Occurrences and Narratives as Constraints in the Branching Structure of the Situation Calculus.

Javier A. Pinto*
Departamento de Ciencia de la Computación
Escuela de Ingeniería
Pontificia Universidad Católica de Chile
jpinto@ing.puc.cl
URL = "http://lyrcc.ing.puc.cl/jpinto/"

Abstract

The Situation Calculus is a logic of time and change in which there is a distinguished initial situation, and all other situations arise from the different sequences of actions that might be performed starting in the initial one. Within this framework, it is difficult to incorporate the notion of an *occurrence*, since all situations after the initial one are *hypothetical*. These *occurrences* are important, for instance, when one wants to represent *narratives*. There have been proposals to incorporate the notion of an *action occurrence* in the language of the Situation Calculus, namely Miller and Shanahan's work on narratives [25] and Pinto and Reiter's work on *actual line*s of situations [30]. Both approaches have in common the idea of incorporating a linear sequence of situations into the tree described by theories written in the Situation Calculus language. Unfortunately, several advantages of the Situation Calculus are lost when reasoning with a narrative line or with an actual line of occurrences.

In this paper we propose a different approach to dealing with action occurrences and narratives, which can be seen as a generalization of *narrative lines* to *narrative trees*. In this approach we exploit the fact that, in the discrete Situation Calculus [16], each situation has a unique history, and interpret occurrences as constraints on valid histories. We argue that this new approach subsumes the *linear* approaches of Miller and Shanahan's, and Pinto and Reiter's. In this framework, we are able to represent various kinds of occurrences; namely, conditional, preventable and non-preventable occurrences. Other types of occurrences, not discussed in this article, can also be accommodated.

1 Introduction

The Situation Calculus [21] is one of the standard logical languages used in Artificial Intelligence to write theories of action and change. The original language is based on an ontology in which actions are instantaneous and non-concurrent. This language has been the focus of much interest in the Knowledge Representation community, and has motivated a great deal of research (e.g., [6, 10, 3, 17, 25, 26]). The Situation Calculus is a *branching time* temporal logic. The structure of situations in the Situation Calculus embodies all possible courses of action that can be taken

*This work has been partly funded by the Fondecyt project number 195-0885.

starting from an initial situation. This view leads to considering every non–initial situation as being *hypothetical*, in the sense that the situation *would* arise only if the actions that lead to it were actually executed. An important advantage of this approach, over linear time approaches, is that one can reason about these different possible alternatives, and this reasoning can be done within the language. Indeed, in the Situation Calculus we have situations that would arise after performing a sequence of actions $a_1, a_2, \ldots, a_n$, along with other situations that would arise after performing totally different sequences of actions $a'_1, a'_2, \ldots, a'_m$ (all starting, possibly, in the same initial situation). All these situations are available at the object level, which is not possible in linear time logics. This feature allows one to express knowledge and perform queries of a hypothetical nature. For instance, it makes it possible to define planning as a deductive problem, as suggested by Green [8].

On the other side of the coin, linear time temporal logics, such as the Event Calculus [11, 39], Allen's Temporal Logic [1] and other logics of time, have the advantage that one can very easily specify, partially or completely, the way in which the world evolves. Thus, one can explicitly state the events that are known to occur in a given setting. On the other hand, the branching time Situation Calculus is ultimately hypothetical. Thus, it provides no tools to state knowledge about the way in which the world actually evolves. In particular, in the Situation Calculus approach, it is difficult to represent the notion of an occurrence, which leads to difficulties in representing:

Narratives: A possibly incomplete specification of actions or events that are known to occur at specific time points.

Triggers: The specification of actions or events that are triggered by the occurrence of other actions or events[1].

The goal of this work is to add to the Situation Calculus the expressiveness required to represent occurrences and narratives and be able to reason hypothetically with them.

The recognition of the aforementioned limitations, has led to an increasing interest in bridging the expressive gap that exists between linear time and branching time temporal logics. This has been accomplished in several settings. A common idea has been to integrate both views of time (linear and branching) within a single formalism.

In the Situation Calculus, there have been, to our knowledge, two approaches to extend the logical language to perform this integration. In [25], Miller and Shanahan introduce a time line in the Situation Calculus, where narratives are described in terms of this time line. Additionally, they associate one situation with each action in the narrative. Also, they introduce a mapping `Sit` from time points to situations, such that each time point (in the positive real time line) is associated with a unique situation. A different, but related approach, is taken by Pinto and Reiter [30]. In this second approach, one path in the tree, which starts at S_0, is selected. This path is called the *actual* path of situations and is meant to identify a path in the tree that represents what really happens in the world. In [28], a circumscription based formalization is used to select preferred interpretations for the *actual* line. Each situation in this path is assigned a time point. There is a clear correspondence between the two approaches. Indeed, the mapping `Sit` of Miller and Shanahan's can be equated with the *actual* situations of Pinto and Reiter's. Furthermore, Shanahan has shown that, under suitable conditions, both approaches yield the same results [40]. In these two approaches, circumscription axioms are used in order to formalize the assumption that no events occur unless they must occur. As Reiter has observed [36], this leads to the *premature minimization problem*, discussed later.

[1] As mentioned later in the article, triggers can also be regarded to be relationships between circumstances and actions. Thus, actions can be triggered by contextual conditions, rather than by other actions.

There have also been research contributions that are based on the Event Calculus. In this approach, the underlying linear time structure of the Event Calculus is enriched with a branching structure. For instance, Provetti, in [32], proposes to add a branching structure to the Event Calculus whose axiomatization is inspired by the Situation Calculus. The extended language allows one to perform hypothetical reasoning in a framework based on the Event Calculus. Along similar lines, another proposal was advanced in [41] by Van Belleghem, Denecker, and De Schreye, where an attempt is made to combine the strong points of the Situation Calculus and the Event Calculus. Instead of relying upon a situation structure, they introduce a tree structure for continuous time, with one actual time line and a number of non-actual hypothetical time lines. An interesting difference between this approach and the ones mentioned before, is that the branching structure of time is continuous.

In a related line of research, Kowalski and Sadri [10] study the relationship between the Situation Calculus and the Event Calculus within a logic programming framework. The semantics of the logic programs is represented by *if and only if* completions augmented by axioms of induction. They present a logic programming formulation for both, the Situation Calculus and the Event Calculus. In order to perform a formal comparison of the two languages, they consider special cases of both. On the one hand they restrict the Event Calculus to eliminate concurrency and, on the other, they linearize the view of time in the Situation Calculus. A striking result is that both formulations are equivalent (modulo induction). Thus, the logic programming formulation of the Situation Calculus with induction implies the formulation of the Event Calculus, and the formulation of the Event Calculus implies the formulation of the Situation Calculus. The linearization of the view of time in the formulation of the Situation Calculus is complemented with the proposal that, as with the event calculus [4], hypothetical reasoning be performed abductively.

Another influential line of research has emerged from the $\mathcal{A}$ family of *action description languages*. In [5], Gelfond and Lifschitz introduced the action description language $\mathcal{A}$, which was designed to express causal laws. These laws specify the effects of actions on fluents describing the state of the world. The semantics of the language models hypothetical reasoning in a manner similar to that of the Situation Calculus. This language has been extended in different ways; for instance, to allow for non-determinism, concurrency, etc. In [2], Baral, Gelfond and Provetti presented the language $\mathcal{L}_1$, which is a modification of the language $\mathcal{A}$. The main goal was to present a languagee, in the same spirit of language $\mathcal{A}$, in which one could express knowledge about actual action occurrnces and observations of truth values of fluents in actual situations. Therefore, $\mathcal{L}_1$ can be seen as a development that parallels the work mentioned above. Indeed, the semantics of $\mathcal{L}_1$ sanctions the existence of a single *actual* line of situations. $\mathcal{L}_1$ supports hypothetical reasoning by allowing one to express queries regarding hypothetical situations. These situations result from hypothetically performing actions starting at some situation along the actual line.

A thorough comparison between the approaches is justified and deserves a separate research contribution. As Baral, Gelfond and Provetti[2], we also believe that all these approaches should be developed further. In particular, if one recognizes that an essential goal of this research area is to gain a *full understanding of the space of all possible formalisms for reasoning about action*[2].

We continue on this pursuit by proposing an alternative interpretation for narratives within the Situation Calculus. Unfortunately, approaches based on the notion of the *actual* or *narrative* line have certain undesirable properties. For instance, let us assume that an agent's behavior is specified using the Situation Calculus with an actual line of situations. In such setting, the specification describes the way in which the world actually evolves. As in linear time theories, it is not possible to hypothesize about the different courses of action that the agent might have

[2] As expressed by Miller and Shanahan in [25].

taken, and that are not actual. Such hypothetical reasoning has to be performed at a meta-level. For instance, assume that one wants to state an occurrence constraint of the form "if the door bell rings, I will open the door". In the actual line approach, this constraint would be expressed as a constraint on the actions that appear in the actual line. Thus, it would leave the hypothetical situations unconstrained. In fact, there will be hypothetical lines in which the door bell rings, and I don't open the door.

Up until now, a narrative was interpreted as describing a distinguished time line, which is considered *actual* or *real*. Instead, we propose to view narratives as constraints on the tree structure of time. Thus, a narrative shall identify a subtree that is composed of those branches that are compatible with the narrative. We consider this approach to be a generalization of the idea of incorporating a time line into the Situation Calculus tree structure. Narratives become *dynamic constraints* that invalidate developments that are inconsistent with them.

In our current approach to modeling occurrences within the Situation Calculus we make use of a *legal* predicate for situations. This predicate was introduced by Reiter in order to constrain the valid situations. The set of legal situations forms a tree rooted at S_0. Legal situations are constrained in several ways. For instance, the precondition axioms for actions enforce the constraint that an action whose preconditions are not satisfied does not lead to a legal situation. Also, Reiter introduced an approach to handling natural actions [31] that provides further constraints for legal situations. In this paper, we propose to view occurrence statements as constraints on the legal situations in the tree. These constraints can be seen as "pruning" statements that tell us what situations in the tree are suitable hypothetical futures. For instance, a narrative will become a set of sentences specifying which paths along the tree of situations can be considered valid developments.

