In The Journal of Logic Programming, Special Issue: reasoning about action and change, 31(1-3), Pages 39-58, 1997.

Reconciling the Event Calculus with the Situation Calculus

Robert Kowalski and Fariba Sadri
Department of Computing, Imperial College,
180 Queen's Gate, London SW7 2BZ

Emails: rak@doc.ic.ac.uk fs@doc.ic.ac.uk

Abstract

In this paper, to compare the situation calculus and event calculus we formulate both as logic programs and prove properties of these by reasoning with their completions augmented with induction. We thus show that the situation calculus and event calculus imply one another. Whereas our derivation of the event calculus from the situation calculus requires the use of induction, our derivation of the situation calculus from the event calculus does not. We also show that in certain concrete applications, such as the missing car example, conclusions that seem to require the use of induction in the situation calculus can be derived without induction in the event calculus. To compare the two calculi, we need to make a number of small modifications to both. As a by-product of these modifications, the resulting calculi can be used to reason about both actual and hypothetical states of affairs, including counterfactual ones. We further show how the core axioms of both calculi can be extended to deal with domain or state constraints and certain types of ramifications. We illustrate this by examples from legislation and the blocks world.

Keywords: temporal reasoning, event calculus, situation calculus, actual and hypothetical events

1 Introduction

In this paper we revisit our earlier comparison of the situation calculus and event calculus [11] using a more general approach to the representations of situations, in the spirit of Van Belleghem, Denecker and De Schreye [20, 21]. The resulting more general formalisations of both calculi combine the situation calculus ability to represent hypothetical situations with the event calculus representation of actual events. For simplicity, we restrict the representation to times which are situations or transitions between situations, in the spirit of Kakas and Miller [6].

We formulate both calculi as logic programs and reason with their iff-completions, induction over situations and integrity constraints as approximations to the intended semantics of those programs. The technique of reasoning about logic programs using iff-completions and induction was introduced by Clark and Tarnlund [4]. The resulting formalisation of the situation calculus is similar to the formalisation of Pinto and Reiter [15].

We show that the main result, reported in [11], holds for these more general versions of the event and situation calculi. The result states that the two calculi logically imply one another. Our derivation of the event calculus from the situation calculus, however, requires the use of induction, whereas our derivation of the situation calculus from the event calculus does not. This indication that the event calculus may be more powerful than the situation calculus, despite their equivalence, is also supported by our analysis of the missing car example (in section 7).

We allow events, such as updating deductive databases or passing, amending and repealing legislation, which initiate and terminate the holding of arbitrary sentences. We formalise reasoning with such sentences using a proof predicate defined as a metalogic program. State or domain constraints and certain kinds of ramifications are dealt with automatically as a by-product of this formulation.

In sections 2, 3 and 4 we present logic programming formulations of the situation and event calculi and their semantics in terms of iff-completions, integrity constraints and induction. In section 5, for ease of reference, we list all the axioms and integrity constraints presented thus far. In section 6 we show that the situation calculus with induction implies the event calculus, whereas the event calculus without induction implies the situation calculus. In sections 7 and 8 we show how the calculi can be used for factual and hypothetical reasoning, including counterfactual reasoning, and how they can be extended to deal with ramifications. Finally, in section 9 we conclude the work presented in this paper.

2 The situation calculus

2.1 The axioms

The following two clauses constitute the core of our formulation of the situation calculus as a logic program. Together they define what sentences *P* hold in the situation *result(A, S)* that is the result of the transition from state *S* by an action of type *A.*

holds(P, result(A, S)) ← happens(A, S) ∧ initiates(A, S, P) S1

holds(P, result(A, S)) ← happens(A, S) ∧ holds(P, S) ∧ ¬ terminates(A, S, P) S2

Terms starting in the upper case are variables. Unless otherwise indicated, all variables are assumed to be universally quantified in front of the formula in which they occur.

The first clause defines what sentences hold because they are initiated by an action. The second clause, called the *frame axiom*, defines what sentences hold because they held before the action and were not terminated by it.

The definitions of the predicates *initiates* and *terminates* are domain-dependant. For example, the fact that an act of donor X giving object Y to recipient Z initiates the property of Z possessing Y and terminates the property of X possessing Y can be defined by the two clauses:

initiates(give(X, Y, Z), S, possess(Z, Y))
terminates(give(X, Y, Z), S, possess(X, Y))

Similarly, in the Yale shooting problem, the fact that if the gun is loaded then an act of shooting initiates the death of the turkey and terminates its life can be represented by the conditional clauses:

initiates(shoot, S, dead) ← holds(loaded, S)
terminates(shoot, S, alive) ← holds(loaded, S)

Most formulations of the situation calculus use a single predicate *abnormal* ([1], for example) instead of the two predicates *initiates* and *terminates*. The frame axiom in this formulation is often given in the form:

$$[\text{holds}(P, S) \leftrightarrow \text{holds}(P, \text{result}(A, S))] \leftarrow \neg\text{abnormal}(A, S, P)$$

In our formulation we can derive an analogous formula

$$[\ \text{holds}(P, S) \leftrightarrow \text{holds}(P, \text{result}(A, S))\ ] \leftarrow \text{happens}(A, S) \wedge \neg\text{initiates}(A, S, P) \wedge \neg\text{terminates}(A, S, P)$$

by using the iff-completion of clauses S1 and S2, which we use for reasoning about our formulation of the situation calculus program. Our use of the iff-completion is discussed in section 3.

Among those formulations of the situation calculus which employ an *abnormality* predicate, many, in effect, interpret *abnormal(A, S, P)* as "*A* initiates *P* in *S* or *A* initiates $\neg P$ in *S*". This prevents two explicitly contradictory properties *P* and $\neg P$ from holding simultaneously. However, it does not prevent the simultaneous holding of two (or more) properties P_1, ..., P_n, which are implicitly contradictory, due to the presence of domain-specific integrity constraints such as

$$\neg\ [\ \text{holds}(P_1, S) \wedge \ldots \wedge \text{holds}(P_n, S)\ ].$$

Our formulation does not prevent explicitly or implicitly contradictory properties from holding simultaneously. We discuss this issue in [17] in the context of the event calculus, and show how the problem can be dealt with in different ways, depending on whether or not the given set of event descriptions is complete.

