Combining Situation Calculus and Event Calculus

Kristof Van Belleghem, Marc Denecker, Danny De Schreye

Department of Computer Science, K.U.Leuven,
Celestijnenlaan 200A, B-3001 Heverlee, Belgium.
{kristof, marcd, dannyd}@cs.kuleuven.ac.be

Abstract

In this paper we study the differences between two logic theories for temporal reasoning, the Situation Calculus and the Event Calculus, and the implications of these differences. We construct a new formalism that combines the advantages of both Situation and Event Calculus and avoids the problems of either. The new formalism is useful for general temporal reasoning in worlds with discrete and continuous change, and enables representation of a wide range of hypothetical temporal reasoning problems. We show that both Situation and Event Calculus are instances of this new formalism.

1 Introduction

The Situation Calculus ([15]) and the Event Calculus ([14]) are two well-known formalisms for temporal reasoning. Although the original versions of both formalisms were created with clearly different goals in mind and did not look alike at all, later simplified versions of both started to show more and more similarities. In [18] an Event Calculus-like time line has been incorporated in the Situation Calculus. More recently, in [13] a thorough comparison of the frame axioms of Situation and Event Calculus was made, and the similarities between the formalisms were highlighted.

In this paper we compare Abductive Logic Programming formalisations of the two formalisms. We consider the similarities between the formalisms sufficiently indicated in [13], and will instead concentrate on the differences between them, hoping to find an explanation for the strong and weak points of either, and to ultimately construct a new, more expressive formalism that combines the advantages of Situation and Event Calculus while avoiding their shortcomings.

In section 2 we briefly describe Abductive Logic Programming, its semantics and its use for representing incomplete knowledge, and the notion of *abductive solution*. In the following two sections then, the basic forms of Situation and Event Calculus are described. In section 5 we analyse the differences between the two formalisms and the implications of these differences. The sixth section contains a proposal for combining Situation and Event Calculus into a more expressive framework, and an application illustrating the use of this formalism. In section 7 we show that Event Calculus and — with an appropriate definition of situations — Situation Calculus are instances of this formalism. We conclude with a discussion in section 8.

2 Abductive Logic Programming

An abductive logic program consists of a set of Horn clauses augmented with negation as failure P , a set of undefined predicates A , and a set of general first order logic constraints I .

The semantics of an abductive logic program is its Console completion semantics ([4]). Essentially, an abductive logic program is equivalent to the first order logic theory obtained by completing the definitions of all predicates in P that are not undefined (Clark completion, [3]), and adding the integrity constraints and Clark's equality axioms to the resulting theory.

In the sequel, we write clauses in the form $A \leftarrow B_1, \dots, B_n$. and constraints as first order logic formulae.

The clauses of an (abductive) logic program can be considered to define predicates (concepts) in terms of more primitive ones. Primitive predicates are simply defined by a set of facts. In an abductive logic program some of the primitive predicates can be undefined instead. Their truth value is not determined by the program, thus making it possible to represent incompletely known domains. In some cases however, there is at least partial domain knowledge which constrains the possible values of undefined predicates. This knowledge is typically represented by FOL axioms, the integrity constraints.

3 The Situation Calculus

In this section we present an Abductive Logic Programming formalisation of the Situation Calculus. The basic concepts of the calculus are *situations* and *actions*. A situation is defined as a period of time during which a certain set of properties hold. Actions are the cause of state transitions: if an action A occurs in a situation S , a new situation $result(A, S)$ results. $holds(P, S)$ means that property P is true in situation S . $initiates(A, S, P)$ and $terminates(A, S, P)$ denote that action A in situation S initiates or terminates the property P , i.e. immediately after A the property will be true (false).

The frame axioms of the Situation Calculus are:

$$\begin{aligned} holds(P, s_0) &\leftarrow initially(P). \\ holds(P, result(A, S)) &\leftarrow initiates(A, S, P). \\ holds(P, result(A, S)) &\leftarrow holds(P, S), not\ terminates(A, S, P). \end{aligned}$$

The above program can be read as follows under Console completion semantics: a property holds in the initial situation s_0 if it is initially true, and it holds in a later situation either if the action leading to that situation initiated the property, or if it already held in the previous situation and was not terminated by the most recent action. Otherwise, the property does not hold.

Formulations of the Situation Calculus in first order logic often use one predicate *abnormal* instead of *initiates* and *terminates*, not distinguishing possible positive changes in truth value from negative ones ([2], [15]). Since this distinction is always explicit in Event Calculus, we make it explicit in

Situation Calculus as well to facilitate comparison of the formalisms. Moreover, as indicated in [13], where this distinction is also explicitated, it is obvious that such explicitation results in a more detailed and therefore more “meaningful” theory.

It is important to note that many other versions of the Situation Calculus have been used in the litterature ([15], [8], [1], [20], [2], [18],). We have chosen a simple logic programming formulation which is very close to the one used in [13].

4 The Event Calculus

The Event Calculus has a world view which differs in some aspects from the Situation Calculus view. In Event Calculus there is one real line of time points. Properties can hold or not hold at a certain time point. The formula $holds(P, T)$ represents that P holds at time T . An event is the occurrence of an action at a certain point in time. The occurrence of an event E at time T is denoted by $happens(E, T)$. Such event can initiate or terminate properties, depending on the action associated with the event. $initiates(E, P)$ ($terminates(E, P)$) means that event E initiates (terminates) the property P . The atom $act(E, A)$ denotes that event E consists of an occurrence of action A . We get the following frame axioms:

$$\begin{aligned} holds(P, T) &\leftarrow happens(E_1, T_1), T_1 < T, initiates(E_1, P), \\ &\quad not\ clipped(T_1, P, T). \\ clipped(T_1, P, T) &\leftarrow happens(E_2, T_2), T_1 < T_2, T_2 < T, \\ &\quad terminates(E_2, P). \end{aligned}$$

The meaning of an Event Calculus program is given by the Console completion of a logic program which consists of the above clauses and a set of program clauses defining $initiates$, $terminates$, $happens$ and $<$. Alternatively, instead of giving a definition for $happens$ and $<$, these basic concepts may be left undefined. If new predicates are introduced in the definition of any of the above, the definitions of these new predicates are also completed or -if they are not completely known- they are left undefined. Any FOL axioms in the theory are added to the completed program.

Actually, in most cases $<$ will be undefined. Any information about $<$ is represented by FOL axioms. In Event Calculus, the order on time points must be a linear order, so the following axioms certainly need to be satisfied for all T_1 , T_2 and T_3 :

$$\begin{aligned} &\neg((T_1 < T_2) \& (T_2 < T_1)) \\ &((T_1 < T_2) \& (T_2 < T_3)) \Rightarrow (T_1 < T_3) \\ &(T_1 < T_2) \vee (T_2 < T_1) \vee (T_1 = T_2) \end{aligned}$$

It has been indicated before, for example in [17] and [6], that not restricting time to a linear order leads to certain problems. Partial orders often give rise to undesired models by allowing events to be unrelated in time. This is especially a problem when actions can have context-dependent effects. We will not discuss this point further here. Note, however, that the restriction to linear orders does

not imply that the order must be completely known. We only require that in each model every pair of different time points is ordered in one way or the other. In *which* way is not necessarily fixed by the theory.

Another requirement we impose is that no more than one event can occur at one point in time, and each event can of course only happen once:

$$(happens(E, T) \& happens(E*, T*)) \Rightarrow ((E = E*) \Leftrightarrow (T = T*))$$

At the beginning of time, say at t_0 , there is a *start* event initiating all properties that are initially true. This is represented by including the following clauses in the definition of the predicates *happens* and *initiates*:

$$\begin{aligned} happens(start, t_0). \\ initiates(start, P) \leftarrow initially(P). \end{aligned}$$

If *happens* is left undefined because it is not completely known, the *happens* fact above is replaced by a FOL axiom $happens(start, t_0)$. The new predicate *initially* which occurs in the definition of *initiates* can either be defined by a set of facts (which is then to be completed as part of the definition of *initiates*), or left undefined if it is not completely known.

Of course, no events are allowed to occur before t_0 :

$$happens(E, T) \Rightarrow ((t_0 < T) \vee (E = start))$$

Note that, where in Situation Calculus there is always an initial situation, in Event Calculus the initial event can actually be omitted, creating a time line which is unbounded in past and future. In most applications however, an initial “state of the world” is a very useful concept. To facilitate a comparison of the calculi, we will always introduce it in our theory.

Again, we must indicate that the version of the Event Calculus used here is not the only one. For one, the Event Calculus has changed considerably since its original presentation in [14], and a number of simplifications led to several formulations ([7], [21], [17], [12]) which are all slightly different. The above axioms are closest to those in [13].

5 A comparison of Situation and Event Calculus

It is clear that the Situation and Event Calculus share an important basic concept: that of properties initiated and terminated by actions. In [13] it was argued that with a number of rather small modifications to both theory and notation, the formalisms can be made essentially equivalent.