Our approach to handle occurrences in the Situation Calculus has several advantages. For instance, it allows for the representation of simple non-preventable occurrences and narratives. Also, it allows for the representation of triggers [42] and preventable occurrences. Furthermore, the advantages of the original Situation Calculus are not lost. In particular, we are still able to do deductive planning in the presence of occurrences. Moreover, this approach does not suffer from what Reiter calls *premature minimization problem* [36]:

> "[This problem] amounts to the assumption that all action occurrences are either specified as part of the axiomatization, or are inferable from it. Closure, in the form of minimization of action occurrences, is enforced by suitably circumscribing these axioms."

In fact, we show that our axiomatization does not preclude one from considering action occurrences that are not entailed by the theory. However, we can do even better. Indeed, we show that it is possible, within the logical language, to characterize sequences of actions that are preferred using some minimality assumptions. Interestingly, we are able to define this notion of *preferred development* within the language of the Situation Calculus without having to appeal to circumscription axioms.

The reader might wonder why we chose the Situation Calculus as a formal framework to build theories for the representation of knowledge about action and change. To date, there is no broadly accepted formalism for the representation of knowledge of this nature. However, the Situation Calculus has emerged as one candidate approach. There is a concerted effort to enrich the language with the ability to represent a variety of types of knowledge relevant for dynamic domains. For instance, there are proposals to deal with concurrency [17, 26, 36], proposals to extend the Situation Calculus to deal with continuous change [26, 27, 23, 36], proposals to

add epistemic features [37, 13], etc. Furthermore, the Situation Calculus has been proposed as the basis for the development of the agent programming language *GOLOG* [12, 14]. The work reported in this article forms part of this research tradition, started by John McCarthy in [19].

An essential stepping stone for our research is Raymond Reiter's solution to the frame problem [33]. This solution provides a framework in which the dynamics of the world is specified utilizing first order axioms. These axioms are called *successor state axioms* which specify necessary and sufficient conditions for change. The successor state axioms are derived from a set of effect axioms (or causal rules) along with the *explanation closure assumption* [38]. Furthermore, it is assumed that all the conditions for the execution of actions are known. These two assumptions justify the use of *if and only if* specifications for action effects and their preconditions. These *if and only if* specifications are similar to sets of implications completed with circumscription axioms. The circumscriptive axiomatizations have the advantage that updates are easier to incorporate. In [16], Lin and Reiter show the relationship between Reiter's solution to the frame problem and an approach based on circumscription.

The structure of this paper is as follows: In section 2, we present the theoretical framework on which the rest of the paper is based. In section 3, we present our approach to modeling occurrences and triggers. In section 4, we present applications of this approach to knowledge representation problems. In section 5, we compare our approach to approaches based on a time line. Finally, in section 6, we present our concluding remarks and discuss future developments of these ideas.

2 Theoretical Framework

In this section we present the formal framework that will be used in the rest of the paper. The basic language and axiomatization are extensions to the Situation Calculus formalism proposed in [26, 31, 36]. The main novelty is that we introduce new predicates in the language that allow us to refer to finite paths or histories in the Situation Calculus tree structure. In the following section, these new predicates will be used to define different notions of legality of situations in the presence of occurrence sentences.

2.1 Language

The Situation Calculus we use is a sorted first order language extended with a second order induction axiom to characterize the space of situations. We distinguish the sorts $\mathcal{A}$, $\mathcal{C}$, $\mathcal{S}$ and $\mathcal{T}$ for primitive actions, concurrent actions, situations and time. We also use a sort $\mathcal{D}$ for other objects. The sort $\mathcal{T}$ ranges over the non-negative real numbers. We use the convention that all variable symbols start with lower case letters, whereas constant symbols start with upper case letters. Also, we use the Greek letter φ as a second order predicate variable, which is only used in the specification of the induction axiom for situations given in section 2.3. The sort of the variables and constants should be inferable from the context. In formulas, we assume that all free variables are universally quantified from the outside. The sort $\mathcal{C}$ corresponds to sets of primitive actions (following [36]). We take the standard interpretation for sets and don't axiomatize them.

Function and Predicate Symbols

$start : \mathcal{S} \rightarrow \mathcal{T}$**.** If s is a situation term, then $start(s)$ denotes the time at which s starts.

$do : \mathcal{C} \times \mathcal{S} \rightarrow \mathcal{S}$. If c denotes a concurrent action and s denotes a situation, then $do(c,s)$ denotes the situation that results from performing c in s. For notational convenience, the term $do(c_n, do(\ldots, do(c_1, S_0)\ldots))$ will be written as $do([c_1,\ldots,c_n], S_0)$.

$\in \subseteq \mathcal{A} \times \mathcal{C}$. This is the standard membership operator for sets.

hasOccurred $\subseteq (\mathcal{A} \cup \mathcal{C}) \times \mathcal{T} \times \mathcal{S}$. The first argument of *hasOccurred* can be either a primitive action or a concurrent action. *hasOccurred*(x,t,s) would be true if action x (primitive or concurrent) occurred at time t on the path that leads from S_0 to s.

$\prec \subseteq \mathcal{S} \times \mathcal{S}$. If s_1 and s_2 denote situations, then $s_1 \prec s_2$ is true if some sequence of concurrent actions leads from s_1 to s_2. We use the abbreviation:

$$s_1 \preceq s_2 \equiv s_1 \prec s_2 \vee s_1 = s_2 .$$

Poss $\subseteq (\mathcal{A} \cup \mathcal{C}) \times \mathcal{S}$. The first argument of *Poss* can be either a primitive action or a concurrent action. If x is an action (primitive or concurrent) and s a situation, then *Poss*(x,s) is true if the action x is possible in s (i.e., if the action preconditions are satisfied in the situation denoted by s).

$< \subseteq \mathcal{S} \times \mathcal{S}$. If s_1 and s_2 denote situations, then $s_1 < s_2$ is true if some sequence of *possible* concurrent actions leads from s_1 to s_2. We use the abbreviation:

$$s_1 \leq s_2 = s_1 < s_2 \vee s_1 = s_2 .$$

$< \subseteq \mathcal{T} \times \mathcal{T}$. We overload this symbol and use it as the standard comparison operator for reals (the abbreviation $\leq$ is also used).

$legal_{poss}, legal \subseteq \mathcal{S}$. These predicate are used to characterize those situations that are valid; a situation is considered valid if and only if the sequence of actions that lead from S_0 to s obeys all the preconditions and occurrence constraints.

proper $\subseteq \mathcal{S}$. Is an auxiliary predicate used to define *legal* situations.

Sit $\subseteq \mathcal{T} \times \mathcal{S} \times \mathcal{S}$. The literal *Sit*$(t,s,s_h)$ is intended to mean that t falls in the situation s in the path that leads from S_0 to s_h. The predicate *Sit* used here is not related to the the mapping `Sit` used in [25].

occurrence predicates. In section 3, we introduce several notions of occurrence. In each case, a different *occurrence* predicate is introduced.

2.2 Representational Restrictions

In this paper, we consider a Situation Calculus in which actions are considered to be instantaneous. As discussed elsewhere (e.g., [26]), this restriction does not limit the representational power of the language, since actions with durations can be modeled as a pair of instantaneous actions, one which starts the original action, and one that finishes it. Furthermore, one can easily introduce a new sort of actions with durations with mappings to instantaneous actions that start and finish them.

Also, we ignore precondition interactions and concurrent effects or cancellations. Precondition interactions arise when the concurrent execution of two actions is precluded in spite of

the fact that each action can be executed independently. The concurrent effects arise when the concurrent execution of two actions brings about effects that neither action would have had when executed individually. Finally, concurrent cancellation arises when the effect of one action is canceled when the action is concurrently executed with another. These problems are discussed in [26].

The issues discussed in this paper are orthogonal to those mentioned above. Therefore, any approach to deal with precondition interactions and concurrent effects/cancellation should integrate well with the theory presented in this paper.

2.3 Basic Situation Calculus Axiomatization

In the Situation Calculus there is an initial situation, denoted by the distinguished constant symbol S_0. Given an arbitrary situation s, a new situation is obtained by performing some concurrent action c in s. Following Reiter [33], we use an induction axiom in order to force the set of situations to form a tree rooted at S_0. Thus, an arbitrary situation s identifies a unique sequence of concurrent actions that leads from S_0 to s. Each situation s in the tree has a unique starting time [30]; this time corresponds to the time of the last action in the sequence leading to s. On the other hand, there is no unique ending time for a situation. In fact, in [30], Pinto and Reiter define an *end* predicate that takes an action and a situation. Roughly, the end of a situation s with respect to an action c corresponds to the start time of the situation $do(c,s)$. These multiple ending times arise from the fact that, in the Situation Calculus, the future is branching.

The basic axioms for situations, explained below, are:

$$(\forall \varphi).[\varphi(S_0) \wedge (\forall s,c)\ (\varphi(s) \supset \varphi(do(c,s)))] \supset (\forall s)\ \varphi(s), \tag{1}$$

$$do(c_1,s_1) = do(c_2,s_2) \supset c_1 = c_2, \tag{2}$$

$$s_1 \prec do(c,s_2) \equiv s_1 \preceq s_2, \tag{3}$$

$$s_1 \prec s_2 \supset \neg s_2 \prec s_1, \tag{4}$$

$$Poss(c,s) \equiv (\forall a)[a \in c \supset Poss(a,s)], \tag{5}$$

$$Poss(c,s) \supset start(s) < start(do(c,s)). \tag{6}$$

The first axiom is a second order induction axiom, similar to Peano's induction axiom used to characterize the natural numbers. This axiom states that the only situations that exist are S_0, the initial situation, and all the situations that are reachable from S_0 by performing sequences of actions starting in S_0. It is important to point out that this is the only use we make of second order logic in this article. Furthermore, in the same manner as is done with the natural numbers, we could replace this axiom with an axiom schema and work with a *standard interpretation*. In theorem proving, the axiom can be left implicit in the proof strategy, as done with *RRL* (Rewrite Rule Laboratory [9]) in [3], where induction is used in order to mechanically prove integrity constraints from a Situation Calculus specification of database transactions.