We use the predicates *initiates* and *terminates* instead of *abnormal* because they facilitate our comparison with the event calculus. Moreover, it can be argued that they provide a semantically more meaningful formulation of the situation calculus itself, distinguishing between abnormalities due to initiation of sentences and those due to termination.

2.2 Restricting situations to possible situations

The inclusion of the condition *happens(A, S)* in S1 and S2 restricts the holding of properties to situations which are possible. The notion of possible situation can also be represented explicitly by the type predicate *situation*, defined as follows:

situation(s0) T1
situation(result(A, S)) $\leftarrow$ happens(A, S) $\wedge$ situation(S) T2

The condition *happens(A, S)* in S1 and S2 could, consequently, be replaced by the condition *situation(result(A, S))*, which introduces *happens(A, S)* indirectly.

The following integrity constraints ensure that any event which happens is a possible event, in the sense that it happens in a possible situation and all its preconditions actually or hypothetically hold in that situation:

holds(P, S) $\leftarrow$ happens(A, S) $\wedge$ precondition(A, S, P) I1
situation(S) $\leftarrow$ happens(A, S) I2

Given I2, the two formulations of the situation calculus, one with *happens* and the other with *situation*, are equivalent.

Integrity constraints, such as I1 and I2, can be thought of as imposing restrictions on predicates, such as *happens*, whose definition is unknown or incomplete. We will discuss the semantics of integrity constraints in the next section.

Whether an event occurrence, *happens(a, s)*, or a property holding, *holds(p, s)*, is actual or hypothetical can be reduced to determining whether the situation *result(a, s)* or *s*, respectively, is actual or hypothetical. This, in turn, can be done by means of assertions such as

actual(s0)
.
.
actual(s)
actual(result(a, s))

where at most one sequence of events is asserted as actual. The distinction between actual and hypothetical situations does not affect the statement or the proof of the main theorem, presented in section 6. We shall discuss actual and hypothetical events in more detail in section 7.

As with the domain-specific predicates *initiates* and *terminates*, the definition of the predicate *precondition* is domain dependent, for example:

precondition(give(X, Y, Z), S, possess(X, Y))

In the standard formulations of the situation calculus, such preconditions of actions are ignored, or else are catered for by including extra conditions in the analogues of clauses S1 and S2. In fact, our inclusion of the condition *happens(A, S)* in S1 and S2, together with the integrity constraint I1, achieves a similar effect to the latter of these two approaches.

2.3 The initial situation

Before we discuss the semantics of the situation calculus in the next section, we need to discuss the representation of the initial situation, s0. There are at least two possibilities. We can define the properties of the initial situation by means of a clause

holds(P, s0) ← initially(P)

together with appropriate clauses defining the predicate *initially.*

Alternatively, we may assume, without loss of generality, that nothing holds in the initial situation *s0* and that all properties of the initial situation are actually properties of the successor of the initial situation, *result(creation, s0)*, where *creation* is defined by clauses of the form *initiates(creation, s0, P).* For our purpose, the second alternative is more convenient than the first.

Because a *creation* event can initiate any property and has no preconditions, we need an integrity constraint

S = s0 ← happens(creation, S) I3

to prevent asserting the occurrence of *creation* in any situation other than the initial situation.

3 The semantics of the situation calculus program

Many different semantics have been defined for logic programs. To compare the situation and event calculi, it is convenient to use the completion semantics [3] augmented with integrity constraints and induction [4].

Like circumscription [14], the completion semantics augmented with induction seeks to formalise the intention that a set of axioms (in our case, the if-halves of logic programs) characterises the *smallest* set of relations which satisfy the axioms. In the case of a program consisting of Horn clauses without equality, there is only one such smallest set, which is the unique minimal Herbrand model of the program [22]. This is the semantics associated with the program by (virtually) all existing semantics of logic programming. For such Horn clause programs, the iff-completion is an approximation to the intended meaning, and the iff-completion augmented with axioms of induction is a better approximation.[1]

The case of normal programs, like S1 and S2, is more complicated. Not only do circumscription and logic programming semantics diverge, but the different semantics for logic programming also diverge. What many of these semantics have in common, however, is that the iff-completion, augmented with induction, is a safe approximation to them. It is safe in the sense that any logical consequence of the iff-completion augmented with induction is a *property* of the program, in that it is true in all *intended models* of the program characterised by these semantics.

3.1 Iff-completion

The construction of the completion of a logic program is well known [3, 13]. Applied to S1 and S2, this construction gives:

$$\begin{aligned}
&\text{holds}(P, S2) \leftrightarrow \\
&\exists A1{,}S1\ [\ [S2 = \text{result}(A1, S1) \land \text{happens}(A1, S1) \land \text{initiates}(A1, S1, P)\ ] \lor \\
&\qquad [S2 = \text{result}(A1, S1) \land \text{happens}(A1, S1) \land \text{holds}(P, S1) \land \\
&\qquad \neg\ \text{terminates}(A1, S1, P)]\ ] \qquad\qquad \text{SC}
\end{aligned}$$

together with the Clark equality theory (CET), which consists of the *unique name axioms* including

$$[A1{=}A2 \land S1{=}S2] \leftarrow \text{result}(A1, S1) = \text{result}(A2, S2) \qquad \text{CET1}$$
$$\neg\ \exists\, S\ [s0 = \text{result}(A, S)] \qquad \text{CET2}$$

and the usual axioms of equality.

Similarly, we need to complete the definitions of the *initiates*, *terminates*, *precondition* and *situation* predicates, but not *happens* and *actual*.