However, in fact the differences between the two formalisms are not unimportant and sometimes crucial, as we hope to demonstrate in this section.

We will illustrate the implications of these differences using the well-known temporal reasoning problem domain of the Russian Turkey Shooting Problem. The most important properties (or *fluents*) in this domain are *loaded*, indicating that a gun is loaded, and *alive*, indicating that a turkey is alive. Basic actions are *load* (with the effect that the gun becomes loaded), *wait* (which has no

effect), *shoot* (which unloads the gun and kills the turkey if the gun was loaded), and *spin* which has an indeterministic effect on the gun (it may or may not become unloaded).

In the Situation Calculus, the effects of these actions are represented by the clauses

$$\begin{aligned} \textit{initiates}(A, S, \textit{loaded}) &\leftarrow A = \textit{load}. \\ \textit{terminates}(A, S, \textit{loaded}) &\leftarrow A = \textit{shoot}. \\ \textit{terminates}(A, S, \textit{alive}) &\leftarrow A = \textit{shoot}, \textit{holds}(\textit{loaded}, S). \\ \textit{terminates}(A, S, \textit{loaded}) &\leftarrow A = \textit{spin}, \textit{goodluck}(S). \end{aligned}$$

where *goodluck*/1 is an undefined predicate. This immediately illustrates how indeterminism, and in particular actions with indeterministic effects, can be modeled in an abductive logic program: an undefined predicate is introduced representing the piece of missing information. The effect of an indeterministic action then depends on the actual truth value of this predicate, which is not determined by the program (though it may follow from additional constraints).

The Event Calculus representation of this domain knowledge is the following:

$$\begin{aligned} \textit{initiates}(E, \textit{loaded}) &\leftarrow \textit{act}(E, \textit{load}). \\ \textit{terminates}(E, \textit{loaded}) &\leftarrow \textit{act}(E, \textit{shoot}). \\ \textit{terminates}(E, \textit{alive}) &\leftarrow \textit{act}(E, \textit{shoot}), \textit{happens}(E, T), \textit{holds}(\textit{loaded}, T). \\ \textit{terminates}(E, \textit{loaded}) &\leftarrow \textit{act}(E, \textit{spin}), \textit{goodluck}(E). \end{aligned}$$

where *goodluck*/1 is again an undefined predicate. Note that in a specific scenario the theory will also contain a definition of *act*. Alternatively, for example in planning problems, *act* can be incompletely known and therefore undefined.

We will use some temporal reasoning scenarios in this problem domain to illustrate the differences between Situation and Event Calculus we are discussing in the remainder of this section.

5.1 Major Differences between the Calculi

Basically, both Situation and Event Calculus describe the truth values of fluents and the way these values change over time because of actions. However, three major differences can be distinguished in the approaches. These are:

- the use of branching time in Situation Calculus versus linear time in Event Calculus
- the explicit use of situations in the Situation Calculus. This introduces a notion of “previous state of the world” which is absent in Event Calculus. Related to this is the different formulation of the frame axiom in the two formalisms.
- the assumption that state transitions are functions, which is reflected in the special naming mechanism for situations in the Situation Calculus (names are uniquely determined by what has gone before). This assumption is absent in Event Calculus.

We illustrate the implications of these three differences for knowledge representation and problem solving below.

5.1.1 Branching Time

The importance of the first difference, the use of branching versus linear time, came to our attention during an attempt to provide a transformation from temporal domain descriptions in the $\mathcal{A}$ language ([9]) to Abductive Event Calculus theories. Where a transformation of $\mathcal{A}$ to Abductive Situation Calculus was established and proven sound and complete in [5], our proposed transformation to Event Calculus was incorrect. A detailed analysis pointed out that the problem could not be fixed by modifying the transformation, but was actually inherent to the Event Calculus, and in particular to its linear time structure.

The problem concerns the representation of hypothetical statements of the form “If A would have happened, then B would hold”. Such statements can be correctly represented and reasoned with in branching time theories like $\mathcal{A}$ and Situation Calculus. The intended conclusions of such statements are entailed by the theory, thus enabling powerful forms of hypothetical reasoning.

At first sight, this kind of hypothetical reasoning can be handled equally well using a form of abductive reasoning in a linear time theory. But however obvious this would seem, it is absolutely not true. The following example clarifies the point.

Assume we have the following information: initially there is a living turkey and a loaded gun. Then, a spin event (with indeterministic effect) and a shoot event occur. We also know that, *if* the spin event had been followed by waiting instead of shooting, the gun would still have been loaded. Question is: can the turkey be alive after the shooting ?

The answer is obviously no. Indeed, we know that after the spin event the gun would still have been loaded if we had waited instead of shooting. Therefore, the gun must have been loaded after spinning, which leads to the conclusion that shooting must have killed the turkey.

An abductive logic programming representation of this problem in the Situation Calculus (so, using branching time) looks as follows. The initial situation is described by the facts

$$\begin{aligned} & \textit{initially}(\textit{loaded}). \\ & \textit{initially}(\textit{alive}). \end{aligned}$$

and we represent the knowledge about the hypothetical situation (which is not determined by our knowledge of what has gone before - spinning has an indeterministic effect) by a FOL axiom

$$\textit{holds}(\textit{loaded}, \textit{result}(\textit{wait}, \textit{result}(\textit{spin}, s_0)))$$

Note that this axiom, though expressed in terms of the *holds* predicate, actually constrains the possible values of *goodluck*, the only predicate which is not completely defined in terms of others. Indeed, the abductive logic program has two models: one in which *goodluck*(s_0) is true and one in which it is false. In the former model, we can derive $\neg \textit{holds}(\textit{loaded}, \textit{result}(\textit{spin}, s_0))$,

and therefore $\neg \text{holds}(\text{loaded}, \text{result}(\text{wait}, \text{result}(\text{spin}, s_0)))$, from the program clauses. This is inconsistent with the above axiom, so this model of the program clauses is not a model of our theory. The only valid model is the one with $\text{goodluck}(s_0) = \text{false}$. In that model, we find $\text{holds}(\text{loaded}, \text{result}(\text{spin}, s_0))$, from which we can deduce, using the clauses for termination of properties, that

$$\neg \text{holds}(\text{alive}, \text{result}(\text{shoot}, \text{result}(\text{spin}, s_0)))$$

This is indeed the intended result.

In Event Calculus (linear time), the representation which springs to mind is the following: since we want to reason about hypothetical events, we assume incomplete knowledge on the events that happen and their order. Therefore we declare *happens* and *act* as well as $<$ to be undefined predicates. This ensures that for every hypothetical sequence of events, there are models in which this sequence occurs. If we want to represent knowledge about a hypothetical situation, we do this in a way very similar to that in Situation Calculus, by an axiom of the form “if this sequence of events happens, then this formula will hold afterward”. To simplify our notation, we first define a new predicate *int_events*/2. *int_events*(T, T^*) indicates that one or more intermediate events occur between T and T^* .

$$\text{int_events}(T, T^*) \leftarrow \text{happens}(E', T'), (T < T'), (T' < T^*).$$

In this particular example, the initial situation is given by the facts

$$\text{initially}(\text{loaded}). \quad \text{initially}(\text{alive}).$$

Our knowledge about the hypothetical evolution of the world above can be represented by the following axiom:

$$\begin{aligned} & (\text{happens}(e_1, t_1) \ \& \ \text{act}(e_1, \text{spin}) \ \& \ \neg \text{int_events}(t_0, t_1) \ \& \\ & \text{happens}(e_3, t_3) \ \& \ \text{act}(e_3, \text{wait}) \ \& \ t_1 < t_3 \ \& \ \neg \text{int_events}(t_1, t_3) \ \& \\ & \Rightarrow \forall T_4 : (t_3 < T_4 \ \& \ \neg \text{int_events}(t_3, T_4)) \Rightarrow \text{holds}(\text{loaded}, T_4) \end{aligned}$$

And then we can express our query as

$$\begin{aligned} & (\text{happens}(e_1, t_1) \ \& \ \text{act}(e_1, \text{spin}) \ \& \ \text{happens}(e_2, t_2) \ \& \ \text{act}(e_2, \text{shoot}) \ \& \\ & \neg \text{int_events}(t_0, t_1) \ \& \ t_1 < t_2 \ \& \ \neg \text{int_events}(t_1, t_2) \ \& \) \\ & \Rightarrow \forall T : (t_2 < T \ \& \ \neg \text{int_events}(t_2, T)) \Rightarrow \text{holds}(\text{alive}, T) \end{aligned}$$

Sadly enough, this is not a correct representation. Indeed, due to the linear time constraint, the lefthandside of the axiom and the lefthandside of the query can never evaluate to true in the same model. Either e_2 can follow e_1 without intermediate events, or e_3 can, but not both. Therefore, our axiom has no influence at all on any useful answers to our query (those answers in which the query’s lefthandside is true), where it should have provided us with extra information.

