vee .... \vee f=F_n \vee f=Neg(F_1) \vee .... \vee f=Neg(F_n) \quad \square$$

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

$$Holds(F1,Result(A,s)) \leftarrow Holds(F2,s)$$

and

$$Holds(Neg(F1),Result(A,s)) \leftarrow Holds(F3,s)$$

the sentence $\neg[Holds(F2,s) \wedge Holds(F3,s)]$ can be derived. In fact, domain constraints are entailed only from pairs of action descriptions of this form (see [Miller, 1994]). Hence the following definition is included, of narrative domain descriptions with no implicit domain constraints.

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

$$[Holds(F,Result(A,s)) \leftarrow [Holds(F_1,s) \wedge ... \wedge Holds(F_m,s)]]$$

and

$$[Holds(Neg(F),Result(A,s)) \leftarrow [Holds(F_{m+1},s) \wedge ... \wedge Holds(F_n,s)]]$$

there is some i, $1 \geq i \geq m$, and some j, $m+1 \geq j \geq n$, such that $F_i=Neg(F_j)$ or $F_j=Neg(F_i)$ □

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

Definition {Initially specified narrative domain description} A narrative domain description N is *initially specified* if for every positive fluent constant *F* in the language, either Holds(*F*,S0) or Holds(Neg(*F*),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 action occurs at time 1 followed by a shoot action at time 3. The full theory is referred to as N_{YSP}.

Example 1 {The Yale Shooting Problem, N_{YSP}}.

$$Holds(Neg(Alive),Result(Shoot,s)) \leftarrow Holds(Loaded,s) \quad (Y1)$$

$$Holds(Loaded,S0) \quad (Y2)$$

$$Holds(Alive,S0) \quad (Y3)$$

$$UNA[Alive,Loaded,And,Neg] \quad (Y4)$$

$$UNA[Sneeze,Shoot] \quad (Y5)$$

$$f=Alive \vee f=Loaded \vee f=Neg(Alive) \vee f=Neg(Loaded) \quad (Y6)$$

$$Happens(Sneeze,1) \quad (Y7)$$

$$Happens(Shoot,3) \quad (Y8)$$

□

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 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 theory is referred to as N_{GATE}.

Example 2. {N_{GATE}}.

$$Holds(Open,Result(Press,s)) \leftarrow Holds(Connected,s) \quad (G1)$$

$$Holds(Neg(Open),S0) \quad (G2)$$

$$UNA[Open,Connected,And,Neg] \quad (G3)$$

$$f=Open \vee f=Connected \vee f=Neg(Open) \vee f=Neg(Connected) \quad (G4)$$

$$Happens(Press,1) \quad (G5)$$

□

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

$$\begin{array}{l} \text{CIRC[N ; Absit ; Ab,Happens,Result,Holds,S0,State]} \\ \quad \wedge \text{ CIRC[N ; Ab,Happens ; Result,S0,State]} \end{array}$$

This circumscription policy is referred to as CIRC_n. Three theorems are useful at this point. Full proofs of these can be found in [Miller, 1994]. The first two theorems show that for the class of theory under consideration 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

$$\text{CIRC}_n[\text{N}] \models \text{CIRC}_b[\text{N}] \qquad \square$$

Theorem 2. Let N be a narrative domain description, and let Ψ be a sentence which does not contain the predicate symbol Happens and does not contain the function symbol State. Then

$$\text{CIRC}_n[\text{N}] \models \Psi \quad \text{if and only if} \quad \text{CIRC}_b[\text{N}] \models \Psi \quad \square$$

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

$$\text{CIRC}_n[\text{N}] \models \text{Happens(a,t)} \equiv \bigvee_{i=1}^{k} [\text{a}=A_i \wedge \text{t}=T_i]$$

and if k=0 then $\text{CIRC}_n[\text{N}] \models \neg\exists \text{a,t}[\text{Happens(a,t)}]$ $\square$

Theorems 1, 2 and 3 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 (Example 1), it can be shown[4] that

$$\text{CIRC}_n[\text{N}_{\text{YSP}}] \models \text{Holds(Neg(Alive),State(5))}$$

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: "Holds_at", "Initially", "Initiates", "Terminates", "ClippedBetween", "ClippedBefore", "Happens", "<" and "≤". 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 F and real number τ, the positive literal "Holds_at(F,τ)" is SLDNF-derivable from EC[N] only if $\text{CIRC}_n[\text{N}] \models \text{Holds}(F,\text{State}(\tau))$. The following definition of *fluent converses* will be useful.