Axiom (2) ensures that the structure of situations forms a tree, and roughly corresponds to Shanahan's arboreality axiom [40]. Axioms (3) and (4) define the relation $\preceq$. The literal $S_1 \prec S_2$ is true if there is a path from S_1 to S_2 in the Situation Calculus tree; i.e., S_2 is reached by performing a sequence of actions starting from S_1.

Axiom (5) states that a concurrent action is possible iff all its constituent actions are individually possible. We assume that the predicate *Poss* for simple actions is completely specified by domain axioms.

As mentioned before, the function *start* denotes the starting time of a situation. Axiom (6) ensures that the situations along a path are chronologically ordered. This ordering is only enforced for situations that are obtained when performing actions that are possible.

2.4 Situation Legality

In the previous sub-section, we presented the basic axiomatization that defines the universe of situations $\mathcal{S}$ in terms of the initial situation S_0 and the set of concurrent actions $\mathcal{C}$. Notice that this universe of situations is defined without taking into consideration whether or not the situations are actually possible. That is, there are situations that are reached by performing sequences of actions that include actions that are not possible. Our objective is to specify which situations in the Situation Calculus tree can be regarded as viable or *legal* given knowledge regarding action preconditions, laws of physics, occurrence statements, etc. In [35], Reiter defines a first notion of legality which basically says that a situation s is legal if it can be reached by performing actions in S_0 and all the actions leading to s are performed in situations in which their preconditions are met. This motivates the following:

Definition 2.1 *A situation s is considered legal with respect to the action preconditions, denoted* $legal_{poss}$*, if all the actions that lead from* S_0 *to s have been* done *in situations in which they are possible. Thus,* $legal_{poss}$ *is defined as:*

$$legal_{poss}(s) \equiv s = S_0 \vee (\forall c, s')\ do(c, s') \preceq s \supset Poss(c, s') \tag{7}$$

Reiter defines legality with the predicate $<$, which he axiomatizes as follows:

$$\neg s < S_0, \tag{8}$$

$$s < do(c, s') \equiv Poss(c, s') \wedge s \leq s'. \tag{9}$$

The relationship between $legal_{poss}$ and $<$ is straightforward. In fact, we have the following simple observation:

Observation 2.1 *From axioms (1)–(9) it follows that:*

$$legal_{poss}(s) \equiv S_0 \leq s.$$

PROOF: The proof is a direct application of the induction axiom (1) with

$$\varphi(s) = (legal_{poss}(s) \equiv S_0 \leq s).$$

□

The legality predicates – $legal_{poss}$, along with other *legal* predicates – are introduced as a tree pruning mechanism. Thus, $legal_{poss}$ is a first step towards specifying which part of the Situation Calculus tree corresponds to valid developments. As a next step in this pruning process, we can incorporate the notion of natural actions [26, 31, 36]. Natural actions are the result of the laws of physics. For instance, the action $hitFloor(Ball)$ might be regarded as natural if *Ball* is falling in such a way that it will hit the floor at a certain time. See [36] for an approach to introducing natural actions as constraints on the legal situations. In this article we ignore natural actions. However, it is not difficult to incorporate them in our framework as further constraints on the legality of situations. We have explored this venue utilizing an approach similar to the one presented in [27], and the results are still preliminary.

2.5 Situations as Histories or Paths

The notion of a history, within the Situation Calculus, corresponds to the sequence of actions that leads to a situation starting from S_0. Also, we refer to this as a *partial path* within the Situation Calculus tree. A *path* in the Situation Calculus tree corresponds to an infinite sequence of situations that start in S_0, a partial path is a prefix of infinitely many paths. Thus, the notion of *path* corresponds to McDermott's notion of *chronicle* [22]. Any situation s identifies a unique partial path that starts at S_0 and ends with s. In fact, any situation in the domain $\mathcal{S}$ is fully identified with a sequence of actions. For instance, the sequence $c_1, c_2, \ldots, c_n$ uniquely identifies the situation S_n, where:

$$S_n = do([c_1, \ldots, c_n], S_0)$$

Thus, all the actions $c_1, \ldots, c_n$ can be said to belong to or have occurred before S_n, and they correspond to the history of situation $do([c_1, \ldots, c_n], S_0)$. It turns out that one essential feature of the Situation Calculus is that there are terms denoting situations, which also denote histories. Thus, a situation s corresponds to a single history or linear time description of one possible evolution of the world starting from S_0. A situation s' is said to appear or belong to a history s iff $s' \prec s$. Thus, S_0 belongs to all histories (except to the empty history, which corresponds to S_0). Notice that the notion of history comes for free in the Situation Calculus and no added machinery needs to be introduced in the language.

The reader might be wondering why we introduce new distinctions (history, partial path) in order to refer to the concept of situation. The reason is that we want to emphasize the fact that situations are looked at in a non-traditional manner. It is very common to consider situations as *snapshots* that don't have an inherent history. We use the term history in order to emphasize that we are looking at an aspect of a situation which is normally not taken into account.

The branching nature of time in the Situation Calculus makes it difficult to impose a unique ending time on situations (they do have a unique start time, since time in the Situation Calculus is left linear). On the other hand, in a given history, any situation has a unique starting and a unique ending time. If s and $do(c,s)$ belong to some history s', then $start(s)$ and $start(do(c,s))$ correspond to the starting and ending times of s along the history s'.

In this paper, we exploit the notion of histories to define occurrences. We do so by defining the notion of a concurrent action c occurring within a history, partial path or situation:

Definition 2.2 *We say that a concurrent action c has occurred at time t along the history of s with the predicate hasOccurred as follows*[3]*:*

$$hasOccurred(c,t,s) \equiv (\exists s')\ do(c,s') \preceq s \wedge start(do(c,s')) = t. \tag{10}$$

This is extended for primitive actions as:

$$hasOccurred(a,t,s) \equiv (\exists c)\ a \in c \wedge hasOccurred(c,t,s). \tag{11}$$

An interesting property of these definitions is the following:

Observation 2.2 *From axioms (1)–(11) it follows that some action c has occurred in a history s if it is equal to the last action in the history, or if it has also occurred in the situation immediately*

[3] Therefore, the past tense in *hasOccurred* refers to the past of a particular situation or history, rather than to the domain as a whole.

preceding s:

$$hasOccurred(c,t,do(c',s)) \equiv c = c' \wedge start(do(c',s)) = t \vee hasOccurred(c,t,s),$$

$$hasOccurred(a,t,do(c',s)) \equiv a \in c' \wedge start(do(c',s)) = t \vee hasOccurred(a,t,s).$$

PROOF: Apply definition (10) and axiom (3).

□

Also, we have:

Observation 2.3 *From axioms (1)–(11) it follows that if two concurrent actions c and c' have occurred at the same time in a given history, then both actions are the same:*

$$hasOccurred(c,t,s) \wedge hasOcurred(c',t,s) \supset c = c'. \tag{12}$$

PROOF: From the basic structural axioms (1)–(4) it is possible to show that the structure of situations is left linear, that is:

$$[s_1 \preceq s \wedge s_2 \preceq s] \supset [s_1 \preceq s_2 \vee s_2 \preceq s_1]$$

This property, along with (6) and definition (10) lead to the proof.

□

Finally, we introduce the following:

Definition 2.3 *We say that $Sit(t,s,s_h)$ is true whenever s corresponds to the situation in the history s_h which includes time t (thus t is between the start of s and the end of s along the history s_h). Formally,*

$$Sit(t,s,s_h) \equiv (\exists c)\ do(c,s) \preceq s_h \wedge start(s) < t \wedge t \leq start(do(c,s)). \tag{13}$$

Using these definitions we have:

Observation 2.4 *From axioms (1) to (13) it follows that the situation that includes time t along the history $do(c,s)$, for some situation s and action c, is s. I.e.:*

$$Sit(start(do(c,s)),s,do(c,s)).$$

Furthermore, if two situations include a single time point along some history, then both situations must be the same, i.e.:

$$Sit(t,s_1,s_h) \wedge Sit(t,s_2,s_h) \supset s_1 = s_2.$$

PROOF: The first part is straightforward. The second part involves the use of the left linear property of the structure of situations, mentioned in the proof of observation 2.3, and the use of induction along with axiom (6).

□

3 Formalizing Action Occurrences

3.1 Further Constraints on Legal Situations

The main purpose of this paper is to deal with the problem of representing the notion of *action occurrence* in the framework of the Situation Calculus. A solution to the problem of representing action occurrences leads to a solution to the problem of representing narratives and triggers.

In this section, we introduce two new predicates $proper \subseteq \mathcal{S}$ and $legal \subseteq \mathcal{S}$. A situation is said to be a proper history, iff it obeys all the constraints up to the start of the situation (i.e., without looking towards the future of the situation). This predicate is introduced in order to characterize those histories that are legal with respect to the preconditions and that conform to the occurrence specifications (introduced later). *proper* is a predicate defined in terms of occurrences and the $legal_{poss}$ legality predicate (perhaps enhanced with restrictions to account for natural actions). However, for presentation purposes, the specification of *proper* will be given as a set of implications or necessary conditions for *proper*. In the end, the set of necessary conditions for *proper* will be regarded as sufficient conditions as well. The situations characterized by the *proper* predicate form a subtree of the original situation tree structure, rooted in S_0,

The first necessary condition for a situation to represent a proper partial path is:

$$proper(s) \supset legal_{poss}(s). \tag{14}$$

Thus, actions that are not possible do not lead to proper histories.

All other constraints on *proper* will be introduced in the context of the occurrence predicates that are introduced in later sections.