This boils down to the following: for each sequence of events, our theory has at least one model in which that sequence occurs. However, within *one* model,

there can only exist *one* sequence of events. In general, combining information about two unrelated sequences of events (for example a real and a hypothetical one in this example) is not possible *within* the formalism, because one sequence always excludes the other.

On the other hand, it should be mentioned that using meta-reasoning on the formalism we can indeed reach the desired conclusions. This is due to the fact that meta-reasoning allows one to consider different object-level models in one meta-level model. As a result, if we restrict time to a linear order we can still perform hypothetical reasoning by reasoning on the formalism, but we can not model hypothetical reasoning within the formalism.

Using a branching time line as in Situation Calculus, on the other hand, we find all hypothetical sequences (branches) of events in one model. This allows us to combine information about these branches, and therefore to perform hypothetical reasoning within the formalism. We will show in section 6 how the explicit assertion of hypothetical events in a branching time formalism based on Event Calculus yields a similar solution.

We can conclude that it is not sufficient that there exist models for each hypothetical situation: we need all different hypothetical situations in the same model if we want to reason on them and on how they relate to each other within the formalism. That is why branching time is necessary for hypothetical reasoning, and the use of abductive reasoning in a linear time theory is not an adequate alternative.

5.1.2 Situations and the Frame Axiom

A second difference between Situation and Event Calculus is the formulation of the frame axiom. In Event Calculus something holds if it was initiated at an earlier time point and not terminated between that time point and the current one. In Situation Calculus something holds if it was initiated by the last action, or if it held in the previous situation and was not terminated by the last action. Under certain conditions these statements are just two equivalent formulations of the same axiom, as shown in [13].

But these conditions are not always satisfied. For example, the Situation Calculus axiom assumes the existence of a notion of “previous state of the world”. Time is in a sense considered a row of blocks (situations) instead of a continuous line. All change is discrete and happens instantaneously between two situations. Within one situation the world does not change: situations are atomic.

In the Event Calculus time can be a continuous line. The period between two actions is not an atomic entity, it is still a line of time points where each time point has an independent existence.

On the other hand, if we look at the Event Calculus frame axiom, we see that only events can initiate or terminate properties, so also in Event Calculus all time points between two consecutive events are equivalent in the sense that the same properties hold at all these time points.

Yet in this case, this “discreteness” is due to an axiom we are using and not to a basic concept, which makes it easier to remove it. For example, if we

want to model any type of continuous change (the classical example of filling water tanks springs to mind), in the Event Calculus we only need to add an extra axiom describing such change, without modifying the existing axioms. Extensions of the Event Calculus to model continuous change have for example been proposed in [22] and [23].

As shown in [19], it is possible to extend the Situation Calculus to deal with continuous change as well, though the extension requires more substantial modifications to the formalism than in Event Calculus. The introduction of an Event Calculus-like time line as in [18] and [19] is a necessary first step. In [19] properties are then divided into two classes: *discrete fluents* and *continuous parameters*. The former behave like properties in the classical Situation Calculus, the latter have a more complex behaviour and can for example change their value continuously within one situation.

Circumstances in which the different formulation of the frame axiom is important do occur sometimes in temporal reasoning applications even when a discrete time structure is present. This is for example the case if *derived properties* are allowed, in which case we are confronted with a form of the *ramification problem* ([10; 11]). Derived properties are properties defined in terms of others by rules like

$$holds(dead, T) \leftarrow not\ holds(alive, T).$$

Such rules are added to the definition of *holds* in the theory before completing that definition.

This kind of rules interacts incorrectly with the Situation Calculus frame axiom, as indicated among others in [16] and [12]. The problem is that the frame axiom, which describes *holds* dependencies between states, is not compatible with rules that describe dependencies within one state. The Situation Calculus frame axiom implicitly assumes that all fluents are independent, which is not always true.

As shown in [12], in Event Calculus the frame axiom does not lead to such problems. There, the frame axiom is completely orthogonal to other dependency rules, and does not assume their absence. This allows for the use of such dependency rules, which often lead to an elegant representation of a domain, though we would like to stress that such rules must be used with great care and in a consistent way. A derived property should for example never be explicitly initiated or terminated, it may only depend on the value of other, non-derived, properties.

Of course, there are other ways to avoid the problems with the Situation Calculus axiom, the most obvious one being not allowing derived properties, or finding another way of representing them. In [12] an explicit division of properties into primitive and derived ones is proposed as a solution. In [20] domain dependent *explanation closure axioms* are applied to (among others) solve this form of the ramification problem. In [2] a special form of circumscription is used to solve the ramification problem in certain, but not all, cases.

It is clear that the problems with the Situation Calculus axiom can be overcome in several ways. The advantage of the Event Calculus axiom is that

it is better suited to deal with at least a large class of ramifications in a correct way, without other additions or modifications to the framework. Moreover, it does not assume the existence of a notion of previous state of the world.

5.1.3 State Transitions as Functions

The final important difference we see between the Situation and Event Calculus is reflected in the naming mechanism used for situations. A situation's name consists of the sequence of actions leading to that situation. At first sight this naming mechanism has no influence on the theory, but in fact it is based on certain assumptions about state transitions that do not always hold.

To illustrate this, let us look at the Situation Calculus axiom for state transitions.

$$\begin{aligned} \text{holds}(P, \text{result}(A, S)) &\leftarrow \text{initiates}(A, S, P). \\ \text{holds}(P, \text{result}(A, S)) &\leftarrow \text{holds}(P, S), \text{not terminates}(A, S, P). \end{aligned}$$

Transforming this axiom from functor representation into atom representation, we find it is equivalent to

$$\begin{aligned} \text{holds}(P, S^*) &\leftarrow \text{res}(A, S, S^*), \text{initiates}(A, S, P). \\ \text{holds}(P, S^*) &\leftarrow \text{res}(A, S, S^*), \text{holds}(P, S), \text{not terminates}(A, S, P). \end{aligned}$$

with the following definition for *res*/3:

$$\text{res}(A, S, \text{result}(A, S)).$$

In this definition, the underlying assumption becomes apparent: *res* is defined for all *A* and *S*, and for each *A* and *S* there is exactly one $S^* = \text{result}(A, S)$.

This implies two things. On one hand, in every situation, every possible action happens and leads to a specific new situation. The naming mechanism used for situations implicitly asserts the occurrence of each possible sequence of actions in each model of our theory. This is of course a very useful consequence of the naming mechanism: it eliminates the need to decide or represent which hypothetical sequences of actions occur. Moreover, stating that each sequence of actions is one hypothetical evolution of the world is a very natural point of view.

On the other hand, we see that each action in each situation leads to exactly one new situation (*res* is a function). Therefore, each sequence of actions can have only one hypothetical outcome. Now, while in a deterministic world the same sequence of actions would always lead to the same state of the world, this is no longer true if we allow for actions with indeterministic effects, like the spin action in our previous examples. After spinning the chamber of a loaded gun, there are two possible resulting states: one where the gun is loaded and one where it is unloaded. It is impossible to represent these two states as different ones in the Situation Calculus, due to the definition of *res* as a function.

One solution would be to include (an) extra parameter(s) with each indeterministic action in the definition of *res*. This parameter would then determine which of a number of indeterministic outcomes the action has in this particular

case. For example, with spinning as the indeterministic action, the goodluck parameter could be included, leading to the following clauses in the definition of *res*:

$$\begin{aligned} &res(spin, S, result(spin(goodluck = true), S)). \\ &res(spin, S, result(spin(goodluck = false), S)). \end{aligned}$$

However, this solution requires a specific choice of parameters for each indeterministic action. It is not clear to us if the parameterized approach can lead to a solution in all cases.

In any case, the restriction of state transitions to functions, as imposed by the classical Situation Calculus naming mechanism, is not always desirable, and the use of names containing no information (like those of Event Calculus events) helps to avoid the problem.

6 The new Calculus

6.1 Formalisation

Having studied the differences between Situation and Event Calculus, we feel an attempt to combine their strong points is appropriate.