3.2 Integrity constraints

Integrity constraints in relational and deductive databases are first order sentences that express invariant properties of relations whose definitions can be updated. Like iff-

[1] It can be argued that the iff-completion together with a first-order axiom schema of induction approximates the intended minimal Herbrand model in exactly the same sense that the logical consequences of the Peano axioms of arithmetic approximate the intended model of arithmetic. This model can be understood, in turn, as the unique minimal Herbrand model of the Horn clauses which define addition and multiplication.

completions and induction, they provide a *safe approximation* to the intended semantics of the definitions in the sense that they are meant to be *true in all intended models*.

Integrity constraints, understood in this way, do not affect the intended meaning of definitions, but simply help to characterise that meaning in syntactic terms (like the only-if halves of definitions and induction axioms). Thus integrity constraints can safely be used as additional premises to prove properties of logic programs.

Integrity constraints can play a useful role both when they describe properties of predicates whose definition is given, and when they describe properties of predicates whose definitions are unknown or inaccessible. A good (and instructive) example of the latter use of integrity constraints is in semantic query optimisation [2], where integrity constraints describing extensional predicates are used together with definitions of intensional predicates, to optimise a query before transmitting it to the extensional database. Integrity constraints can similarly be used to describe properties of abducible predicates, whose definitions are unknown or incomplete, and which can appear in answers to queries, in much the same way that extensional predicates appear as the result of transforming a query by means of semantic query optimisation.

In our formulations of the situation calculus and event calculus, the predicates *happens* and *actual* can be understood as predicates with inaccessible or only partially accessible definitions[2]. Integrity constraints, such as I1-I3, are properties of the complete, but unknown definitions. Consequently these integrity constraints can be used as premises to prove properties of logic programs such as S1 and S2, augmented with definitions for *happens* and *actual*. So long as in any given application, whenever a complete definition of these predicates is given, the integrity constraints hold in all intended models of these definitions, then any properties proved with the aid of the integrity constraints will also be properties of the definitions themselves.

3.3 Temporal ordering

One of the limitations of most formulations of the situation calculus is that situations need to be named explicitly in terms of a sequence of actions following the initial state. As a result, it is virtually impossible to represent narratives with incomplete information. This limitation can be overcome by relating situations to one another by means of an ordering relation. Such ordering relations have been defined for the situation calculus by Shanahan [19] and Pinto and Reiter [15].

For our purposes, it is convenient to introduce a term, *trans(A, S)*, representing the transition from situation *S* to *result(A, S)* caused by an event of type *A*. This allows the occurrence of any number of alternative events immediately following a given situation, giving rise to branching timelines.

[2] In section 7, we will consider the case where *happens(A, S)* is interpreted as "*A* is possible in *S*", in which case *happens* can be defined in such a way that I1-I3 are properties of the definition.

In the following definitions, the auxiliary predicate << represents the immediate succession of time points, and < represents its transitive closure:

$S << trans(A, S) \leftarrow situation(result(A, S))$
$trans(A, S) << result(A, S) \leftarrow situation(result(A, S))$
$T1 < T2 \leftarrow T1 << T2$
$T1 < T2 \leftarrow T1 < T \wedge T << T2$

As is the case with all definitions, these definitions need to be completed. We name these completions LESS.

We also need the following unique name axioms in the Clark equality theory:

$\neg\exists\ A, S\ [result(A, S) = trans(A, S)]$	CET3
$\neg\exists\ A, S\ [s0 = trans(A, S)]$	CET4
$[A1=A2 \wedge S1=S2] \leftarrow trans(A1, S1) = trans(A2, S2)$	CET5

3.4 Induction

Induction can be defined either over all time points or over all situations. The latter is more convenient for our purposes.

$\forall S\ [\Phi(S) \leftarrow situation(S)] \leftarrow$
$\quad \Phi(s0) \wedge \forall A,S\ [\Phi(result(A, S)) \leftarrow \Phi(S) \wedge situation(result(A, S))]$ IND

where Φ is a metavariable standing for any first-order formula (as in Peano arithmetic).

Given the completed definitions of <<, <, and *situation*, the unique name axioms, and IND, we can show the following properties of the predicate < . We need (a), (b), (c), and (f) in the proof of the main theorem in section 6, and we use (j) in the missing car example in section 7.

a)	$\neg\exists S\ [S < s0]$
b)	$\forall S1,S2,S3\ [S1 < S3 \leftarrow S1 < S2 \wedge S2 < S3]$
c)	$\forall S1,S2\ [\neg S2 \leq S1 \leftarrow S1 < S2]$
d)	$\forall S,A\ \neg\exists S1\ [S < S1 \wedge S1 < result(A, S) \wedge situation(S1)]$
e)	$\forall S\ \neg S < S$
f)	$\forall S,A\ \neg\exists T\ [trans(A, S) < T \wedge T < result(A, S)]$
g)	$\forall S,A\ \neg\exists T\ [S < T \wedge T < trans(A, S)]$
h)	$\forall S1,S2\ [situation(S1) \leftarrow S1<S2 \wedge \neg\exists A,S\ (S1=trans(A, S))]$
i)	$\forall S1,S2\ [situation(S2) \leftarrow S1<S2 \wedge \neg\exists A,S\ (S2=trans(A, S))]$
j)	$\forall S1,S2,A\ [happens(A, S1) \leftarrow trans(A, S1)<S2]$

4 The simplified event calculus

The original version of the event calculus [12] allowed concurrent and partially ordered events as well as the persistence of properties both backwards and forwards over time. It also catered for properties holding both during time periods and at time points.

To compare the event calculus with the situation calculus we shall consider a special case of the former that is analogous to the restrictions of the latter. We can obtain this special case by removing all consideration of backward persistence of properties and pre-processing away all reference to time periods. We do the latter by replacing all conditions referring to properties holding for time periods by their definition in terms of events happening at time points. Such a special case was introduced in ([10], [5], [18]). In [17] we showed that this special case, in completion form augmented with integrity constraints, has much of the power of the original event calculus, and even overcomes some of its deficiencies.