On the other hand, the predicate *legal* should be taken as a refinement of the legality notion introduced with the $legal_{poss}$ predicate. In fact, if a situation is *legal*, then it is also $legal_{poss}$. A situation is *legal* if it obeys all constraints, related to action preconditions and *occurrence* sentences. The definition of *legal* is done in terms of the *proper* predicate:

Definition 3.1 *A situation or history s is considered to be legal with respect to the action preconditions and occurrence statements if every time after its start there is a proper future history that contains s. This is formalized as:*

$$legal(s) \equiv [(\forall t)\ start(s) < t \supset (\exists s')\ s \prec s' \wedge t < start(s') \wedge proper(s')]. \tag{15}$$

The definition of legal is done in terms of *proper*, which, in turn, is defined in terms of $legal_{poss}$. This means that we concentrate our interest on the subtree in which all the situations are legal with respect to the action preconditions. Notice that according to (15) for a situation to be legal, at any time t after its start, some action must always be possible some time t' after t.

In what follows we discuss several forms in which the notion of occurrence might arise in knowledge representation. We divide occurrence statements into several classes. We discuss occurrences that cannot be prevented. We also discuss occurrences that arise unless certain preventing conditions are satisfied. In each of the cases discussed, a simple example is presented. To model each type of occurrence we introduce new occurrence predicates and establish constraints between *proper* and these predicates.

It should be noted here that the notion of a history or partial path, which is formally equivalent to a situation, is essential in the definition of occurrences. In fact, the basic language introduced in the previous section might be considered as the foundational layer that allows one to represent occurrences. Different notions of occurrences correspond to different uses of the path predicates.

The reader might regard the *family* of occurrence predicate as too diverse, and might feel that a single notion of occurrence should be defined. Indeed, the definition of *hasOccurred* for actions, times, and histories (or situations) corresponds to this single occurrence predicate.

3.2 Non-Preventable Occurrences

3.2.1 Occurrences in Time

Consider the sentence:

The sun will rise tomorrow at 6:03 a.m.

Assuming that this sentence is true, we are faced with the problem of representing the occurrence in the Situation Calculus. In [30], Pinto and Reiter addressed this problem by stating that there is an actual line that describes the *real* evolution of the world and that in this line the sun rises at 6:03 a.m. Unfortunately, this way of modeling this type of occurrence allows for hypothetical developments in which the sunrise does not happen at all. The problem with Pinto and Reiter's framework is that they consider occurrences as being assertions regarding the actual line rather than assertions about the nature of the world. Instead, we consider occurrences to be assertions that constrain the paths in the tree that are proper. We introduce simple non-preventable occurrences with the predicate $occurs_{np} \subseteq \mathcal{A} \times \mathcal{T}$ such that $occurs_{np}(a,t)$ will mean that the action must be part of any legal history.

Thus, we have:

$$proper(s) \supset [occurs_{np}(a,t) \wedge t < start(s) \supset hasOccurred(a,t,s)]. \tag{16}$$

Thus, the sunrise statement above would be modeled as:

$$occurs_{np}(Sunrise, 6{:}03am).$$

Now we have:

Theorem 3.1 *From axioms (1) to (16) it follows that any legal situation that starts after a time t must include all actions that non-preventably occur before time t:*

$$occurs_{np}(a,t) \wedge legal(s) \wedge t < start(s) \supset hasOccurred(a,t,s).$$

PROOF: It follows directly from (15) and (16).

□

As illustrated by the example of section 4.1, theorem 3.1 establishes essential conditions to deal with narratives.

3.2.2 Triggered Occurrences

A triggered occurrence is related to the notion of triggers in active database systems [42]. A trigger is a relationship between two actions. If a_1 and a_2 are actions, then a_1 triggers a_2 if the first action *causes* the second one to occur at a later time. It is not our intention to provide an account for *causality*. However, the formal language that we have introduced provides the

necessary support for modeling crucial features of causation[4]. The notion of trigger used here is different but related to the notion of trigger used in related research. For instance, in [23], Rob Miller specifies a formalism in which actions can be *triggered* at an instant of time depending on a particular set of circumstances.

As a first example consider:

If you eat the forbidden fruit you will be expelled.

In this example, we are given information that is counterfactual in nature. We are told that if we were to eat the forbidden fruit, then we would be punished. One attempt at modeling this piece of information might be[5]:

$$\texttt{occurs}_{\text{np}}(\texttt{EatFruit},\texttt{t}) \supset \texttt{occurs}_{\text{np}}(\texttt{Expel},\texttt{t}+\Delta). \qquad \text{(ex*)}$$

Unfortunately, this is inadequate. The reason is that if $\texttt{occurs}_{\text{np}}(\texttt{EatFruit},\texttt{t})$ is not provable, we would not be able to correctly hypothesize about the consequences that eating forbidden fruits might have. This is because it is not possible to treat material implication as counterfactual implication. In order to model this type of occurrence statements, we introduce the predicate $occurs_{to} \subseteq \mathcal{A} \times \mathcal{T} \times \mathcal{S}$. The literal $occurs_{to}(a,t,s)$ is intended to mean that action a will definitely occur at time t after s has started. Therefore, to model the previous sentence, we write:

$$EatFruit \in c \supset occurs_{to}(Expel, start(do(c,s)) + \Delta, do(c,s)) \qquad \text{(ex)}$$

Thus, if c is some concurrent action that includes an $EatFruit$ primitive action, then the $Expel$ action will necessarily occur Δ time units later. For a more earthly example, let A_1 denote the event "the high temperature alarm goes off" and A_2 denote the action "initiate emergency cool-down procedure". Then, to specify that A_1 triggers A_2 after Δ time units, we should write:

$$A_1 \in c \supset occurs_{to}(A_2, start(do(c,s)) + \Delta, do(c,s))$$

Since trigger statements have a common syntactic form, we can introduce the predicate $trigger \subseteq \mathcal{A} \times \mathcal{A} \times \mathcal{T}$ as an abbreviation:

$$trigger(a_1,a_2,\delta) \equiv [(\forall c,s)\ a_1 \in c \supset occurs_{to}(a_2, start(do(c,s)) + \delta, do(c,s))] \qquad (17)$$

This notion of trigger can be used to model some physical phenomena at an abstract level. For instance, we can write $trigger(Drop, HitGround, \Delta)$ to mean that Δ time units after a $Drop$ is performed a $HitGround$ event occurs. Thus, triggers allow us to model natural events at a coarse level of granularity, abstracting away from details of the physical processes involved. There are circumstances in which we might prefer to reason about *natural actions* [26, 27, 36] in this manner, as opposed to treating them as a result of some intricate physical process.

The set of situations in *proper* are further constrained by $occurs_{to}$. In fact, we need to introduce a new necessary condition for this predicate, namely:

$$proper(s) \supset [occurs_{to}(a,t,s') \wedge t < start(s) \wedge s' \prec s \supset hasOccurred(a,t,s)]. \qquad (18)$$

[4] In particular, causal relations between occurring events, as opposed to causal effects of actions on fluents.

[5] We use teletype font to indicate that we consider the sentence to incorrectly model the knowledge we want to express.

Theorem 3.2 *From axioms (1) to (18) it follows that if a_1 is an action that triggers a_2, then a_2 will be triggered in all legal developments:*

$$occurs_{np}(a_1,t) \wedge trigger(a_1,a_2,\delta) \wedge legal(s) \wedge t+\delta < start(s) \supset \\ hasOccurred(a_2,t+\delta,s).$$

PROOF: It follows from (10), (11), (16), (17) and (18).

□

It can be argued that triggered occurrences do not arise in the real world. Indeed, events or actions are normally triggered by conditions that are true in the world. For instance, in the fruit eating example, one might argue that the *Expel* action is triggered by the condition resulting from *Eat* (i.e., $Fruit - eaten$) and not from the *Eat* action itself. While this seems a reasonable view for modeling the world as it *really* is, the triggering events concept is a useful abstraction. In particular, when modeling or specifying artificial devices, one would like to express relations between actions in this manner. For instance, this is true of control systems, or active databases, where one wants systems to react to events happening in the world (alarms going off, transactions being attempted, etc.). Therefore, triggers need to be supported in order to allow for a higher level of abstraction.

3.3 Conditional Occurrences

3.3.1 Preventable Occurrences.

Consider the following assertion "the train to Ottawa leaves every day at 7 p.m." To simplify things, we will assume that this information is valid *forever* (i.e., the train schedule does not change). However, this piece of knowledge must not be taken literally, even if we consider that it is valid forever. In fact, this statement is *defeasible* in the sense that *the train will leave as long as nothing wrong happens*. Knowledge of this kind is very common and we will refer to this type of occurrence as a *preventable occurrence*. An occurrence is said to be preventable if it depends on the state of the world at the time the occurrence has been predicted.

We introduce a new special predicate $occurs_{po} \subseteq \mathcal{A} \times \mathcal{T}$ to state that a given action occurs *if it is possible for it to occur*. For instance:

$$\psi(t) \supset occurs_{po}(TrainLeaves,t)$$

states that every time that ψ is true of a time point t[6], the train leaves if it is possible for it to leave.

The relationship between $occurs_{po}$ and the predicate *proper* is established with a new necessary condition for *proper* situations:

$$proper(s) \supset [occurs_{po}(a,t) \wedge t < start(s) \wedge \\ Sit(t,s',s) \wedge Poss(a,s') \supset hasOccurred(a,t,s)]. \quad (19)$$

Assume that an action a is said to preventably occur at time t. Then, given this new constraint on proper histories, any situation s that starts after t must include a in its history, unless a was not possible at time t along s's history.

[6] ψ might be a formula like $(\exists i)\ t = T_0 + K \times i$, where K and T_0 are some positive constants and i denotes a positive integer number.

3.3.2 Conditional Triggered Occurrences

Consider the assertion "If my neighbor's burglar alarm goes off while I am at home, I will call the police." The difference between a conditional trigger and a non-preventable one is that the *caused* action can be averted. Here, we introduce the predicate $occurs_{ct} \subseteq \mathcal{A} \times \mathcal{T} \times \mathcal{S}$ in order to model this type of trigger. The literal $occurs_{ct}(a,t,s)$ is interpreted to mean that action a will occur at time t after the time $start(s)$, unless it is not possible for it to occur.