As a basic concept, we prefer not to use situations but rather a continuous time line, which should give us more flexibility and expressivity. As a frame axiom we take the more general version used in Event Calculus. *event*(*E*, *T*) indicates the occurrence of event *E* at the hypothetical time point *T*. The distinction between *event*/2 and the *happens*/2 predicate of section 4 is related to the branching time we introduce below. *holds*(*P*, *T*) means *P* holds at (hypothetical) *T*.

$$\begin{aligned} holds(P, T) &\leftarrow event(E_1, T_1), T_1 < T, initiates(E_1, P), \\ &\quad not\ clipped(T_1, P, T). \\ clipped(T_1, P, T) &\leftarrow event(E_2, T_2), T_1 < T_2, T_2 < T, \\ &\quad terminates(E_2, P). \end{aligned}$$

To make hypothetical reasoning possible, we choose a branching time model. The time structure then basically forms a tree, with one actual time line and a number of non-actual, hypothetical time lines branching away from it. This is ensured by the following constraints on *<*:

$$\begin{aligned} &\neg((T_1 < T_2) \& (T_2 < T_1)) \\ &((T_1 < T_2) \& (T_2 < T_3)) \Rightarrow (T_1 < T_3) \\ &((T_1 < T_3) \& (T_2 < T_3)) \Rightarrow (T_1 < T_2) \vee (T_2 < T_1) \vee (T_1 = T_2) \end{aligned}$$

We still assume events correspond to exactly one time point and time points to at most one event, though this last restriction could be eliminated.

$$(event(E, T) \& event(E*, T*)) \Rightarrow ((E = E*) \Leftrightarrow (T = T*))$$

No special naming mechanism for events is used, leaving open the possibility that the same sequence of actions leads to several distinct situations. This will allow us to handle a larger class of hypothetical reasoning problems.

To handle problems with an initial situation, we again introduce an event at the beginning of time by including the following clauses in the definitions of *event* and *initiates*.

$$\begin{aligned} & \text{event}(\text{start}, t_0). \\ & \text{initiates}(\text{start}, P) \leftarrow \text{initially}(P). \end{aligned}$$

and adding the FOL axiom

$$\text{event}(E, T) \Rightarrow ((t_0 < T) \vee (E = \text{start}))$$

In certain applications, most notably in natural language processing, the distinction between real and hypothetical actions and situations may become important. Therefore, among the many hypothetical branches in our model, we will mark one branch as being the *real* time line. *real*(*T*) is used to denote that *T* is a real time point. The initial time point *t*₀ is always real, and real time points of course form one line, as expressed by the additional axioms

$$\begin{aligned} & \text{real}(t_0). \\ & (\text{real}(X) \& \text{real}(Y)) \Rightarrow ((X < Y) \vee (Y < X) \vee (X = Y)) \end{aligned}$$

The meaning of our theory is given by the following FOL theory: we take all clauses in this section, add to them definitions in the form of clauses for *initiates* (partially given in terms of *initially* above), *terminates*, *event*, *<* and *real* (and any predicates occurring in these clauses), and complete the resulting program using Console Completion. Again, a predicate may be declared undefined and FOL axioms may be added to the resulting theory. If *event* and/or *real* are undefined, the above partial definitions for them are replaced by FOL axioms.

6.2 Application

The following application illustrates how the new formalism can be used for hypothetical reasoning. We again take the example from section 5.1.1. which could not be modeled in Abductive Event Calculus.

We leave *<* undefined, rather restricting it by constraints than giving a complete enumeration. Of course, *goodluck* is also undefined. The real scenario is asserted by the facts

$$\begin{aligned} & \text{initially}(\text{loaded}). \quad \text{event}(e_1, t_1). \quad \text{act}(e_1, \text{spin}). \\ & \text{initially}(\text{alive}). \quad \text{event}(e_2, t_2). \quad \text{act}(e_2, \text{shoot}). \end{aligned}$$

and the FOL axiom

$$t_1 < t_2$$

The hypothetical wait event is then also asserted, indicating that the hypothetical situation we want to talk about exists:

$$\text{event}(e_3, t_3). \quad \text{act}(e_3, \text{wait}).$$

The axioms

$$\begin{aligned} & t_1 < t_3 \\ & \neg(t_2 < t_3) \& \neg(t_3 < t_2) \end{aligned}$$

are added, indicating that t_3 is after t_1 but in a different branch of time than t_2 . For the sake of simplicity, we assume complete knowledge of all events and actions. Also, we don't make the distinction between real and hypothetical time points explicit (all time points are hypothetical).

Observations about hypothetical situations are in general represented by FOL axioms. We again define the predicate *int_events/2* to simplify notation:

$$int_events(T, T^*) \leftarrow event(E', T'), (T < T'), (T' < T^*).$$

The FOL axiom in this case is

$$\forall T_4 : ((t_3 < T_4) \& \neg int_events(t_3, T_4) \Rightarrow holds(loaded, T_4))$$

As we can easily check, this axiom is only satisfied in those models where *goodluck*(e_1) is false. In all those models, *holds*(*loaded*, t_2) is true, which entails *terminates*(e_2 , *alive*). This leads to the desired conclusion that

$$\forall T : ((t_2 < T) \& \neg int_events(t_2, T) \Rightarrow \neg holds(alive, T))$$

holds in all models that satisfy the FOL axioms.

We conclude that the above form of hypothetical reasoning is possible in the new formalism, where it is not in Event Calculus. The addition of branching time is responsible for this.

On the other hand, continuous change and a number of other problem cases for the Situation Calculus can be represented in exactly the same way as they can in the Event Calculus. A discussion of continuous change at this point would lead us too far, but we refer the reader to for example [22] or [23]. The formalism described in the latter paper does not suffer from an extension to branching time, though the one in [22], due to its use of a numerical time line, might not be so easily adapted to branching time applications.

Finally, there are those cases where hypothetical reasoning is required in the presence of indeterministic actions. As indicated in section 5.1.3, such problems can not be represented in standard Situation Calculus, and certainly not in Event Calculus. In the new formalism however, it is obvious that representing two different branches that consist of exactly the same sequence of actions, but with different effects, is not a problem. There is no naming mechanism making this impossible, nor is there a constraint prohibiting it.

7 Relation of the Calculus to the original Calculi

The new calculus combines important aspects of the Situation and Event Calculus to create a more expressive framework. In this section, we show that both Situation and Event Calculus are actually instances of the new formalism.

The proof that Event Calculus is an instance of the new formalism is straightforward. If we replace the predicate *event/2* in the new calculus by *happens/2*, omit the notion *real/1*, and replace the axiom

$$\forall T_1, T_2, T_3 : ((T_1 < T_3) \& (T_2 < T_3)) \Rightarrow (T_1 < T_2) \vee (T_2 < T_1) \vee (T_1 = T_2)$$

by the stronger

$$\forall T_1, T_2 : (T_1 < T_2) \vee (T_2 < T_1) \vee (T_1 = T_2)$$

we simply end up with the Event Calculus.

With an appropriate definition of situations, we can show that the Situation Calculus as well is an instance of the new formalism. To this end, we add axioms to our theory representing the restrictions imposed by Situation Calculus, and we introduce a notion of situation. We define when a property holds in a situation, and show that this definition coincides with the one given by the Situation Calculus frame axiom.

We first introduce an axiom in our formalism representing the assumption in Situation Calculus that in each “situation” each action occurs and leads to exactly one new situation. In terms of events, this means that for each event E and for each action A , there is exactly one event E^* consisting of the occurrence of A after E and such that there are no events between E and E^* . We assume that a domain dependent type predicate *action/1* exists, with the following restrictions:

$$\begin{aligned} \exists A : \text{action}(A) \\ \forall E, A : \text{act}(E, A) \Rightarrow \text{action}(A) \end{aligned}$$

In other words, only actions defined as such can occur, and there exists at least one action. Our axiom can then be written as follows:

$$\forall E, T, A : ((\text{event}(E, T) \ \& \ \text{action}(A)) \Rightarrow \exists E^*, T^* : \quad (ia) \\ [\text{event}(E^*, T^*) \ \& \ (T < T^*) \ \& \ \text{act}(E^*, A) \ \& \ \neg \text{int_events}(T, T^*)])$$

$$\begin{aligned} \forall E, T, E', T', E^*, T^*, A : ((\text{event}(E, T) \ \& \ \text{action}(A) \\ \ \& \ \text{event}(E', T') \ \& \ \text{act}(E', A) \ \& \ T < T' \ \& \ \neg \text{int_events}(T, T') \quad (ib) \\ \ \& \ \text{event}(E^*, T^*) \ \& \ \text{act}(E^*, A) \ \& \ T < T^* \ \& \ \neg \text{int_events}(T, T^*) \\ \Rightarrow (E' = E^*) \ \& \ (T' = T^*)) \end{aligned}$$

Now, we introduce the concept of situation in our calculus. Situations are sets of time points defined inductively as follows:

$$\begin{aligned} s_0 &= \{T \mid (t_0 < T) \ \& \ \neg \text{int_events}(t_0, T)\} \\ \text{result}(A, S) &= \{T \mid \exists E', T' : T' \in S \ \& \ \text{event}(E', T') \ \& \ \text{act}(E', A) \\ &\quad \& \ (T' < T) \ \& \ \neg \text{int_events}(T', T)\} \end{aligned}$$

We express this in the following axioms:

$$\begin{aligned} \text{member}(T, s_0) &\Leftrightarrow ((t_0 < T) \ \& \ \neg \text{int_events}(t_0, T)) \\ \text{member}(T, \text{result}(A, S)) &\Leftrightarrow \exists E', T' : (\text{event}(E', T') \ \& \ \text{member}(T', S) \ \& \\ &\quad \text{act}(E', A) \ \& \ (T' < T) \ \& \ \neg \text{int_events}(T', T)) \end{aligned}$$