Definition {Converse of a fluent} Let F be a ground fluent term. Then the *converse* of F, written F^*, is

- Neg(F) if F is a positive fluent constant
- F' if F is of the form Neg(F') for some positive fluent constant F' $\square$

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 {IN[IC]}. Let F be a ground fluent term and let IC be an initial conditions description of the form "Holds(F,S0)". The program clause IN[IC] is defined as

$$\text{Initially}(F) \qquad \square$$

Definition {INIT[AD] and TERM[AD]}. Let $F, F_1,...,F_n$ be ground fluent terms, A an action constant, s a situation variable and t a time-point variable. Let AD be an action description of the form

$$\begin{array}{l} \text{Holds}(F,\text{Result}(A,\text{s})) \leftarrow \\ \qquad [\text{Holds}(F_1,\text{s}) \wedge ...... \wedge \text{Holds}(F_n,\text{s})] \end{array}$$

The program clause INIT[AD] is defined as

$$\begin{array}{l} \text{Initiates}(A,F,\text{t}) \leftarrow \\ \qquad [\text{Holds\_at}(F_1,\text{t}) \wedge ...... \wedge \text{Holds\_at}(F_n,\text{t})] \end{array}$$

and the program clause TERM[AD] is defined as

$$\begin{array}{l} \text{Terminates}(\text{A},F^*,\text{t}) \leftarrow \\ \quad [\textit{not}\ \text{Holds\_at}(F_1^*,\text{t}) \wedge ...... \wedge \textit{not}\ \text{Holds\_at}(F_n^*,\text{t})] \quad \square \end{array}$$

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. The following definition gives a complete translation of a narrative domain description into a logic program.

[4]A derivation of this is given in [Miller, 1994]

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

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

together with the following domain-independent clauses

Holds_at(f,t) $\leftarrow$ (EC1)
[Initially(f) $\wedge$ *not* ClippedBefore(f,t)]

Holds_at(f,t3) $\leftarrow$ (EC2)
[Happens(a,t1) $\wedge$ t1<t3 $\wedge$
Initiates(a,f,t1) $\wedge$
not ClippedBetween(t1,f,t3)]

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

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

Clauses (EC1) and (EC2) above are persistence axioms. Clause (EC1) states that a fluent *F* holds at a time *T* if it is initially true and it has not been cancelled ("clipped") before *T* by some action. Clause (EC2) states that a fluent holds at a time *T3* if it is initiated by an action occurrence at some previous time *T1*, 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 "Holds_at(Neg(f),t) $\leftarrow$ *not* Holds_at(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

$$\text{EC[N]} \nvdash_{SLDNF} \text{Holds\_at}(F,\tau)$$

which should be interpreted as "it is not provable that *F* holds at time τ", and

$$\text{EC[N]} \vdash_{SLDNF} \text{Holds\_at(Neg}(F),\tau)$$

which should be interpreted as "*F* does not hold at time τ".