The example of the alarm is handled as:

$$AlarmOff \in c \supset occurs_{ct}(CallPolice, \Delta, do(c,s))$$

Thus, I will call the police Δ time units after the alarm goes off, unless I am prevented from doing so. The new necessary condition for *proper* becomes:

$$\begin{aligned} proper(s_h) \supset occurs_{ct}(a,t,s_1) \wedge t' = start(s_1) + t \wedge t' < start(s_h) \wedge \\ s_1 \prec s_h \wedge Sit(t', s_t, s_h) \wedge Poss(a, s_t) \supset hasOccurred(a, t', s_h). \end{aligned} \tag{20}$$

Formula (20) is analogous to (18) with the added condition that the occurring action be possible when it is meant to occur.

3.4 Simple Occurrences

In some cases we might want to express knowledge regarding some action occurrence without specifying any details. For instance, we might want to state that action *Finish* will eventually occur; i.e., that the action will appear somewhere in the time line. If we were using a linear view of time we would express this with a sentence like:

$$(\exists \mathtt{t})\mathtt{occurs}(\mathtt{Finish}, \mathtt{t}) \tag{so*}$$

which might be taken to mean that eventually the action *Finish* will occur. In a branching time logic, we need to specify that *Finish* must arise in any possible development. In our language, we could choose to model the eventual occurrence of *Finish* with the statement:

$$(\exists \mathtt{t})\ \mathtt{occurs}_{\mathtt{np}}(\mathtt{Finish}, \mathtt{t}). \tag{so**}$$

Indeed, this statement ensures that there is a time t such that any legal situation that starts after t has a history that includes *Finish*. However, (so**) is too strong. The problem that arises is that this statement forces *Finish* to occur at the same time in all the histories. What we need is a statement that leaves the time of the occurrence of *Finish* unconstrained. To achieve this, we introduce the predicate $occurs_{by} \subseteq \mathcal{A} \times \mathcal{T}$. The intended meaning of $occurs_{by}(a,t)$ is that action a has occurred before time t. Thus, instead of (so*) we write[7]:

$$(\exists t) occurs_{by}(a,t), \tag{so}$$

which constrains *proper* as follows:

$$proper(s) \supset [occurs_{by}(a,t) \wedge t < start(s) \supset (\exists t')\ t' < t \wedge hasOccurred(a,t',s)]. \tag{21}$$

[7] The reader should keep in mind that the sentences (so*) and (so), which appear almost identical, belong to theories with a totally different background axiomatization.

Therefore, the statement $occurs_{by}(Finish, t)$ states that all the legal situations that start at t or later have histories in which *Finish* has already taken place.

The simple occurrences discussed here are essential to express statements like "Eventually a occurs". For instance, "after getting into Dr. X's office, I will eventually come out." Statements of the form (so) allow us to express the future occurrence of actions without committing to a time at which they ought to occur. This notion of *eventuality* is related to the similar notion in temporal logics [18]. The main difference is that, in temporal logics, the operator $\Box$, used to express the notion of *eventuality*, is applicable to truth valued sentences as opposed to actions or events.

3.5 Final Remarks on Proper Histories

As mentioned before, we consider *proper* to be a defined predicate whose definition is obtained from the closure of its necessary conditions. Therefore, by closing the axioms for predicate *proper* with (14), (16), (18), (19), (20) and (21), we obtain:

$$\begin{aligned} proper(s) \equiv\ & legal_{poss}(s) \wedge \\ & [occurs_{np}(a,t) \wedge t < start(s) \supset hasOccurred(a,t,s)] \wedge \\ & [occurs_{to}(a,t,s') \wedge t < start(s) \wedge s' \prec s \supset hasOccurred(a,t,s)] \wedge \\ & [occurs_{po}(a,t) \wedge t < start(s) \wedge \\ & \quad Sit(t,s',s) \wedge Poss(a,s') \supset hasOccurred(a,t,s)] \wedge \\ & [occurs_{ct}(a,t,s_1) \wedge t' = start(s_1) + t \wedge t' < start(s) \wedge \\ & \quad s_1 \prec s \wedge Sit(t',s_t,s) \wedge Poss(a,s_t) \supset hasOccurred(a,t',s)] \wedge \\ & [occurs_{by}(a,t) \wedge t < start(s) \supset (\exists t')\ t' < t \wedge hasOccurred(a,t',s)]. \end{aligned} \tag{22}$$

Thus, the definition of *proper* is done in terms of the occurrence predicates, which are primitive, and should be specified by the axiom writer. Each occurrence statement introduces *dynamic constraints* in terms of the *hasOccurred* predicate. Therefore, in principle, one can simply define, for a particular application, the *proper* predicate directly in terms of *hasOccurred*. However, when one represents temporal knowledge, there are certain *dynamic constraints* that arise naturally. The occurrence predicates presented here characterize some of these constraints. The variety of occurrence predicates introduced, which might seem confusing, is due to the variety of dynamic constraints that one can represent. We do not claim that the set of occurrence predicates that we have presented is exhaustive. In fact, there are other types of occurrence that can also be specified in a similar form. For instance, we could add simple occurrences that are preventable and/or triggered. The main contribution of this work is to set the stage for the definition of these notions of occurrences. The closure of this predicate should not be taken as a completeness assumption. Rather, it corresponds to a complete specification of the set of *proper* situations. In the next section, we illustrate how one can use occurrence statements in different applications.

Our work relies on a fairly elaborate notion of situation legality. As already stated, legal situations form a subtree of situations rooted in the initial situation. A natural question that arises is regarding non-legal situations. What role do they play? Can they give rise to contradictions? From our point of view, non-legal situations are *ghost* situations or *dead-wood* (term suggested by one of the referees of this article). As discussed by Reiter [33], in order to avoid contradictions, one should ensure certain consistency conditions in the formalization of the domain. These

consistency conditions can be summarized as requiring that no action make a fluent false and true in any situation. However, we would like to establish conditions under which we can guarantee that a given domain axiomatization is consistent. This is matter for further investigation.

4 Applications

In this section, we take Σ to denote the background axioms, i.e.:

- The basic axioms of the Situation Calculus, (1)–(9).
- Definition of *hasOccurred* plus the *path* predicates, (10)–(13).
- Definition of the *legal* predicate (15), the definition of the notion of trigger (17), and the definition of proper history (22).

The occurrence predicates have to be specified in the formalization of any domain, and no background axioms are provided for them.

4.1 Narratives and Preferred Developments

4.1.1 Narratives

Consider a slightly modified version of Miller and Shanahan's briefcase example[25]:

> Before breakfast one morning, a lecturer, Mary, puts her lecture notes inside the briefcase. She eats breakfast, and shortly thereafter carries her briefcase to college. Mary concludes that the lecture notes are in her briefcase at college. However, shortly after she sits down at her desk, her husband telephones to apologize for accidentally removing her notes from the briefcase before she left for work. With her new knowledge of her husband's actions, Mary concludes that her notes are not at college.

We will consider the actions $PutIn$, Eat, $TakeNotes$, $Carry$ which denote the *put in briefcase, eat breakfast, take notes from briefcase* and *carry briefcase to college* actions.

Let Ω denote the following sentences:

$$occurs_{np}(PutIn, 6{:}30am), \tag{23}$$
$$occurs_{np}(Eat, 6{:}35am), \tag{24}$$
$$occurs_{np}(Carry, 7{:}00am). \tag{25}$$

Then, using theorem 3.1 we obtain:

Observation 4.1 *The background axioms Σ, along with the occurrence statements Ω, entail that the history of any legal situation, that starts after the time of the last action in the narrative, will contain all the actions in the narrative.*

$$\begin{aligned}\Sigma \cup \Omega \models\ (\forall s)\ .legal(s) \wedge 7{:}00am < start(s) \supset \\ hasOccurred(PutIn, 6{:}30am, s) \wedge \\ hasOccurred(Eat, 6{:}35am, s) \wedge \\ hasOccurred(Carry, 7{:}00am, s).\end{aligned} \tag{26}$$

PROOF: *It follows directly from the definition of legal (15), and constraint (16).*

□

Thus, the question is whether we can prove that Mary's notes are at College with her, given the set Ω of occurrences. According to the *actual line* of events[8], this should be the case. However, what should a Situation Calculus based theory entail? If we want to keep the hypothetical view of the world we must accept the possibility that other things might happen aside from the occurrences that are explicitly stated. For instance, it might be the case that the action $TakeNotes$, that refers to the husband's action, happens or that it does not. Since our theory does not say that it occurs, then it might or it might not occur.

In fact, we have:

Observation 4.2 *Given axioms $\Sigma \cup \Omega$, there are legal histories that contain the action $TakeNotes$ by which Mary's husband removes the notes from her briefcase. This can be formalized:*

$$\Sigma \cup \Omega \models (\exists s,t).legal(s) \wedge 7{:}00am < start(s) \wedge hasOccurred(TakeNotes,t,s).$$

Notice that there models of $\Sigma \cup \Omega$ in which the $TakeNotes$ action is concurrent with Eat.

Each path in the tree of situations can be seen as a *possible* way in which the world might evolve. This is akin to Kripke's possible worlds semantics. That is, each one of the paths represents a possible world or a possible development. If we state that some action A occurs, then we mean that any path that does not contain A is deemed ill-formed. Also, in a similar spirit to the possible worlds semantics, the negation of an occurs literal will not mean that its action argument does not appear in any path, it simply means that there is a hypothetical course of events in which this action is not present.

4.1.2 Preferred Developments

A great advantage of the Situation Calculus over modal logics is that situations, which could be viewed as worlds, are objects in the language. Therefore, we can predicate over situations within the language. Here, we will use this feature of the Situation Calculus to define a notion of preference between situations. The idea is to be able to compare two legal situations and decide which one might better explain the information that we have at hand. The preference relation between two situations will be based on a comparison between their common histories. Thus, if s starts at time 12 and s' starts at time 15, then we compare the actions that appear in their histories between times 0 (the start of S_0) and time 12 (the situation that starts earlier). The preference that we define below is such that s is preferred over s', written $s \trianglelefteq s'$, if all the actions that appear in s between time 0 and time $min(start(s), start(s'))$[9] also appear in s'. The definition is:

$$\begin{aligned} legal(s) \wedge legal(s') \supset (s \trianglelefteq s' \equiv \\ (\forall a,t)\ [t \leq min(start(s), start(s')) \wedge hasOccurred(a,t,s) \supset \\ hasOccurred(a,t,s')]). \end{aligned} \tag{27}$$

[8] Here, we use the terms *event* and *action* interchangeably

[9] We take $min(r_1, r_2)$ denote the minimum real between r_1 and r_2.