The special case of the event calculus has at its core a single axiom:

$$\begin{aligned} \text{holds}(P, T2) \leftarrow\ & \text{happens}(E1, T1) \wedge \text{initiates}(E1, P) \wedge T1 < T2 \wedge \\ & \neg \exists E^*,T^*[\text{happens}(E^*, T^*) \wedge \text{terminates}(E^*, P) \wedge T1 \leq T^* \wedge T^* < T2] \end{aligned}$$

which expresses that a property holds at a time point if it was initiated by an event which happened at an earlier time point and no intervening event happened to terminate the property.

The condition $T1 < T2$ has the effect of ensuring that properties hold only *after* their initiating event happens. The inequality $T^* < T2$ ensures that properties, including preconditions of events, hold at the time they are terminated by some event.

We further restrict our version of the event calculus to the special case where the < relation on time points is identified with the < relation on situations and transactions. In this case we can replace the condition $T1 \leq T^*$ by the inequality $T1 < T^*$, using the assumptions that two different events do not occur at the same time (expressed by CET5), and that no event initiates and terminates the same property. Also event tokens are unnecessary because individual events can be identified by their action or event types and the situations in which they cause a transition. We may, therefore, instantiate the event calculus axiom by means of a particular representation for events, representing event tokens E by pairs (A, S) consisting of an action or event type A and a situation S. This reformulates the event calculus predicates in the situation calculus vocabulary, as follows:

Replace	happens(E,T)	by	happens((A, S), trans(A, S))
	where trans(A, S) is the time T between S and result(A, S) when E occurs. For simplicity we abbreviate this to happens(A, S).		
Replace	initiates(E, P)	by	initiates(A, S, P).

Replace terminates(E, P) by terminates(A, S, P).

Finally, we restrict the holding of properties to situations. The resulting completion of the event calculus axiom has the iff-form:

holds(P, S2) $\leftrightarrow$
situation (S2) $\wedge$
$\exists$A1,S1 { happens(A1, S1) $\wedge$ initiates(A1, S1, P) $\wedge$ trans(A1, S1) < S2 $\wedge$
$\neg$ $\exists$A*,S* [happens(A*, S*) $\wedge$ terminates(A*, S*, P) $\wedge$
trans(A1, S1) < trans(A*, S*) $\wedge$ trans(A*, S*) < S2] } EC

Like our version of the situation calculus, this formulation of the simplified event calculus can be used for reasoning with both actual and hypothetical situations and events.

Our formulations of the situation calculus and of the simplified event calculus differ only in the core axioms SC and EC. The remaining axioms, IND, LESS, CET, constraints I1-I3 and the completion of the type definition T1 and T2, are common to both calculi.

5 Summary of all axioms and integrity constraints

The situation calculus core axiom:

SC holds(P, S2) $\leftrightarrow$
$\exists$A1,S1 [[S2= result(A1, S1) $\wedge$ happens(A1, S1) $\wedge$ initiates(A1, S1, P)] $\vee$
[S2= result(A1, S1) $\wedge$ happens(A1, S1) $\wedge$ holds(P, S1) $\wedge$
$\neg$ terminates(A1, S1, P)]]

The event calculus core axiom:

EC holds(P, S2) $\leftrightarrow$
situation (S2) $\wedge$
$\exists$A1,S1 { happens(A1, S1) $\wedge$ initiates(A1, S1, P) $\wedge$ trans(A1, S1) < S2 $\wedge$
$\neg\exists$A\*,S* [happens(A*, S*) $\wedge$ terminates(A*, S*, P) $\wedge$
trans(A1, S1) < trans(A*, S*) $\wedge$ trans(A*, S*) < S2] }

The situation type definition (i.e. the completion of definition T1 and T2):

TYPE situations(S) $\leftrightarrow$
S=s0 $\vee$
$\exists$A1,S1 [S=result(A1, S1) $\wedge$ happens(A1, S1) $\wedge$ situation(S1)]

The definitions of << and < :

LESS T1 << T2 $\leftrightarrow$
$\exists$A1,S1 { [T2=trans(A1, T1) $\wedge$ situation(result(A1 , T1)] $\vee$

$[T1=trans(A1, S1) \wedge T2=result(A1, S1) \wedge situation(result(A1, S1))] \}$

$T1 < T2 \leftrightarrow$
$T1 << T2 \vee \exists T\, [\, T1 < T \wedge T << T2\,]$

The integrity constraints:

I1	$holds(P, S) \leftarrow happens(A, S) \wedge precondition(A, S, P)$
I2	$situation(S) \leftarrow happens(A, S)$
I3	$S = s0 \leftarrow happens(creation, S)$

The axiom schema of induction:

IND $\forall S\ [\Phi(S) \leftarrow situation(S)] \leftarrow$
$\Phi(s0) \wedge \forall A,S\ [\Phi(result(A, S)) \leftarrow \Phi(S) \wedge situation(result(A, S))]$

The Clark unique name axioms

CET1	$[A1=A2 \wedge S1=S2] \leftarrow result(A1, S1) = result(A2, S2)$
CET2	$\neg \exists S\ [s0 = result(A, S)]$
CET3	$\neg\exists A, S\ [result(A, S) = trans(A, S)]$
CET4	$\neg\exists A, S\ [s0 = trans(A, S)]$
CET5	$[A1=A2 \wedge S1=S2] \leftarrow trans(A1, S1) = trans(A2, S2)$

6 The relationship between the two calculi

The relationship between our formulations of the situation and event calculi can now be stated as follows.

Theorem: i) COMM, EC |= SC
ii) COMM, SC, IND |= EC

where SC and EC are the iff-forms of the situation calculus and event calculus core axioms respectively, IND is the axiom schema of induction and COMM consists of I2, LESS, TYPE and the properties (a), (b), (c) and (f) common to the two calculi.