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

Initiates(Shoot,Neg(Alive),t) $\leftarrow$ INIT[(Y1)]
Holds_at(Loaded,t)

Terminates(Shoot,Alive,t) $\leftarrow$ TERM[(Y1)]
not Holds_at(Neg(Loaded),t)

Initially(Loaded) IN[(Y2)]

Initially(Alive) IN[(Y3)]

Happens(Sneeze,1) (Y7)

Happens(Shoot,3) (Y8)

so that

$$\text{EC}[N_{YSP}] \vdash_{SLDNF} \text{Holds\_at(Neg(Alive),5)}$$

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

Initiates(Press,Open,t) $\leftarrow$ INIT[(G1)]
Holds_at(Connected,t)

Terminates(Press,Neg(Open),t) $\leftarrow$ TERM[(G1)]
not Holds_at(Neg(Connected),t)

Initially(Neg(Open)) IN[(G2)]

Happens(Press,1) (G5)

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

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

then the query "Holds_at(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 theory 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 *F* 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(A,τ')". Then $\text{CIRC}_n[N] \models \text{Holds}(F,\text{State}(\tau))$ if $\text{EC[N]} \vdash_{SLDNF} \text{Holds\_at}(F,\tau)$ □

Theorem 5. Let N be an initially specified fluent independent narrative domain description, let τ be a real number, and let *F* 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(A,τ')". Then $\text{CIRC}_n[N] \models \text{Holds}(F,\text{State}(\tau))$ if and only if $\text{EC[N]} \vdash_{SLDNF} \text{Holds\_at}(F,\tau)$ □

Space limitations do not permit full proofs to be given here; these can be found in [Miller, 1994]. 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* [Lin & Shoham, 1991]. 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 show that, in this general case, it is possible to partially compute what holds in any situation named by action sequences 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) provides precisely the strength of default persistence required.

The Situation Calculus style programs contain a clausal counterpart to the frame axiom (F1). They also contain domain-specific Ab clauses analogous to the Terminates clauses described above. The essential difference between the two is in their expression of default persistence, and their correspondence is shown using induction on the number of occurrence descriptions in the narrative.

In [Miller, 1994] a simple general procedure is described which transforms programs so that they are both sound and complete in the more general case (i.e. where the domains are not initially specified). The transformed programs achieve completeness by testing queries with every possible consistent initial situation, using "second order" programming techniques.

5. 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 [1994] is closely related to 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 [1993] 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 [1995] 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 [1993] shows a correspondence between Baker's formalism and the *Language A* introduced in [Gelfond & Lifschitz, 1992]. Furthermore, Gelfond and Lifschitz, Dung [1993], Baral and Gelfond [1993], and Denecker and De Schreye [1993] have each shown how the *Language 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

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

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

[Denecker & De Schreye, 1993], Marc Denecker and Danny De Schreye, Representing Incomplete Knowledge in Abductive Logic Programming, *Proceedings, ISLP 1993*

[Dung, 1993], Phan Minh Dung, Representing Actions in Logic Programming and its Applications in Database Updates, *Proceedings ICLP*, ed David S. Warren, MIT Press, pages 222-238

[Gelfond & Lifshitz, 1992] Michael Gelfond and Vladimir Lifschitz, Representing Actions in Extended Logic Programming, *Proceedings, JICSLP*, ed. Krzysztof Apt, MIT Press, page 560

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

[Kowalski & Sadri, 1994] R.A.Kowalski and F.Sadri, The Situation Calculus and Event Calculus Compared, *Proceedings, ILPS 94*

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

[Lifschitz, 1995] V.Lifschitz, Circumscription, in *Handbook of Logic in Artificial Intelligence*, ed. D.Gabbay, C.Hogger and J.A.Robinson, Oxford University Press, pages 297-352.

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

[Miller, 1994] Rob Miller, Narratives in the Context of Temporal Reasoning, *Imperial College Research Report DoC 94/3*, 1994

[Miller & Shanahan, 1994] R.S.Miller and M.P.Shanahan, Narratives in the Situation Calculus, *Journal of Logic and Computation — Special Issue on Actions and Processes* (1994), Volume 4, number 5, Oxford University Press, pages 513–530

[Pinto & Reiter, 1993] J. Pinto and R. Reiter, Temporal Reasoning in Logic Programming: A Case for the Situation Calculus, in *Proceedings ICLP 93.*, ed. David S. Warren, MIT Press

[Shanahan, 1995] M.P.Shanahan, A Circumscriptive Calculus of Events, *Artificial Intelligence* (1995), Volume 75, number 2.