Thus, if one looks at the narrative formalized with sentences (23)-(25), we have that the situation $do([\{PutIn\},\{Eat\},\{Carry\}],S_0)$[10], with time assignments:

$$\begin{array}{l} start(S_0) = 0, \\ start(do(\{PutIn\},S_0)) = 6:30am, \\ start(do([\{PutIn\},\{Eat\}],S_0))) = 6:35am, \\ start(do([\{PutIn\},\{Eat\},\{Carry\}],S_0)) = 7:00am, \end{array} \tag{28}$$

would be preferred over the legal situation $do([\{PutIn\},\{Eat\},\{TakeNotes\},\{Carry\}],S_0)$, with time assignments:

$$\begin{array}{l} start(S_0) = 0, \\ start(do([\{PutIn\}],S_0)) = 6:30am, \\ start(do([\{PutIn\},\{Eat\}],S_0)) = 6:35am, \\ start(do([\{PutIn\},\{Eat\},\{TakeNotes\},S_0])) = 6:40am, \\ start(do([\{PutIn\},\{Eat\},\{TakeNotes\},\{Carry\}],S_0)) = 7:00am. \end{array} \tag{29}$$

Also, we can define:

$$minimal(s) \equiv legal(s) \wedge (\forall s')\ [(legal(s') \wedge s' \trianglelefteq s) \supset s \trianglelefteq s'], \tag{30}$$

where *minimal* is true of a situation iff it is a most preferred situation.
Using these predicates, we have:

Observation 4.3 *Let Σ_m be the definitions (27) and (30). We have that:*

$$\Sigma \cup \Omega \cup \Sigma_m \models (\forall s)\ legal(s) \wedge 7{:}00am < start(s) \wedge minimal(s) \supset \neg(\exists t) t \leq 7{:}00am \wedge hasOccurred(TakeNotes, t, s).$$

PROOF: *From observation 4.1 we know that all legal situations that start after the Carry action will include the three actions explicitly mentioned in the narrative. Thus, if s' were a legal situation that included a TakeNotes action before 7:00 am, there would always be a preferred situation not including it. Therefore, s' would not be minimal.*

□

Thus, with our approach to handling occurrences, we are still able to do all sorts of hypothetical reasoning. But, more interestingly, we can also define preference criteria for the set of legal situations. As part of our ongoing work, we plan to explore the issue of defining a richer set of properties to describe situations. For instance, given some notion of elementary likelihood, perhaps defined in terms of agent choices, one can define a derived notion of situation likelihood. In such a language, one would be able to describe situations that are likely to arise. Given this notion of likelihood, one can devise planning systems that plan for likely courses of events first and reason about possible, but less likely, developments later.

[10]The function symbol *do* takes a concurrent action as a first argument. The set $\{PutIn\}$ is used to denote a concurrent action composed of a single action.

It should be noted that this style of minimization is not problem free. To illustrate, assume that we add a sentence of the form:

$$(\exists\, t) occurs_{by}(GetBack, t)\,,$$

with the intention of stating that the *GetBack* action must eventually occur. If one uses the minimality notion defined before, it turns out that the preferred situations would be those in which the eventually occurring action is postponed indefinitely. Thus, in this case, we do not obtain a satisfactory characterization of preferred developments. Perhaps, one might find other more refined forms of minimization to address this type of problem. However, we have not yet explored this venue.

4.1.3 Narratives and Planning

One advantage of our approach to representing occurrences is the fact that planning can be specified as a deductive problem within a first order framework (extended with induction). Following Green [8], a planning problem corresponds to the question "Does there exist a situation s that satisfies a goal description G?" If we assume that there are no occurrence specifications, in a Situation Calculus language we would like to find a constructive proof for the following theorem:

$$\Sigma \models (\exists\, s)\ G(s)\,.$$

Thus, finding a situation S for which $G(S)$ is true solves the planning problem. This solution is satisfactory because the proof would yield the specification of a situation as a sequence of actions performed in S_0 as:

$$S = do([c_1, c_2, \ldots, c_p], S_0)\,,$$

where p is the number of actions in the plan. Therefore, the theorem guarantees that after performing $c_1, c_2, \ldots, c_p$ we would reach a situation in which the goal is true. Applying the same idea to theories in the Situation Calculus language presented in this article, a planning problem is formulated as finding a constructive proof for the theorem:

$$\Sigma \cup \Omega \models (\exists\, s)\ G(s)\,. \tag{31}$$

Where Ω denotes a set of occurrence statements. Therefore, any situation that is found to conform to the goal specification would also conform to the occurrence constraints.

For instance, in the spirit of the previous example, let us assume that Mary's husband wants to play a tasteless practical joke on her. Thus, he would like a plan that would lead to a situation in which her notes don't make it to college with her. Using the definition (31) we need to specify a goal formula that characterizes the husband's goal. A first requirement is that the situation obtained be legal and that Mary's notes don't make it to school:

$$g_o(s) \equiv legal(s) \wedge \neg notesAtSchool(s)\,,$$

where *notesAtSchool* is a fluent predicate that is true in those situations in which Mary's notes are at school. Finally, the goal G can be defined as:

$$G(s) \equiv g_0(s) \wedge (\forall\, s')\ [g_0(s') \wedge s' \trianglelefteq s) \supset s \trianglelefteq s']\,, \tag{32}$$

Thus, in this case, the plan is formulated as one in which a *minimality* criterion, like the one defined in (30), is also imposed. That is, the plan that we want is obtained by making the explicit assumption that the the agent arrives at a goal situation by performing a minimal number of actions.

4.2 Scheduled Events

In this subsection we consider the following type of situation: An action A_1 triggers A_2, which would occur, if possible, Δ time units after A_1. Action A_2 cannot occur unless it has been triggered by A_1. This type of situation is common when modeling the intended behavior of certain industrial processes. For instance, every time we receive a request, we schedule an acknowledge event. The difference between the triggers we discussed previously and these scheduled events is that the latter are used to model triggered events that cannot occur unless a specific action triggers it.

To model this we introduce a fluent predicate $Scheduled \subseteq \mathcal{A} \times \mathcal{T} \times \mathcal{S}$. Initially, we may assume that there are no scheduled actions:

$$\neg Scheduled(a, t, S_0)$$

Also, for A_2 we have:

$$Poss(A_2, s) \equiv (\exists\, \delta)\ \delta > 0 \wedge Scheduled(A_2, \delta, s) \wedge \pi_{A_2}(s)$$

Thus, A_2 can only occur if it has been scheduled. $\pi_{A_2}(s)$ are further preconditions for the performance of A_2 in situation s. To complete the formalization, we need to add that:

$$trigger(A_1, A_2, \Delta), \tag{33}$$

$$A_1 \in c \supset Scheduled(A_2, \Delta, do(c, s)). \tag{34}$$

As a more specific example, consider the following information:

> My house has a burglar alarm. If the alarm is connected, I have exactly 60 seconds to deactivate it after opening the main door. If I am unable to disconnect the alarm, it will go off[11].

The axioms for this example are:

$$\neg Scheduled(a, \delta, S_0), \tag{35}$$

$$Poss(AlarmOff, s) \equiv Scheduled(AlarmOff, \delta, s) \wedge alarmConnected(s), \tag{36}$$

$$trigger(OpenDoor, AlarmOff, 60), \tag{37}$$

$$Scheduled(AlarmOff, 60, do(\{OpenDoor\}, s)). \tag{38}$$

Therefore, if the door is open, then (38) ensures that the alarm will be scheduled to go off 60 seconds after, unless the alarm is not connected at that time. To complete the formalization of *Scheduled* we need a successor state axiom[33]. This axiom is trivial in this case, since there are no actions that change the value of the fluent *Scheduled*.

4.3 Active Databases

The Situation Calculus has proven to be an extremely powerful language. Here, we argue that the language can be used effectively as a tool for the modeling and analysis of active database systems. As an illustration, consider the following example (proposed by Zaniolo [43]), where there are two relations in a relational database `EMP(E#, Ename, JobTitle, SAL, Dept#)` and `HPaid(JobTitle)`:

[11] Thanks to Michael Gruninger for suggesting this example.

> Upon an insertion into `EMP` or an update to `EMP`, the new `SAL` is checked, and if it exceeds \$100,000, then the `JobTitle` of this employee is added to `HPaid`, assuming that it was not there already.

This is straightforward to model using the language that we have described already. Indeed, following Reiter [34], each database relation is modeled as a fluent. For simplicity, we eliminate fields in the relations that are not relevant for the example and assume that the language is conveniently extended with subsorts of the sort $\mathcal{D}$[12]. We use the fluent *emp* that takes an employee number, a job-title, a salary and a situation as arguments. Thus, $emp(e\#, jt, sal, s)$ is true in the situation s if the employee with number $e\#$ has a job with title jt and salary sal. Furthermore, we introduce the fluent *HPaid* that takes a job title and a situation as arguments. Thus, $HPaid(jt, s)$ is true in s if the job title jt is highly paid in that situation. Also, we have the actions $upgradeSalary(e\#, newsal)$ and $addJobTitle(jt)$. The first action is meant to change the salary of the employee with number $e\#$ to a new salary $newsal$. The second action is used to add a new job title to the table of highly paid positions.