Proof:

i) Given COMM, EC is equivalent to

$holds(P, S2) \leftrightarrow (event1) \vee (event2)$

where (event1) has the form

$\exists A1,S1$

$\{$ situation(S2) $\wedge$ happens(A1, S1) $\wedge$ initiates(A1, S1, P) $\wedge$ S2=result(A1, S1) $\wedge$
$\neg \exists$ A*,S* [happens(A*, S*) $\wedge$ terminates(A*, S*, P) $\wedge$
trans(A1, S1)< trans(A*, S*) $\wedge$ trans(A*, S*)< S2] $\}$

and (event2) has the form

$\exists$ A1,S1,T
$\{$ situation(S2) $\wedge$ happens(A1, S1) $\wedge$ initiates(A1, S1, P) $\wedge$ trans(A1, S1)< T $\wedge$ T<<S2 $\wedge$
$\neg \exists$ A*,S* [happens(A*, S*) $\wedge$ terminates(A*, S*, P) $\wedge$
trans(A1, S1)< trans(A*, S*) $\wedge$ trans(A*, S*)< S2] $\}$

This equivalence can be shown by replacing the condition trans(A1, S1)< S2 in EC by its definition in LESS and distributing disjunction over conjunction.

On the other hand, SC is equivalent to

holds(P, S2) $\leftrightarrow$ (sit1) $\vee$ (sit2)

where (sit1) has the form

$\exists$ A,S [S2 = result(A, S) $\wedge$ happens(A, S) $\wedge$ initiates(A, S, P)]

and (sit2) has the form

$\exists$ A,S [S2 = result(A, S) $\wedge$ happens(A, S) $\wedge$ holds(P, S) $\wedge$ $\neg$ terminates(A, S, P)]

It is not difficult to show that (event1) and (sit1) are equivalent in the theory COMM using lemma (f), stating that there is no time *T* between *trans(A1, S1)* and *result(A1, S1)*, and the assumption I2, that events happen only in situations.

Also, using LESS and TYPE, (event2) is equivalent to

$\exists$ A1,S1,A,S,T
$\{$ S2=result(A, S) $\wedge$ happens(A, S) $\wedge$ situation(S) $\wedge$
happens(A1, S1) $\wedge$ initiates(A1, S1, P) $\wedge$ trans(A1, S1)<T $\wedge$ T<<result(A, S) $\wedge$
$\neg$ terminates(A, S, P) $\wedge$
$\neg\exists$A\*,S* [happens(A*, S*) $\wedge$ terminates(A*, S*, P) $\wedge$
trans(A1, S1)< trans(A*, S*) $\wedge$ trans(A*, S*)< S] $\}$

But this is equivalent in turn to (sit2) using the following instance EC* of EC

holds(P, S) $\leftrightarrow$
situation(S) $\wedge$
$\{\exists$ A1,S1$\{$ happens(A1, S1) $\wedge$ initiates(A1, S1, P) $\wedge$ trans(A1, S1)< S $\wedge$
$\neg\exists$ A*,S*[happens(A*, S*) $\wedge$ terminates(A*, S*, P) $\wedge$

$$\text{trans}(A1, S1) < \text{trans}(A^*, S^*) \wedge \text{trans}(A^*, S^*) < S] \}\}$$

and the fact that given *situation(result(A, S))*

$$\exists T [\text{trans}(A1, S1) < T \wedge T << \text{result}(A, S)] \equiv \text{trans}(A1, S1) < S$$

by LESS.

ii) The proof of this part of the theorem is identical to that of part (i), up to the point of using EC* to show that (event2) is equivalent (in COMM) to (sit2). For part (ii) we use induction with EC* as the induction hypothesis and with EC as the theorem to be proved. This together with the assumption SC proves the induction conclusion EC. To complete the proof, we need to prove the base case of the induction, i.e. to show that EC holds when S2=s0. But EC in this case is equivalent to

$$\neg \exists P [\text{holds}(P, s0)]$$

which states that nothing holds in the initial state. This is an immediate consequence of SC.

So by induction we have shown

$$\forall S2 \{ [\text{holds}(P, S2) \leftrightarrow \text{conditions of EC}] \leftarrow \text{situation}(S2) \}$$

Since under SC and I2 both the conditions of EC and the head, *holds(P, S2)*, imply *situation(S2)*, this condition can be removed from above, and thus we prove EC.

Q.E.D.

7 Actual and hypothetical situations

7.1 Distinguishing between actual and hypothetical situations

Whereas the event calculus was originally intended for representing and reasoning about actual events, the situation calculus was designed for reasoning about hypothetical events and situations. Our inclusion of the conditions *happens(A1, S1)* in SC and EC is intended to unify the treatment of hypothetical and actual events.

In the case of the standard versions of the event calculus, in which all events are actual, the integrity constraints I1-I3 restrict actual events to those which are possible. In the case of the standard formulations of the situation calculus, however, in which all events are hypothetical, the same integrity constraints are properties of the definition of what it means for an event of type *A* to be possible in a situation *S*. The definition can be given as follows:

$$\text{happens}(A, S) \leftrightarrow$$
$$\{ \forall P [\text{holds}(P, S) \leftarrow \text{precondition}(A, S, P)] \wedge \text{situation}(S) \wedge [S=s0 \leftarrow A=\text{creation}] \}$$

We employ the integrity constraints, I1-I3, because they capture the common ground shared by actual and hypothetical events. Notice that I1-I3 are the only-if half of the definition.

To represent an actual event, in both calculi, we simply assert that the event has happened and that the resulting situation is actual. To represent a hypothetical event, we also assert that the event has happened (either because it is given as input or because it is assumed by abduction), but say nothing about whether or not the resulting situation is actual.

To ensure that at most one sequence of events is actual, we need to ensure that the following integrity constraints are properties of any definition of *actual*:

actual(S) ← actual(result(A, S))
A1=A2 ← actual(result(A1, S)) ∧ actual(result(A2, S))

7.2 Reasoning with multiple timelines

The following example, based on an example in [20], shows how information about an event on one timeline can be used to derive information about an event on another timeline. The example is neutral with respect to whether any of the two timelines is actual.

Here is an informal description of the problem. The event of *creation* happens in the initial state and initiates a gun being loaded and an individual being alive. Then an event of spinning the chamber of the gun happens. We then consider two alternative scenarios. In the first, an event of waiting happens, and in the second an event of shooting. The problem is to show that after the shooting the individual is not alive if after the waiting the gun would have remained loaded.