We again choose the term representation for situations instead of using a predicate *res*. This simplification is possible because of the above axioms, which ensure that a situation is uniquely determined by its preceding situation and the action responsible for the last state transition. We define a type predicate for situations as follows:

$$\begin{aligned} \text{situation}(S) &\Leftrightarrow (S = s_0 \vee \exists A, S', T, E : [\text{situation}(S') \ \& \ \text{action}(A) \\ &\quad \& \ S = \text{result}(A, S') \ \& \ \text{member}(T, S') \ \& \ \text{event}(E, T) \ \& \ \text{act}(E, A)]) \end{aligned}$$

We say that a property holds in a situation if and only if it holds at all time points belonging to that situation. To avoid confusion with *holds* for time points, we use the predicate *holds_in/2* to express the truth value of properties in a situation.

$$\text{holds_in}(P, S) \Leftrightarrow \forall T : (\text{member}(T, S) \Rightarrow \text{holds}(P, T))$$

Now, we define predicates *init_s/3* and *term_s/3* for situations, based on the predicates *initiates* and *terminates*:

$$\begin{aligned} \forall A, S, P : (\text{init_s}(A, S, P) &\Leftrightarrow \exists E, T : \\ &(\text{member}(T, S) \& \text{event}(E, T) \& \text{act}(E, A) \& \text{initiates}(E, P))) \\ \forall A, S, P : (\text{term_s}(A, S, P) &\Leftrightarrow \exists E, T : \\ &(\text{member}(T, S) \& \text{event}(E, T) \& \text{act}(E, A) \& \text{terminates}(E, P))) \end{aligned}$$

Next, assume a predicate *holds'/2* is defined by the Situation Calculus frame axiom, where the definitions of *initiates/3* and *terminates/3* coincide with *init_s/3* and *term_s/3*:

$$\begin{aligned} \text{holds}'(P, s_0) &\leftarrow \text{initially}(P). \\ \text{holds}'(P, \text{result}(A, S)) &\leftarrow \text{init_s}(A, S, P). \\ \text{holds}'(P, \text{result}(A, S)) &\leftarrow \text{holds}(P, S), \text{not term_s}(A, S, P). \end{aligned}$$

Assume then *T* is the abductive logic program consisting of the clauses and FOL axioms of both this and the previous section, with undefined *initiates*, *terminates*, *event*, *action*, *act* and *<* predicates.

Also assume we have an induction axiom *I* on situations

$$\begin{aligned} \forall \Phi : [(\forall S : (\text{situation}(S) \Rightarrow \Phi(S))) &\Leftarrow \\ (\Phi(s_0) \& \forall A, S : [(\Phi(S) \& \text{action}(A) \& \text{situation}(S)) &\Rightarrow \Phi(\text{result}(A, S))])] \end{aligned}$$

Theorem 1

$$T \cup I \models \forall P, S : \text{situation}(S) \Rightarrow (\text{holds_in}(P, S) \Leftrightarrow \text{holds}'(P, S))$$

The proof is included in the appendix for the referees.

8 Discussion

In this paper, we have studied the main differences between the Situation Calculus and the Event Calculus. Earlier comparative studies have been presented in [18] and [13].

In the former, an Event Calculus-like time line is embedded in the Situation Calculus, running through one branch of situations (the real branch). It is shown that this formalism is more expressive than an early version of the Event Calculus.

In the latter, simplified versions of Event Calculus and Situation Calculus are compared. In Event Calculus, situations are introduced and the time

relation is instantiated to be the transitive closure of the *result* function. In Situation Calculus, explicit event assertions are added, eliminating the branching time aspect through constraints on the occurrence of events. Also, an induction axiom is added to the Situation Calculus. An equivalence between the two resulting formalisms is then proven, thus showing the great similarities between the original calculi.

Now that a certain equivalence between both formalisms has been proven, in this paper we found the next step was to concentrate on the differences between them and their implications for temporal representation and reasoning. Studying these differences, we have then constructed a more expressive general framework, which combines the stronger points of both calculi.

In the new formalism, hypothetical reasoning is possible just like in Situation Calculus, where the linear time of Event Calculus inhibits this. The notion of “previous state of the world” from Situation Calculus is absent, resulting in a more general frame axiom which is better suited for dealing with ramifications, as well as for modeling continuous change. Finally, the absence of a strong naming mechanism allows for the representation of a wider range of hypothetical reasoning problems than can be handled by Situation Calculus.

There are still some open problems, the most important one being that in this new formalism hypothetical events need to be somehow asserted in the theory. Which events need assertion moreover depends on the hypothetical situations we have information on and the queries that need to be answered. One possibility is indeed asserting events depending on whether they are needed for the problem at hand or not, as we have done in this paper. Another possibility is the addition of an axiom like axiom (ia) from the previous section, which asserts the occurrence of each action in each situation. (Actually, an even stronger axiom is needed: in each situation there must exist an event corresponding to each possible outcome of each (indeterministic) action.)

In fact, if we add to the new calculus the axioms that for every possible sequence of actions there is exactly one branch in our time tree, and if we then explicitly introduce situations as periods of time between events, we end up with a formalism that is equivalent to the Situation Calculus. Similarly, if we restrict time to one line instead of a tree, we obtain the Event Calculus. This shows that both original formalisms are actually instances of this new formalism.

Acknowledgements

Kristof Van Belleghem is partly supported by ESPRIT BR project Compulog II and partly by the Belgian IWT. Marc Denecker is supported by Dienst Onderzoekscoördinatie, K.U.Leuven. Danny De Schreye is a senior research associate of the Belgian NFWO.

References

- [1] K. Apt and M. Bezem. Acyclic programs. In *Proc. of the International*

Conference on Logic Programming, pages 579–597. MIT press, 1990.

- [2] A. Baker. Nonmonotonic Reasoning in the Framework of the Situation Calculus. *Artificial Intelligence*, 49:5–23, 1991.
- [3] K. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and databases*, pages 293–322. Plenum Press, 1978.
- [4] L. Console, D. Theseider Dupre, and P. Torasso. On the relationship between abduction and deduction. *Journal of Logic and Computation*, 1(5):661–690, 1991.
- [5] M. Denecker and D. De Schreye. Representing incomplete knowledge in abductive logic programming. *to appear in: Journal of Logic and Computation*, 1994.
- [6] M. Denecker, L. Missiaen, and M. Bruynooghe. Temporal reasoning with abductive event calculus. In *Proceedings of ECAI 92, Vienna*, 1992.
- [7] K. Eshghi. Abductive planning with event calculus. In R. Kowalski and K. Bowen, editors, *Proceedings of the 5th ICLP*, 1988.
- [8] C. Evans. Negation as failure as an approach to the Hanks and McDermott problem. In *Proceedings of the second International Symposium on Artificial Intelligence*, 1989.
- [9] M. Gelfond and V. Lifschitz. Describing Action and Change by Logic Programs. In *Proc. of the 9th Int. Joint Conf. and Symp. on Logic Programming*, 1992.
- [10] S. Hanks and D. McDermott. Default reasoning, nonmonotonic logic, and the frame problem. In *Proceedings of the National Conference on Artificial Intelligence, Philadelphia*, pages 328–333, 1986.
- [11] S. Hanks and D. McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.
- [12] R. A. Kowalski. Database updates in the event calculus. *Journal of Logic Programming, 1992*, 1992.
- [13] R. A. Kowalski. The situation calculus and event calculus compared. In M. Bruynooghe, editor, *Proceedings of ILPS 1994*, 1994.
- [14] R. A. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(4):319–340, 1986.
- [15] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.
- [16] L. Missiaen. Situation calculus and event calculus in modal logic Z. Technical Report 85, Department of Computer Science, K.U.Leuven, 1989.

- [17] L. Missiaen. *Localized abductive planning with the event calculus*. PhD thesis, Department of Computer Science, K.U.Leuven, 1991.
- [18] J. Pinto and R. Reiter. Temporal Reasoning in Logic Programming: A Case for the Situation Calculus. In *Proc. of the International Conference on Logic Programming*, pages 203–221, 1993.
- [19] J. A. Pinto. Temporal reasoning in the situation calculus. Technical Report KRR-TR-94-1, Computer Science Department, University of Toronto, 1994.
- [20] L. Schubert. Monotonic solution of the frame problem in the situation calculus. In H.E. Kyburg, Jr et al., editor, *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer Academic Publishers, 1990.
- [21] M. Shanahan. Prediction is deduction but explanation is abduction. In *Proceedings of IJCAI 89*, page 1055, 1989.
- [22] M. Shanahan. Representing continuous change in the event calculus. In *Proceedings of the 9th ECAI*, page 598, 1990.
- [23] K. Van Belleghem, M. Denecker, and D. De Schreye. Representing continuous change in the abductive event calculus. In *Proc. of the International Conference on Logic Programming*, 1994.