We assume that the action *upgradeSalary* has no preconditions. Also, we assume that $addJobTitle$ is possible if and only if the job title is not already in the table *HPaid*. This is written as:

$$Poss(upgradeSalary(e\#, newsal), s) \equiv True,$$
$$Poss(addJobTitle(jt), s) \equiv \neg HPaid(jt, s).$$

As a next step, we present successor state axioms that tell us when the fluents are true in a situation $do(c, s)$. For fluent *emp*, we have:

$$\begin{aligned} Poss(c, s) \supset emp(e\#, jt, sal, do(c, s)) \equiv \\ emp(e\#, jt, sal, s) \wedge upgradeSalary(e\#, newsal) \notin c \vee \\ upgradeSalary(e\#, sal) \in c. \end{aligned}$$

On the other hand, for *HPaid*, we have:

$$\begin{aligned} Poss(c, s) \supset HPaid(jt, do(c, s)) \equiv \\ HPaid(jt, s) \vee addJobTitle(jt) \in c. \end{aligned}$$

Finally, in order to model the *trigger* of an $addJobTitle$ event by the *upgradeSalary* event provided that the salary is over the \$100,000 threshold, we have:

$$\begin{aligned} [newsal > \$100,000 \wedge emp(e\#, jt, sal, s) \wedge upgradeSalary(e\#, newsal) \in c] \supset \\ occurs_{ct}(addJobTitle(jt), start(do(c, s)) + \Delta, do(c, s)), \end{aligned}$$

where Δ is some positive real constant.

Notice the similarity of this formula with formula (17) that defines the notion of a $trigger$. The reader might wonder why on this occasion we did not use definition (17), or a similar definition involving conditional triggers (see section (3.3.2)). The reason is that formula (17) defines triggers as context independent. In fact, the $trigger$ predicate does not include a situation argument. In this example, however, the triggered event is context dependent. In fact, the triggered

[12] To be completely rigorous we need to add sorts for employee numbers, job titles, and other domain specific sorts.

action is $addJobTitle(jt)$, and the argument jt is obtained from the context in which the triggering action is executed.

Another important detail has to do with the value of the constant Δ. An appropriate value for this example might be a small constant that guarantees that the transaction is performed immediately after or soon after the triggering event occurs. In some applications, one might need to ensure that no transaction can possibly be performed while a triggered event is not completed. This can be accommodated by adding a fluent identifying pending transactions, so that that no transactions, other than the pending ones, are possible until there are no pending transactions.

It should be clear that the axioms above guarantee that Δ units of time after an $upgradeSalary$ is performed, an $addJobTitle$ is also performed.

5 Relationship with other Approaches

5.1 Linear Time Approaches in the Situation Calculus

As mentioned before, previous approaches to handle occurrences and narratives in the Situation Calculus relied on the idea of imposing a time line as one path in the Situation Calculus tree. One of these is Pinto and Reiter's *actual line* approach. Note that the formalism of Pinto and Reiter does not deal with concurrent actions. However, in [26] Pinto and Reiter's work is extended to deal with concurrent actions. A minor variant of this formalism is used in this section. Our intent is to show that the approach presented in this paper is a generalization of the actual line approach.

Assume that we have the following axiomatization for *actual* (based on [29]):

$$actual(S_0), \tag{39}$$

$$(\forall c,s).actual(do(c,s)) \supset actual(s) \wedge Poss(c,s), \tag{40}$$

$$(\forall c_1,c_2,s).actual(do(c_1,s)) \wedge actual(do(c_2,s)) \supset c_1 = c_2. \tag{41}$$

Furthermore, we have the predicate $occurs_{\mathcal{T}}$ which is true for an action and a time point, if the action occurs at the specified time point:

$$occurs_{\mathcal{T}}(a,t) \equiv (\exists c,s).a \in c \wedge actual(do(c,s)) \wedge start(do(c,s)) = t. \tag{42}$$

Assume that a narrative is represented using a set Ω of $occurs_{np}$ literals. To establish a relationship between our current approach to occurrences and the actual line approach, we add:

$$occurs_{np}(a,t) \supset occurs_{\mathcal{T}}(a,t). \tag{43}$$

We have:

Theorem 5.1 *Let Σ_a denote axioms of (1)-(16), (22) and (39)-(43). Also, let Ω denote a narrative expressed as $occurs_{np}$ literals. From $\Sigma_a \cup \Omega$ we can infer that the actual line of situations must be legal. Thus:*

$$\Sigma_a \cup \Omega \models (\forall s)\ actual(s) \supset legal(s)$$

PROOF: Follows from the definitions. Intuitively, (43) forces the actual line to conform to the legality as defined by axiom (15).

□

Theorem 5.2 *Let $\mathcal{M}$ be a model of Σ_a (as defined above). Also, let $\sigma_{\mathcal{S}}[\cdot]$ be an assignment function for situation variables. The following holds:*
If

$$\mathcal{M}, \sigma_{\mathcal{S}}[s] \models legal(s)$$

then, there is a model $\mathcal{M}'$ that interprets everything the same except for actual and $occurs_{\mathcal{T}}$, such that:

$$\mathcal{M}', \sigma_{\mathcal{S}}[s] \models actual(s).$$

PROOF: The proof is done by building an interpretation $\mathcal{M}'$ just like $\mathcal{M}$, except for the interpretation of *actual*. The interpretation of *actual* in $\mathcal{M}'$ can be set in such a way that $\sigma_{\mathcal{S}}(s)$ is actual. Given that $\sigma_{\mathcal{S}}(s)$ belongs to the interpretation of *legal* it must be the case that $\mathcal{M}'$ satisfies the axioms for *actual*.

□

An important consequence of this theorem is that our theory is categorical for *actual* if and only if there is a unique *legal* path in the situation tree. This theorem allows us to claim that our approach to occurrences subsumes Pinto and Reiter's approach. In fact, a consequence of the theorem is that if an action can be proven to $occurs_{\mathcal{T}}$ at a certain time point, then this action must be present in every legal history that includes that time point. As discussed in the introduction, Shanahan showed that there is an equivalence, under certain conditions[13], between Miller and Shanahan's characterization of a time line and Pinto and Reiter's. Therefore, under the same conditions, our new approach to occurrences should also subsume Miller and Shanahan's approach.

To date, there have been several proposals in which the branching and linear views of time are integrated within a single logical framework [25, 30, 32, 41, 2]. We suspect that these logical frameworks have equivalent expressive power. If this were the case, then the results of the greater generality of our framework (with respect to Pinto and Reiter's approach) should also be applicable to the other approaches. Namely, those based on the Event Calculus and based on the $\mathcal{L}_1$ variant of language $\mathcal{A}$. However, a formal comparison between the aforementioned logical frameworks is a prerequisite. Notice that in Kowalski and Sadri's [10] comparison of the Event Calculus and the Situation Calculus does not extend to enriched versions of both languages. In particular, it does not consider an Event Calculus extended with a branching temporal structure.

5.2 Hypothetical Reasoning in Linear Time Theories

In [24], Rob Miller proposes a novel approach to perform deductive-style planning in the Event Calculus [11]. Miller's work is based on Shanahan's Circumscriptive Event Calculus [39]. The Circumscriptive Event Calculus solves the frame problem by appealing to Circumscription [20, 15]. The language of the Circumscriptive Event Calculus includes sorts for actions, fluents, time-points, and domain objects. Important predicates are *Initiates* and *Terminates* which are ternary predicates with actions, fluents and time-points as arguments. These predicates are used in order to write domain axioms to express effects that actions have on fluents, when the actions are performed at certain time points. If a fluent holds before a time point and an action terminates it (i.e., it makes the fluent change from holding to not holding) at that point, then the fluent will not hold after, unless another action initiates it at a later point. Also, a binary predicate *Happens* for

[13] The conditions refer to the form in which circumscription axioms are applied in both formalizations. The details are beyond the scope of this paper, and the reader is referred to [40, chapter 9] for further details.

action and time points is used to mean that an action happens at a time point. A full description of the language would take us too far afield and is not necessary for our present discussion.

At the core of Miller's proposal is the introduction of a distinction between actions that *occur*, and that are external to the agent being modeled, and the actions that are *performed* by the agent. Thus, Miller introduces the axiom:

$$Happens(a,t) \equiv [Occurs(a,t) \vee Perform(a,t)].$$

The domain axioms would only include specifications for narratives expressed in terms of *occurrences*. Miller proceeds to specify a circumscription policy that involves the separate minimization of the *Initiates* and *Terminates* predicates in the context of the domain effect axioms. Also, he minimizes *Occurs* in the context of the axioms for *Narratives*. The final circumscription policy involves other minimizations as well. An important feature of the circumscription policy is that it leaves out *Happens* and *Perform*. Therefore, the models of a Miller style Event Calculus theory are minimal in the extension of *Occurs* but leave the interpretation of *Perform* free. The intent of such a policy is to characterize models in which the events that occur externally to the agent are minimal (i.e., the things that occur are only those that must occur, as dictated by the narratives), whereas the actions performed by the agent are constrained only by the action preconditions.

A plan $\mathcal{P}$ for a time τ, in Miller's framework, is a formula of the form

$$Perform(a,t) \equiv ((a = \alpha_1 \wedge t = \tau_1) \vee \ldots \vee (a = \alpha_n \wedge t = \tau_n),$$

where the α_i's τ_i's are ground terms, such that each τ_i is greater than τ, where τ is the time for which the plan has been designed. Therefore, a plan specifies a complete set of actions, and their times, that the agent has to perform. If $\mathcal{G}$ denotes a goal, then a valid plan $\mathcal{P}$ is such that the theory entails

$$\mathcal{P} \supset \mathcal{G}.$$

Since *Perform* does not participate in the Circumscription policy, the space of possible plans is only limited by the action precondition axioms.

Furthermore, $\mathcal{P}$ must be satisfiable. However, Miller shows that the test for satisfiability is not necessary for theories of a certain general form.

Finally, hypothetical reasoning about possible plans would involve the use of implications like the one above. Thus, if one wants to reason about a possible plan, one needs to look at the models in which that plan is carried out. This works only if the plans that are entertained are satisfiable. This characterization of hypothetical reasoning is possible due to the circumscription policy that was selected.