Such reasoning is possible in the standard formulations of the situation calculus. Here we show that it is also possible in our formulation of the event calculus. We formalise the problem as follows.

Suppose we are given

holds(loaded, result(wait, result(spin, result(creation, s0)))) (1)

where the if-halves of the definitions of *initiates* and *terminates* include the following assertions.

initiates(creation, s0, loaded) (2)
initiates(creation, s0, alive) (3)
terminates(shoot, S, alive) ← holds(loaded, S) (4)

Assume, moreover, that it is a property of the complete definitions that

$\neg \exists S$ initiates(shoot, S, alive) (5)
$\neg \exists S$ initiates(wait, S, loaded) (6)
$\neg \exists S$ initiates(spin, S, loaded) (7)

We make no other assumptions about the definitions of *initiates* and *terminates*.

From assumption (1), stating that the gun would be loaded after waiting, using the only-if half of EC, (6) and (7), we can derive

$\neg$ terminates(spin, result(creation, s0), loaded)

Now, using the if-half of EC, we can conclude

holds(loaded, result(spin, result(creation, s0)))

Therefore, by (4)

terminates(shoot, result(spin, result(creation, s0)), alive)

Finally, using (5) and the contrapositive of the only-if half of EC we conclude

$\neg$holds(alive, result(shoot, result(spin, result(creation, s0))))

Moreover, using (1), EC and TYPE, we can show

situation(result(shoot, result(spin, result(creation, s0)))).

There are two relatively minor differences between this example and the treatment of the example in [20]. First, we use the situation structure of the situation calculus, whereas [20] uses more general time points, in the spirit of the original simplified event calculus. Second, we use properties (2)-(7) of the completion of *initiates* and *terminates*, whereas [20] uses the iff-completion of the definitions explicitly.

7.3 The missing car example

This problem [7] illustrates the theorem of section 6. The problem can be solved in both calculi. However, its solution requires induction in the situation calculus but not in the event calculus.

The scenario is that a car is parked in a car park initially, but it is not there later. The problem is to explain the absence of the car from the car park.

Formally, we know that

$\exists S'$ [$\neg$holds(in, S') $\wedge$ trans(park, s0) $<$ S' $\wedge$ situation(S')] (1)

Furthermore, we assume that the following assertions are included in the definitions of *initiates* and *terminates*.

initiates(park, S, in) (2)
terminates(steal, S, in) (3)
terminates(tow, S, in) (4)

Assume, moreover, that it is a property of the complete definitions that

$$\text{terminates}(A, S, \text{in}) \rightarrow A{=}\text{steal} \lor A{=}\text{tow} \quad (5)$$

We would like to conclude that at some time after the car was parked and before S' it was stolen or towed away.

From assumption (1), using the contrapositive of the if-half of EC, lemma (j) of subsection 3.4 and (2) we can derive

$$\exists A^*,S^*,S' \ [\text{happens}(A^*, S^*) \land \text{terminates}(A^*, S^*, \text{in}) \land \text{trans}(\text{park}, s0) < \text{trans}(A^*, S^*) \land \text{trans}(A^*,S^*) < S' \land \text{situation}(S')]$$

Using property (5) of *terminates*, we get the expected result, i.e.

$$\exists A^*,S^*,S' \ \{ \text{happens}(A^*, S^*) \land [A^*{=}\text{steal} \lor A^*{=}\text{tow}] \land \text{trans}(\text{park}, s0) < \text{trans}(A^*, S^*) \land \text{trans}(A^*,S^*) < S' \land \text{situation}(S') \}$$

The derivation in the situation calculus is more complicated and requires the use of induction. Using (1), SC, TYPE and LESS we can derive

$$\exists A,S,S' \{ S'{=} \text{result}(A, S) \land \text{happens}(A, S) \land \text{situation}(S) \land \text{trans}(\text{park}, s0) < S' \land \neg \text{initiates}(A, S, \text{in}) \land [\neg \text{holds}(\text{in}, S) \lor \text{terminates}(A, S, \text{in})] \}$$

Using property (5) of terminates, and distributing some of the disjunction over the conjunction, we get

$$\exists A,S,S' \{ [S'{=}\text{result}(A, S) \land \text{happens}(A, S) \land \text{trans}(\text{park}, s0) < S' \land [A{=}\text{steal} \lor A{=}\text{tow}]] \lor \text{Rest} \}$$

The disjunct Rest represents an unbounded number of further answers, which can be obtained in a similar manner. It is easy to see that, to obtain the single, more general conclusion of the event calculus, it is necessary to use induction.

8 Ramifications

8.1 Extending SC and EC

The predicate *holds(P, S)* can be understood as a metapredicate which expresses that the sentence named *P* holds explicitly (i.e. as an axiom) in the situation named *S*. A situation can be thought of either as a time point or as the set of all sentences, namely the theory, that holds at a given time point. Under this latter interpretation, *holds(P, S)* is a special case of the predicate *demo(S, P)* used to represent the proof predicate in metaprogramming (e.g. [8]).

In the general case *P* in *holds(P, S)* can name an arbitrary sentence. Such generality is necessary, for example, in applications to the formalisation of deductive database updates, the passing, amendment and repeal of legislation, and belief revision in intelligent agents. In practice, and for all the examples in this paper, it suffices for *P* to name a sentence in logic programming form. Standard formulations of the situation calculus either do not use a metapredicate at all or, if they do, limit themselves to cases where *P* names an atomic sentence or, at most, the negation of an atomic sentence.

The use of the metapredicate *holds(P, S)* already caters for the possibility that *P* might name an arbitrary sentence. The core situation calculus and event calculus axioms apply, without change, to this more general case. However, without additional axioms they do not completely characterise the set of all sentences that hold at any given time. They characterise, instead, only those sentences which hold explicitly (as axioms) at that time. To define the set of all sentences which hold explicitly or implicitly we need additional axioms such as:

demo(S, P) ← holds(P, S) D1
demo(S, P) ← demo(S, P← Q) ∧ demo(S, Q) D2
demo(S, P∧Q) ← demo(S, P) ∧ demo(S, Q) D3
demo(S, ¬P) ← ¬ demo(S, P) D4
demo(S, P) ← demo(S, ∀X P) D5

Here the metapredicate *demo(S, P)* expresses that the sentence named *P* can be demonstrated from the object level theory at situation S.