Appendix: Proof of Theorem 1

We first prove some important lemmas. First we need a lemma that in each situation, each action occurs and leads to exactly one new situation.

Lemma 1

$$\forall S, A : (situation(S) \& action(A)) \Rightarrow \\ \exists E, T : member(T, S) \& event(E, T) \& act(E, A)$$

(Lemma 1 a)

$$\forall S, A, E, T, E', T' : ((situation(S) \& member(T, S) \& event(E, T) \\ \& act(E, A) \& member(T', S) \& event(E', T') \& act(E', A)) \\ \Rightarrow (E = E' \& T = T'))$$

(Lemma 1 b)

This is easily proven given axioms (ia) and (ib). We prove part a first: for all S and A , if $situation(S) \& action(A)$ then

$$action(A) \& ((S = s_0) \vee [\exists B, S^* : (S = result(B, S^*) \& situation(S^*) \\ \& action(B) \& \exists E^*, T^* : (member(T^*, S) \& event(E^*, T^*) \& act(E^*, B))]))$$

With the fact $event(start, t_0)$ and axiom (ia) this implies

$$[S = s_0 \& event(start, t_0) \& \\ \exists E, T : (event(E, T) \& t_0 < T \& act(E, A) \& \neg int_events(t_0, T))] \\ \vee \\ [\exists E, T, E^*, T^*, B, S^* : (S = result(B, S^*) \& action(B) \& situation(S^*) \\ \& member(T^*, S) \& event(E^*, T^*) \& act(E^*, B) \\ \& event(E, T) \& T^* < T \& act(E, A) \& \neg int_events(T^*, T))]$$

which, using the definition of *member*, leads to

$$\exists E, T : member(T, s_0) \& S = s_0 \& event(E, T) \& act(E, A) \\ \vee \\ \exists E, T, B, S^* : [member(T, result(B, S^*)) \& event(E, T) \& act(E, A) \\ \& S = result(B, S^*)]$$

and therefore

$$\exists E, T : member(T, S) \& event(E, T) \& act(E, A)$$

which is what we needed to prove.

The proof of part b is as follows: given S, A, E, T, E', T' , assume

$$(situation(S) \& member(T, S) \& event(E, T) \& act(E, A) \\ \& member(T', S) \& event(E', T') \& act(E', A))$$

Again, we use the definition of *situation* to obtain

$$[S = s_0 \vee \exists B, S^*, E^*, T^* : (S = \text{result}(B, S^*) \& \text{action}(B) \& \text{situation}(S^*) \\ \& \text{member}(T^*, S^*) \& \text{event}(E^*, T^*) \& \text{act}(E^*, B)) \\ \& \text{member}(T, S) \& \text{event}(E, T) \& \text{act}(E, A) \\ \& \text{member}(T', S) \& \text{event}(E', T') \& \text{act}(E', A))]$$

which can be rewritten as

$$[S = s_0 \& \text{member}(T, S) \& \text{event}(E, T) \& \text{act}(E, A) \\ \& \text{member}(T', S) \& \text{event}(E', T') \& \text{act}(E', A)] \\ \vee \\ [\exists B, S^*, E^*, T^* : (S = \text{result}(B, S^*) \& \text{action}(B) \& \text{situation}(S^*) \\ \& \text{member}(T^*, S^*) \& \text{event}(E^*, T^*) \& \text{act}(E^*, B)) \\ \& \text{member}(T, S) \& \text{event}(E, T) \& \text{act}(E, A) \\ \& \text{member}(T', S) \& \text{event}(E', T') \& \text{act}(E', A))]$$

and using the definition of *member* we get

$$[S = s_0 \& t_0 < T \& \neg \text{int_events}(t_0, T) \& \text{event}(E, T) \& \text{act}(E, A) \& t_0 < T' \\ \& \neg \text{int_events}(t_0, T') \& \text{event}(E', T') \& \text{act}(E', A) \& \text{event}(\text{start}, t_0)] \\ \vee \\ [\exists B, S^*, E^*, T^* : (S = \text{result}(B, S^*) \& \text{action}(B) \& \text{situation}(S^*) \\ \& \text{member}(T^*, S^*) \& \text{event}(E^*, T^*) \& \text{act}(E^*, B)) \\ \& T^* < T \& \neg \text{int_events}(T^*, T) \& \text{event}(E, T) \& \text{act}(E, A) \\ \& T^* < T' \& \neg \text{int_events}(T^*, T') \& \text{event}(E', T') \& \text{act}(E', A))]$$

Applying axiom (*ib*) to either part of the disjunction, we find

$$E = E' \& T = T'$$

which proves the lemma. $\square$

Another lemma which is needed in our proof is the following: if a property holds at one time point belonging to a situation, it holds at all time points in that situation, and vice versa:

Lemma 2

$$\forall S : \text{situation}(S) \Rightarrow \\ ((\exists T : \text{member}(T, S) \& \text{holds}(P, T)) \Leftrightarrow [\forall T : \text{member}(T, S) \Rightarrow \text{holds}(P, T)])$$

The proof of the “ $\Leftarrow$ ” part is straightforward: all that needs to be proven for all S is $\exists T : \text{member}(T, S)$, which follows immediately from lemma 1 a and the axiom that at least one action exists.

To prove the $\Rightarrow$ part, assume S is a situation. We know that

$$\exists T : \text{member}(T, S) \& \text{holds}(P, T)$$

and need to prove $\forall T : \text{member}(T, S) \Rightarrow \text{holds}(P, T)$. To this end, we rewrite the above formula as

$$\exists T, E', T' : \text{member}(T, S) \& \text{event}(E', T') \& T' < T \& \text{initiates}(E', P) \\ \& \neg \text{clipped}(T', P, T)$$

where $\text{clipped}(T', P, T)$ is used (from now on) as a shorter notation for

$$\exists E^*, T^* : \text{event}(E^*, T^*) \& T' < T^* \& T^* < T \& \text{terminates}(E^*, P)$$

Using the definition of situation, we now get

$$\begin{aligned} & [\exists T, E', T' : (S = s_0 \& \text{member}(T, s_0) \& \text{event}(E', T') \& T' < T \& \\ & \quad \text{initiates}(E', P) \& \neg \text{clipped}(T', P, T))] \\ & \vee \\ & [\exists T, E', T', B, S^*, E^*, T^* : (S = \text{result}(B, S^*) \& \text{member}(T, \text{result}(B, S^*)) \\ & \quad \& \text{situation}(S^*) \& \text{member}(T^*, S^*) \& \text{event}(E^*, T^*) \& \text{act}(E^*, B) \\ & \quad \& \text{event}(E', T') \& T' < T \& \text{initiates}(E', P) \& \neg \text{clipped}(T', P, T))] \end{aligned}$$

In the first part of the disjunction, we can derive from $\text{member}(T, s_0)$ that $t_0 < T$ and $\neg \text{int_events}(t_0, T)$. Since $T' < T$ and $t_0 < T$ hold, one of $T' < t_0$, $t_0 < T'$ or $t_0 = T'$ must be true. The first formula is inconsistent with the axiom that t_0 is the first time point, the second formula together with $T' < T$ is inconsistent with $\neg \text{int_events}(t_0, T)$. Therefore $t_0 = T'$, and we can derive from the above disjunction that

$$\begin{aligned} & [\exists T, E', T' : (S = s_0 \& T' = t_0 \& E' = \text{start} \& t_0 < T \& \neg \text{int_events}(t_0, T) \\ & \quad \& \text{event}(\text{start}, t_0) \& \text{initiates}(\text{start}, P) \& \neg \text{clipped}(t_0, P, T))] \\ & \vee \\ & [\exists T, E', T', E^*, T^* : (S = \text{result}(B, S^*) \& T^* < T \& \neg \text{int_events}(T^*, T) \& \\ & \quad \text{situation}(S^*) \& \text{member}(T^*, S^*) \& \text{event}(E^*, T^*) \& \text{act}(E^*, B) \& \\ & \quad \text{event}(E', T') \& T' < T \& \text{initiates}(E', P) \& \neg \text{clipped}(T', P, T))] \end{aligned}$$

It is clear that a conjunctive formula implies any other formula obtained by omitting conjuncts. We use this knowledge here and later in the appendix to simplify formulae by eliminating conjuncts irrelevant for the proof. At this point, the first disjunct is simplified in this manner. In the second disjunct, we know that $T' < T$ and $T^* < T$. Therefore, either $T' < T^*$ or $T' = T^*$ must hold ($T^* < T'$ can not hold because of $\neg \text{int_events}(T^*, T)$). So the above disjunction implies

$$\begin{aligned} & [S = s_0 \& \text{initiates}(\text{start}, P)] \\ & \vee \\ & [\exists T, E', T', E^*, T^* : (S = \text{result}(B, S^*) \& T^* < T \& \text{member}(T^*, S^*) \\ & \quad \& \neg \text{int_events}(T^*, T) \& \text{event}(E^*, T^*) \& \text{act}(E^*, B) \& \\ & \quad \text{event}(E', T') \& T' < T^* \& \text{initiates}(E', P) \& \neg \text{clipped}(T', P, T))] \\ & \vee \\ & [\exists T, E', T' : (S = \text{result}(B, S^*) \& T' < T \& \neg \text{int_events}(T', T) \& \\ & \quad \text{member}(T', S^*) \& \text{event}(E', T') \& \text{act}(E', B) \& \text{initiates}(E', P) \& \\ & \quad \neg \text{clipped}(T', P, T))] \end{aligned}$$

Now, what we must prove is that for each member of S , P holds. So assume $member(T^+, S)$. We must prove $holds(P, T^+)$. Using the disjunction above, we find for each T^+

$$\begin{aligned}
& [t_0 < T^+ \ \& \ \neg int_events(t_0, T^+) \ \& \ event(start, t_0) \ \& \ initiates(start, P)] \\
& \quad \vee \\
& \quad [\exists T, E', T', E^*, T^*, E'', T'' : (event(E'', T'') \ \& \ member(T'', S^*) \ \& \\
& \quad \quad act(E'', B) \ \& \ T'' < T^+ \ \& \ \neg int_events(T'', T^+) \ \& \ T^* < T \\
& \quad \& \ \neg int_events(T^*, T) \ \& \ member(T^*, S^*) \ \& \ event(E^*, T^*) \ \& \ act(E^*, B) \ \& \\
& \quad \quad event(E', T') \ \& \ T' < T^* \ \& \ initiates(E', P) \ \& \ \neg clipped(T', P, T))] \\
& \quad \vee \\
& \quad [\exists T, E', T', E'', T'' : (event(E'', T'') \ \& \ member(T'', S^*) \ \& \ act(E'', B) \ \& \\
& \quad \quad T'' < T^+ \ \& \ \neg int_events(T'', T^+) \ \& \ T' < T \ \& \ \neg int_events(T', T) \ \& \\
& \quad \quad member(T', S^*) \ \& \ event(E', T') \ \& \ act(E', B) \ \& \ initiates(E', P) \ \& \\
& \quad \quad \neg clipped(T', P, T))]
\end{aligned}$$

Now, we can apply lemma 1 b to the second and third disjunct and derive that

$$\begin{aligned}
& [t_0 < T^+ \ \& \ \neg int_events(t_0, T^+) \ \& \ event(start, t_0) \ \& \ initiates(start, P)] \\
& \quad \vee \\
& \quad [\exists E', T', E'', T'' : (event(E'', T'') \ \& \ member(T'', S^*) \ \& \ act(E'', B) \ \& \\
& \quad \quad T'' < T^+ \ \& \ \neg int_events(T'', T^+) \ \& \ T'' < T \ \& \ \neg int_events(T'', T) \ \& \\
& \quad \quad event(E', T') \ \& \ T' < T'' \ \& \ initiates(E', P) \ \& \ \neg clipped(T', P, T))] \\
& \quad \vee \\
& \quad [\exists T, E', T' : (event(E', T') \ \& \ member(T', S^*) \ \& \ act(E', B) \ \& \\
& \quad \quad T' < T^+ \ \& \ \neg int_events(T', T^+) \ \& \ T' < T \ \& \ \neg int_events(T', T) \ \& \\
& \quad \quad initiates(E', P) \ \& \ \neg clipped(T', P, T))]
\end{aligned}$$

In the second disjunct now, looking at the definition of *clipped/3* we see that $\neg clipped(T', P, T)$ together with $T' < T''$, $T'' < T$ and $event(E'', T'')$ implies the formula

$$\neg clipped(T', P, T'') \ \& \ \neg terminates(E'', P) \ \& \ \neg clipped(T'', P, T)$$

This formula together with $\neg int_events(T'', P, T^+)$ implies $\neg clipped(T', P, T^+)$. So from the disjunction above it follows that

$$\begin{aligned}
& [t_0 < T^+ \ \& \ \neg int_events(t_0, T^+) \ \& \ event(start, t_0) \ \& \ initiates(start, P)] \\
& \quad \vee \\
& \quad [\exists E', T', E'', T'' : (event(E'', T'') \ \& \ member(T'', S^*) \ \& \ act(E'', B) \ \& \\
& \quad \quad T'' < T^+ \ \& \ event(E', T') \ \& \ T' < T'' \ \& \ initiates(E', P) \ \& \\
& \quad \quad \neg clipped(T', P, T^+))] \\
& \quad \vee \\
& \quad [\exists T, E', T' : (event(E', T') \ \& \ member(T', S^*) \ \& \ act(E', B) \ \& \\
& \quad \quad T' < T^+ \ \& \ \neg int_events(T', T^+) \ \& \ T' < T \ \& \ initiates(E', P))]
\end{aligned}$$

Which leads, in all three disjuncts, to the conclusion that

$$holds(P, T^+)$$

which proves the lemma.

□

Proof of Theorem 1

Since we have the induction axiom I on situations available, proving the theorem can be reduced to a proof that our theory entails the following formulae:

$$\forall P : \text{holds}'(P, s_0) \Leftrightarrow \text{holds_in}(P, s_0)$$

$$\forall P, A, S : (\text{situation}(S) \ \& \ \text{action}(A)) \Rightarrow [(\text{holds}'(P, S) \Leftrightarrow \text{holds_in}(P, S)) \Rightarrow (\text{holds}'(P, \text{result}(A, S)) \Leftrightarrow \text{holds_in}(P, \text{result}(A, S)))]$$

The first formula can be proven as follows: using the definition of holds_in , we write $\text{holds_in}(P, s_0)$ as

$$\forall T : [(t_0 < T \ \& \ \neg \text{int_events}(t_0, T)) \Rightarrow \text{holds}(P, T)]$$

and using the definition of holds this is equivalent to

$$\begin{aligned} & \forall T : [(t_0 < T \ \& \ \neg \text{int_events}(t_0, T)) \Rightarrow \\ & \exists E', T' : (\text{event}(E', T') \ \& \ T' < T \ \& \ \text{initiates}(E', P) \ \& \ \neg \text{clipped}(T', P, T))] \end{aligned}$$

Because of the precondition $t_0 < T \ \& \ \neg \text{int_events}(t_0, T)$ and the axiom that t_0 is the first time point, we know that $T' = t_0$. The above formula then reads

$$\begin{aligned} & \forall T : [(t_0 < T \ \& \ \neg \text{int_events}(t_0, T)) \Rightarrow \\ & (\text{event}(\text{start}, t_0) \ \& \ t_0 < T \ \& \ \text{initiates}(\text{start}, P) \ \& \ \neg \text{clipped}(t_0, P, T))] \end{aligned}$$

Provided that $\exists T : (t_0 < T \ \& \ \neg \text{int_events}(t_0, T))$, which follows immediately from axiom (ia), this formula implies $\text{initiates}(\text{start}, P)$. On the other hand, $\text{initiates}(\text{start}, P)$ implies the above formula, as $\neg \text{clipped}(t_0, P, T)$ trivially follows from $\neg \text{int_events}(t_0, T)$.

So we find that $\text{holds_in}(P, s_0) \Leftrightarrow \text{initiates}(\text{start}, P)$. Of course, we also know that $\text{initiates}(\text{start}, P)$ is equivalent to $\text{initially}(P)$ and therefore to $\text{holds}'(P, s_0)$. This completes the proof of the first formula needed for the theorem.

Next we need to prove for all A, P and S that, given $\text{situation}(S)$ and $\text{action}(A)$, and given $\text{holds}'(P, S) \Leftrightarrow \text{holds_in}(P, S)$, it follows that

$$\text{holds}'(P, \text{result}(A, S)) \Leftrightarrow \text{holds_in}(P, \text{result}(A, S))$$

To this end, we rewrite $\text{holds}'(P, \text{result}(A, S))$ and $\text{holds_in}(P, \text{result}(A, S))$ first. $\text{holds}'(P, \text{result}(A, S))$ is equivalent to (using its definition)

$$[\text{holds}'(P, S) \ \& \ \neg \text{term_s}(A, S, P)] \vee \text{init_s}(A, S, P)$$

which is the same as (using the induction hypothesis)

$$[\text{holds_in}(P, S) \ \& \ \neg \text{term_s}(A, S, P)] \vee \text{init_s}(A, S, P)$$

and using the definitions of holds_in , term_s and init_s , this is equivalent to

$$\begin{aligned} & [\forall T' : (\text{member}(T', S) \Rightarrow \text{holds}(P, T')) \ \& \ \neg \exists E^*, T^* : \\ & (\text{member}(T^*, S) \ \& \ \text{event}(E^*, T^*) \ \& \ \text{act}(E^*, A) \ \& \ \text{terminates}(E^*, P))] \\ & \vee \\ & \exists E^+, T^+ : (\text{member}(T^+, S) \ \& \ \text{event}(E^+, T^+) \ \& \ \text{act}(E^+, A) \ \& \\ & \text{initiates}(E^+, P)) \end{aligned}$$

We will call this formula $\mathcal{F}$.