Clauses D2 and D3 constitute the familiar "vanilla meta-interpreter" which defines the proof predicate for propositional Horn clause theories. D4 can be understood as defining object level negation , ¬P, as failure to prove P. Clause D5 can be regarded as relating the so-called ground and non-ground representations of variables as discussed in [9].

D1-D5 exemplify the use of logic programming as a metalanguage to define its own proof predicate. Other clauses could be added to the definition to extend the object language beyond the logic programming case. D4, which interprets object level negation as negation as failure, could be replaced by clauses which define negation differently. For simplicity, and for the purposes of this paper, we can assume that the completion of D1-D5 is the definition of the *demo* predicate.

Although the core situation calculus and core event calculus axioms require no modification to deal with the initiation and termination of arbitrary sentences, the integrity constraint I1 needs to be generalised:

demo(S, P) ← happens(A, S) ∧ precondition(A, S, P) to take into account the fact that preconditions of events might be ramifications of other events. Similarly, some domain-specific assertions of the *initiates* and *terminates* predicates may have to be generalised. For example:

initiates(shoot, S, dead) ← demo(S, loaded)
terminates(shoot, S, alive) ← demo(S, loaded)

8.2 An Example from legislation

The following example illustrates the need to extend the *holds* predicate by means of *demo*. Here bna48 and bna83 name the British Nationality Acts of 1948 and 1983, respectively. For simplicity and naturalness, we use the uncompleted representation and we use time points instead of situations.

happens(bna48, 1948)
happens(bna83, 1983)
initiates(bna48, ∀X [citizen(X, uk) ← born(X, uk)])
terminates(bna83, ∀X [citizen(X, uk) ← born(X, uk)])

happens(accept-belief, 1941)
happens(reject-belief, 1967)
initiates(accept-belief, born(mary, uk))
terminates(reject-belief, born(mary, uk))

Assume a completion in which no event between bna48 and bna83 terminates the law that anyone born in the UK is a citizen of the UK, and in which no event between accept-belief and reject-belief terminates the belief that Mary was born in the UK. Then the original simplified event calculus axiom (introduced at the beginning of section 4) implies

holds(∀X [citizen(X, uk) ← born(X, uk)], T) ← 1948 < T < 1983
holds(born(mary, uk), T) ← 1941 < T < 1967

Together with D1, D2 and D5, these imply

demo(T, citizen(X, uk)) ← demo(T, born(X, uk)) ∧ 1948 < T < 1983 B1
demo(T, born(mary, uk)) ← 1941 < T < 1967 B2

Using transitivity of < and the facts that 1941<1948 and 1967<1983, these imply

demo(T, citizen(mary, uk)) ← 1948 < T < 1967

i.e. between 1948 and 1967 it is believed that Mary is a citizen of the UK.

A similar result would be obtained if we used situations instead of time points. We simply use four situations s1, s2, s3, s4 corresponding to the times just before each of 1941, 1948, 1967, 1983, respectively, and assert result(accept-belief, s1) < result(bna48, s2) < result(reject-belief, s3)< result(bna83, s4).

Using EC, LESS and transitivity of <, we would then be able to derive

demo(S, citizen(mary, uk)) ← situation(S) ∧ result(bna48, s2) ≤ S≤ s3

Standard treatments of the situation calculus, which restrict the predicate *P* in *holds(P, S)* to atoms or their negation, cannot represent this kind of reasoning. However, they can approximate it by using so-called state or domain constraints. In this example, domain constraints would be used to formulate B1 and B2 as domain-specific axioms in terms of *holds* rather than in terms of *demo*. Our treatment derives the analogues of domain constraints from more basic assumptions. Moreover, expressing B1 and B2 in terms of *demo* rather than *holds* enables us to complete the definition of *holds* without having to worry about interactions between the completion and domain constraints.

The simplified British Nationality Act example shows how the combination of events which initiate and terminate the holding of sentences together with the definition of the *demo* predicate deals automatically with certain kinds of ramifications. In this case, the belief that Mary is a citizen of the UK, which begins after 1948, is a ramification of passing bna48 and the earlier belief that Mary was born in the UK, initiated in 1941. The blocks world provides another example.

8.3 A blocks world example

In this example, apart from the event of *creation*, all other events affect only the location of objects. The property of an object being clear is a ramification of there being nothing on it. The act of *creation* itself can be thought of as initiating the sentences that state that an object is clear if it is not covered, and that it is covered if there is something on it. As before, the axioms for the domain specific *happens*, *initiates* and *terminates* predicates are common to both calculi:

happens(creation, s0)
initiates(creation, s0, ∀U [clear(U) ← ¬ covered(U)])
initiates(creation, s0, ∀U,V [covered(U) ← on(V,U)])

Assume a completion in which the two sentences, above, initiated by *creation* are never terminated, i.e. assume that

¬∃A,S { terminates(A, S, ∀U [clear(U) ← ¬ covered(U)]) }

¬∃A,S { terminates(A, S, ∀U,V [covered(U) ← on(V,U)]) }

are properties of the completion.

Using the event calculus axiom EC we can derive

$holds(\forall U\ [clear(U) \leftarrow \neg\ covered(U)], S) \leftarrow$
$\qquad trans(creation, s0) < S \wedge situation(S)$
$holds(\forall U,V\ [covered(U) \leftarrow on(V,U)], S) \leftarrow$
$\qquad trans(creation, s0) < S \wedge situation(S)$

These can also be derived using the situation calculus axiom SC. However, in this case the proof requires the use of induction.

In both calculi, to reason further with these clauses, we can use the definition of *demo* to derive

$demo(S, clear(U)) \leftarrow \neg\ demo(S, covered(U)) \wedge trans(creation, s0) < S \wedge situation(S)$
$demo(S, covered(U)) \leftarrow demo(S, on(V, U)) \wedge trans(creation, s0) < S \wedge situation(S)$

In standard treatments of the situation calculus, similar clauses, expressed in terms of *holds* rather than *demo*, would need to be given explicitly as state or domain constraints. In our case, they are derivable from more fundamental assumptions.

9 Conclusions and related work

To compare the situation calculus and event calculus, we have made minor modifications to both calculi, and formulated them as logic programs. Not only are the resulting calculi in some sense equivalent, but they both inherit the ability from standard formulations of the situation calculus to deal with hypothetical situations, and from the simplified event calculus to deal with actual states of affairs. For these latter two purposes, we have adapted the technique of [20, 21] for representing branching time.

However, our representation of time is less general than that of [20, 21], and is closer to that of [6], because, for the purposes of our comparison, we restrict ourselves to the special case where times are situations or transitions between situations.

We have seen that our derivation of the event calculus from the situation calculus requires proof by induction, whereas our derivation of the situation calculus from the event calculus does not. This suggests that the event calculus, even when restricted to times which are situations and transitions is more powerful than the situation calculus, as exemplified by our representation of the missing car example. We also believe that the more general case of the event calculus, similar to that studied by [20, 21], is more natural and more flexible than the restricted case. This was illustrated, for instance, in section 8.2 in the British Nationality Act example.

In this paper we have been more careful than in our earlier paper [11] to clarify our understanding of the iff-completions, induction and integrity constraints, in relation to the semantics of logic programs. Our current view is that the former should be understood as approximations to the intended semantics, in the same way that Peano axioms approximate truth in arithmetic. This point of view may be useful in understanding better the relationship between logic programming semantics and circumscription, which defines semantics directly in terms of truth in all minimal models. We believe that clarifying this relationship is an important direction for future research, and that the situation calculus and event calculus provide a promising context for such an investigation.

Our use of the iff-completion and induction yields a formalisation of the situation calculus which is closer to Reiter's formulation [16] than to other formulations using circumscription. However, one difference between his approach and ours is the treatment of state constraints and ramifications. Our formulation is based upon allowing the *holds* predicate to take arbitrary sentences as arguments and using the *demo* predicate to formalise reasoning with such sentences. We believe that investigating more closely the relationship between these and other approaches to state constraints and ramifications is a fruitful direction of research.

Acknowledgements

We are grateful to Ray Reiter, Murray Shanahan, Mark Denecker and Danny De Schreye for motivating and encouraging us to do this work, and to Mark Denecker, Danny De Schreye, Rob Miller, Marek Sergot, Murray Shanahan and Francesca Toni for helpful discussions. We would also like to thank the anonymous referees for their comments and suggestions for improving the paper. This research was supported by Fujitsu Laboratories.

References

1. Baker A.B., Nonmonotonic reasoning in the framework of the situation calculus, *Artificial Intelligence*, 49: page 5 (1991).

2. Charkravarthy U.S., Grant J. and Minker J., Logic-based approach to semantic query optimization, *ACM Transactions on Database Systems*, 15(2): 162-207 (1990) .

3. Clark K.L., Negation as failure, in: Gallaire H. and Minker J. (eds.), Logic and data bases, Plenum Press, 292-322, 1978.

4. Clark K.L. and Tarnlund S. A., A first-order theory of data and programs, Proceedings of IFIP, North Holland, 939-944, 1977.

5. Eshghi K., Abductive planning with event calculus, Proceedings of ICLP, MIT Press, Kowalski R.A. and Bowen K.A. (eds.), 562-579, 1988.

6. Kakas A. and Miller R., A simple declarative language for describing narratives with actions, *Journal of Logic Programming* (this issue) (1996).

7. Kautz H., The logic of persistence, Proceedings of AAAI, page 401, 1986.

8. Kowalski R.A., Logic for problem solving, Elsevier North Holland, 1979.

9. Kowalski R.A., Problems and promises of computational logic, in Computational logic, Symposium Proceedings, Lloyd J. (ed.), Springer, 80-95, 1990.

10. Kowalski R.A., Database updates in the event calculus, *Journal of Logic Programming* 12:121-146 (1992).

11. Kowalski R. and Sadri F., The situation calculus and event calculus compared, Proceedings of ILPS, Bruynooghe M. (ed.), MIT Press, 539-553, 1994.

12. Kowalski R.A. and Sergot M.J., A logic-based calculus of events, *New Generation Computing* 4:67-95 (1986).

13. Lloyd J. W. , Foundations of logic programming, Springer Verlag Symbolic Computation Series, 1987.

14. McCarthy J., Applications of circumscription to formalizing common sense knowledge, *Artificial Intelligence* 26:page 89 (1986).

15. Pinto J. and Reiter R., Temporal reasoning in logic programming, a case for the situation calculus, Proceedings of ICLP, MIT Press, Warren D.S. (ed.), 203-221, 1993.

16. Reiter R., Proving properties of states in the situation calculus, *Artificial Intelligence* 64(2): 337-351 (1993).

17. Sadri F. and Kowalski R., Variants of the event calculus, Proceedings of ICLP, MIT Press, Stirling L. (ed.), 67-82, 1995.

18. Shanahan M.P., Prediction is deduction but explanation is abduction, Proceedings of IJCAI, 1055-1060, 1989.

19. Shanahan M.P., Explanation in the situation calculus, Proceedings of IJCAI, 160-165, 1993.

20. Van Belleghem K., Denecker M. and De Schreye D., Combining situation calculus and event calculus, Proceedings of ICLP, MIT Press, Stirling L. (ed.), 83-97, 1995.

21. Van Belleghem K., Denecker M. and De Schreye D., On the relation between situation calculus and event calculus, *Journal of Logic Programming* (this issue) (1997).

22. Van Emden M. H. and Kowalski R. A., The semantics of predicate logic as a logic programming language, *J. ACM.* 23(4):733-742 (1976).