On the other hand, $holds_in(P, result(A, S))$ is equivalent to

$$\forall T : member(T, result(A, S)) \Rightarrow holds(P, T)$$

This formula will be called $\mathcal{G}$.

We need to prove that $\mathcal{F} \Leftrightarrow \mathcal{G}$. First we prove that $\mathcal{F} \Rightarrow \mathcal{G}$. Assume $\mathcal{F}$ is given. We can write $member(T, result(A, S))$ as

$$\begin{aligned} \exists E', T' : (event(E', T') \& T' < T \& act(E', A) \& member(T', S) \\ \& \neg int_events(T', T)) \end{aligned}$$

If we add this formula to both disjuncts of $\mathcal{F}$ we can derive

$$\begin{aligned} & [\exists E', T' : (event(E', T') \& T' < T \& act(E', A) \& member(T', S) \& \\ & \neg int_events(T', T) \& holds(P, T') \& \neg terminates(E', P))] \\ & \vee \\ & [\exists E', T', E'', T'' : (event(E', T') \& T' < T \& act(E', A) \& \\ & member(T', S) \& \neg int_events(T', T) \& event(E'', T'') \& member(T'', S) \& \\ & act(E'', A) \& initiates(E'', P))] \end{aligned}$$

Applying lemma 1 b to the second disjunct, this turns into

$$\begin{aligned} & [\exists E', T' : (event(E', T') \& T' < T \& act(E', A) \& member(T', S) \& \\ & \neg int_events(T', T) \& holds(P, T') \& \neg terminates(E', P))] \\ & \vee \\ & [\exists E', T', E'', T'' : (event(E', T') \& T' < T \& act(E', A) \& member(T', S) \& \\ & \neg int_events(T', T) \& initiates(E', P))] \end{aligned}$$

Then we use the definition of $holds$, and derive $\neg clipped(T', P, T)$ from the stronger $\neg int_events(T', T)$ in both disjuncts:

$$\begin{aligned} & [\exists E', T', E'', T'' : (event(E', T') \& T' < T \& act(E', A) \& member(T', S) \& \\ & \neg clipped(T', P, T) \& \neg terminates(E', P) \& event(E'', T'') \& T'' < T' \& \\ & initiates(E'', P) \& \neg clipped(T'', P, T'))] \\ & \vee \\ & [\exists E', T', E'', T'' : (event(E', T') \& T' < T \& act(E', A) \& \\ & \neg clipped(T', P, T) \& initiates(E', P))] \end{aligned}$$

Now, from

$$\neg clipped(T', P, T) \& event(E', T') \& \neg terminates(E', P) \& \neg clipped(T'', P, T')$$

and the time relations, we can derive $\neg clipped(T'', P, T)$, which leads to the conclusion that in both disjuncts we can apply the definition of $holds$ to get

$$holds(P, T)$$

which proves $\mathcal{G}$.

Now the only thing left for us to prove is that $\mathcal{G} \Rightarrow \mathcal{F}$. This can be proven as follows: $\mathcal{G}$ is, because of lemma 2, equivalent to

$$\exists T : \text{member}(T, \text{result}(A, S)) \& \text{holds}(P, T)$$

or, using the definitions again

$$\begin{aligned} \exists T, E', T', E'', T'' : (T'' < T \& \text{event}(E'', T'') \& \text{initiates}(E'', P) \& \\ \neg \text{clipped}(T'', P, T) \& T' < T \& \text{event}(E', T') \& \text{act}(E', A) \& \\ \text{member}(T', S) \& \neg \text{int_events}(T', T)) \end{aligned}$$

Here we see $T'' < T$ and $T' < T$, which implies either $T'' < T'$ or $T'' = T'$ ($T' < T''$ is impossible because of $\neg \text{int_events}(T', T)$). So we find

$$\begin{aligned} [\exists T, E', T', E'', T'' : (T'' < T' \& \text{event}(E'', T'') \& \text{initiates}(E'', P) \& \\ \neg \text{clipped}(T'', P, T) \& T' < T \& \text{event}(E', T') \& \text{act}(E', A) \& \\ \text{member}(T', S) \& \neg \text{int_events}(T', T))] \\ \vee \\ [\exists T, E', T' : (\text{event}(E', T') \& \text{initiates}(E', P) \& \neg \text{clipped}(T', P, T) \& \\ T' < T \& \text{act}(E', A) \& \text{member}(T', S) \& \neg \text{int_events}(T', T))] \end{aligned}$$

From the first disjunct, we can derive on one hand

$$\begin{aligned} \exists T, E', T', E'', T'' : (T'' < T' \& \text{event}(E'', T'') \& \text{initiates}(E'', P) \& \\ T' < T \& \text{member}(T', S) \& \\ \neg \exists E^*, T^* : [\text{event}(E^*, T^*) \& T'' < T^* \& T^* < T \& \text{terminates}(E^*, P)]) \end{aligned}$$

and on the other hand

$$\begin{aligned} \exists T, E', T', E'', T'' : (T'' < T' \& \text{event}(E', T') \& \text{act}(E', A) \& \\ T' < T \& \text{member}(T', S) \& \\ \neg \exists E^*, T^* : [\text{event}(E^*, T^*) \& T'' < T^* \& T^* < T \& \text{terminates}(E^*, P)]) \end{aligned}$$

and we can simplify the second disjunct. In this way, we find the disjunction above implies the formula

$$\begin{aligned} [\exists T, E', T', E'', T'' : (T'' < T' \& \text{event}(E'', T'') \& \text{initiates}(E'', P) \& \\ T' < T \& \text{member}(T', S) \& \\ \neg \exists E^*, T^* : [\text{event}(E^*, T^*) \& T'' < T^* \& T^* < T \& \text{terminates}(E^*, P)])] \\ \& \\ \exists T, E', T', E'', T'' : (T'' < T' \& \text{event}(E', T') \& \text{act}(E', A) \& \\ T' < T \& \text{member}(T', S) \& \\ \neg \exists E^*, T^* : [\text{event}(E^*, T^*) \& T'' < T^* \& T^* < T \& \text{terminates}(E^*, P)])] \end{aligned}$$

$$\vee$$

$$\exists T, E', T' : (\text{event}(E', T') \& \text{initiates}(E', P) \& \text{act}(E', A) \& \text{member}(T', S))$$

which in its turn implies

$$\begin{aligned} [\exists T', E'', T'' : (T'' < T' \& \text{event}(E'', T'') \& \text{initiates}(E'', P) \& \text{member}(T', S) \\ \& \neg \exists E^*, T^* : [\text{event}(E^*, T^*) \& T'' < T^* \& T^* < T' \& \text{terminates}(E^*, P)])] \\ \& \\ \exists E', T' : (\text{event}(E', T') \& \text{act}(E', A) \& \text{member}(T', S) \& \neg \text{terminates}(E', P))] \\ \vee \\ [\exists E', T' : (\text{event}(E', T') \& \text{initiates}(E', P) \& \text{act}(E', A) \& \text{member}(T', S))] \end{aligned}$$

Using the definition of *holds* and lemma 1 b on the first disjunct, we find

$$\begin{aligned}
& [\exists T' : (member(T', S) \& holds(P, T')) \\
& \quad \& \\
& \quad \forall E', T' : ((event(E', T') \& act(E', A) \& member(T', S)) \Rightarrow \\
& \quad \quad \neg terminates(E', P))] \\
& \quad \vee \\
& [\exists E', T' : (event(E', T') \& initiates(E', P) \& act(E', A) \& member(T', S))]
\end{aligned}$$

Finally, we use lemma 2 and some rewriting on the first disjunct to obtain

$$\begin{aligned}
& [\forall T' : (member(T', S) \Rightarrow holds(P, T')) \\
& \quad \& \\
& \quad \neg \exists E', T' : (event(E', T') \& act(E', A) \& member(T', S) \& terminates(E', P))] \\
& \quad \vee \\
& [\exists E', T' : (event(E', T') \& initiates(E', P) \& act(E', A) \& member(T', S))]
\end{aligned}$$

which is equivalent to $\mathcal{F}$.

This then completes the proof that in our theory T , for all A, P and S , $situation(S) \& action(A) \& (holds'(P, S) \Leftrightarrow holds_in(P, S))$, implies

$$holds'(P, result(A, S)) \Leftrightarrow holds_in(P, result(A, S))$$

Together with $holds'(P, s_0) \Leftrightarrow holds_in(P, s_0)$ and the induction axiom, this proves the theorem that $holds'/2$ and $holds_in/2$ are equivalent.

□