There are two important differences between Miller's approach and the one we have presented in this article, namely:

- Our approach deals with the frame problem, and with the problem of specifying action preconditions, by appealing to *successor state axioms* and *if and only if* completions of the precondition axioms. Thus, the use of circumscription is avoided. On the other hand, Miller's approach is circumscriptive. The second order nature of circumscription needs not be a problem since, in some cases, it can translate into some equivalent first order formula. Unfortunately, the use of circumscription leads to the *premature minimization* problem, discussed in the introduction. Indeed, the minimization of *Occurs* precludes one

from entertaining situations in which *unnecessary*[14] events do occur. Although, one can reason hypothetically about the actions that can be performed by the agent.

An important advantage of the circumscriptive approach is that the solution to the frame problem is more general and can be applied to theories that include state constraints as well as effect axioms. However, the correctness of solutions based on circumscription is hard to assess.

- We do not need to give a different treatment to actions performed by the agent and actions that occur externally. In fact, minimizing *Occurs* separately makes it difficult to include occurrences that are triggered by some action performed by the agent. However, this is not a problem if one characterizes triggers as a relation between world properties and triggered actions as done in [23].

6 Conclusion

A great deal of research in knowledge representation about dynamic domains has concentrated on the modeling of the effects that actions have on the properties of the world. In particular, the frame problem motivated the development of logical frameworks in which certain common-sense assumptions could be formalized. As Shanahan remarks [40], "... like the mind body problem in philosophy, there's no universal agreement as to what would constitute a solution." Furthermore, Shanahan also suggests that "... we are very close to a complete solution to the frame problem." While this might seem contradictory on first reading, the claim is substantiated by the fact that we have at our disposal a rich set of theoretical tools to deal with the problem.

Most solutions to the frame problem involve the use of some sort of minimal model semantics. In contrast, our work is based on an inductive solution to the frame problem (due to Raymond Reiter [33]). However, this solution is not completely general and more work is needed in order to extend it to deal with general theories. For instance, we need to extend this solution to domains in which actions have uncertain effects.

However, if we were to assume that the frame problem is solved, then we need to worry about the problem of writing theories of the world in which we take into account the free will of the agents that populate it. We need languages in which one could characterize how these agents behave. This language should allow us to plan for courses of actions that take into account certain *common-sense assumptions* that we must make in order to plan. If we are unable to make these assumptions, then planning is not possible. For example, my plan to fly to Toronto is reasonable given that I assume that the agents in the world behave according to certain patterns. For instance, I assume that the airplane pilot will not intentionally crash the plane, etc.

While our approach is first order with induction, the formalization of assumptions of agent-behavior might require more expressive languages. A different approach could be based on second-order circumscriptive axiomatizations. For instance, Miller's work, discussed in section 5.2, goes in this direction, Miller is seeking for a balance between what can be constrained or minimized (external occurrences) and what should be allowed to be free (the agent's actions). This research needs to evolve in order for us to provide support for the characterization of the evolution of the world. This characterization will surely involve common-sense assumptions, without which our agents would be paralyzed.

[14] Unnecessary from the point of view of the axiomatization

In this article we present a formal framework that allows one to deal with occurrences in a general form. In particular, the specification of action occurrences constrains the histories that can be considered legal in a branching time structure. We are currently investigating the use of this framework to model agent behavior. For instance, statements of the form "If a message is received from some agent , then a reply must be sent to that agent" can be written as triggering statements. As a first experiment, we are modeling agent interaction through communication protocols, where the agents are constrained to react to events with actions specified by the protocol.

Also, the approach presented allows one to reason about hypothetical developments which obey all occurrence constraints imposed by the occurrence axioms. An essential aspect of our approach is that we rely on the notion of *history*. This notion is equivalent to the notion of *chronicle* in Drew McDermott's *Temporal Logic for Reasoning about Processes and Plans* [22]. To sum up, in this paper we:

1. Build upon a solution to the frame problem in the Situation Calculus based on a first order language extended with induction.
2. Extend the language with explicit references to actions in histories.
3. Use histories to define external occurrences.
4. Define different notions of occurrences that can be used to express conditional occurrences, triggers, narratives, etc.

There are several important extensions to this work that we would like to pursue, namely:

- Add the ability to deal with observations. That is, we would like to study the effect of adding statements regarding the values of fluents at different points in time.
- In this article we consider triggers to be relationships between two actions (one triggering the other). We would like to investigate how to incorporate actions that are triggered by circumstances (i.e., propositional combinations of fluent values), as in [23].
- Complex occurrences: We would like to consider occurrence statements involving complex actions (actions that involve conditionals, loops, etc.).
- We would like to investigate how to integrate this approach to occurrences with the current work on the GOLOG language[14] being developed at the Cognitive Robotics group in the University of Toronto. Also, it is important to analyze the relationship between the notion of interrupts in GOLOG with our notion of occurrence.
- In this paper we do not address the computational issues of reasoning. It is likely that tasks like *plan verification* will be difficult. In particular, non-preventable occurrences may introduce difficulties in proving the legality of situations. The reason is that one might be forced to think about the future of a situation in order to determine its legality. This difficulty does not always arise, and we would like to characterize classes of theories for which this reasoning is easy.
- Planning: The theoretical advances that have been made in knowledge representation using the Situation Calculus provide a framework in which planning can be extended to richer

application areas. For instance, in [13], Levesque studies the formal specification of planning problems in the presence of sensing in a Situation Calculus extended with the ability to incorporate knowledge.

The work presented here allows for the incorporation of *exogenous events*, which can be formalized using occurrence statements. Thus, we would like to extend this work by providing a characterization of planning in the presence of events that are external to the agent that is doing the planning. For instance, it would be interesting to be able to come up with a plan for an agent to go to the airport knowing that the bus to the airport leaves for the airport every hour on the hour. However, we should not be able to prove that the plan is successful unless we assume that the bus does not run into problems. A plan, then, should be provably correct only under explicit assumptions regarding the behavior of external agents.

- In [30], it was shown that the Situation Calculus with actual line subsumes the temporal logic of concurrency [7]. We would like to further explore the issue of expressiveness of the Situation Calculus in comparison to the expressiveness found in conditional and other modal logics.

References

[1] ALLEN, J. F. Towards a General Theory of Action and Time. *Artificial Intelligence 23* (1984), 123–154.

[2] BARAL, C., GELFOND, M., AND PROVETTI, A. Reasoning about actions: Laws, observations and hypotheses. *Journal of Logic Programming* (1997). To appear.

[3] BERTOSSI, L., PINTO, J., SAEZ, P., KAPUR, D., AND SUBRAMANIAM, M. Automating Proofs of Integrity Constraints in Situation Calculus. In *Lecture Notes in Artificial Intelligence: Proceedings of the 9th International Symposium on Methodologies for Intelligent Systems* (1996), Z. Pawlak and Z. W. Ras, Eds., Springer-Verlag, pp. 212–222.

[4] ESHGHI, K. Abductive Planning With Event Calculus. In *Proceedings of the Tenth International Conference on Logic Programming* (1988), MIT Press, pp. 562–579.

[5] GELFOND, M., AND LIFSCHITZ, V. Representing Action and Change by Logic Programs. *The Journal of Logic Programming 17* (1993), 301–322.

[6] GELFOND, M., LIFSCHITZ, V., AND RABINOV, A. What are the Limitations of the Situation Calculus? In *Working Notes, AAAI Spring Symposium Series. Symposium:Logical Formalization of Commonsense Reasoning.* (Mar. 1991), pp. 59–69.

[7] GOLDBLATT, R. *Logics of Time and Computation*. CSLI, 1987.

[8] GREEN, C. C. Applications of Theorem Proving to Problem Solving. In *Proceedings of the First International Joint Conference on Artificial Intelligence (IJCAI-69)* (1969), pp. 219–239. Reprinted in *Readings in Artificial Intelligence*, Webber, B. L. and Nilsson, N. J., editors, Tioga Publishing Co., Los Altos, California, pp. 202-222, 1981.

[9] KAPUR, D., AND ZHANG, H. RRL: A Rewrite Rule Laboratory. In *Proc. Ninth International Conference on Automated Deduction (CADE-9), LNCS 310* (1988), no. 310 in LNCS, Springer-Verlag.

[10] KOWALSKI, R., AND SADRI, F. The Situation Calculus and Event Calculus Compared. In *Proceedings of the International Logic Programming Symposium, Ithaca, New York* (November 1994), M. Bruynooghe, Ed., The MIT Press, pp. 539–553.

[11] KOWALSKI, R., AND SERGOT, M. A Logic-based Calculus of Events. *New Generation Computing 4* (1986), 67–95.

[12] LESPÉRANCE, Y., LEVESQUE, H., LIN, F., MARCU, D., REITER, R., AND SCHERL, R. A Logical Approach to High-Level Robot Programming – A Progress Report. In *Control of the Physical World by Intelligent Systems, Papers from the 1994 AAAI Fall Symposium*, B. Kuipers, Ed. AAAI, New Orleans, November 1994, pp. 79–85.

[13] LEVESQUE, H. What is planning in the presence of sensing. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-96)* (1996).

[14] LEVESQUE, H. J., REITER, R., LESPÉRANCE, Y., LIN, F., AND SCHERL, R. B. GOLOG: A Logic Programming Language for Dynamic Domains. *The Journal of Logic Programming* (1996). To appear.

[15] LIFSCHITZ, V. Computing Circumscription. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)* (1985), pp. 121–127.

[16] LIN, F., AND REITER, R. State Constraints Revisited. *Journal of Logic and Computation 4*, 5 (1994). Special Issue on Actions and Processes.

[17] LIN, F., AND SHOHAM, Y. Concurrent Actions in the Situation Calculus. In *Working Notes of the 4th International Workshop on Nonmonotonic Reasoning* (1992), pp. 133–138.

[18] MANNA, Z., AND PNUELI, A. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, New York, 1992.

[19] MCCARTHY, J. Programs with Common Sense. In *Semantic Information Processing*, M. Minsky, Ed. The MIT Press, 1968, ch. 7, pp. 403–418.

[20] MCCARTHY, J. Applications of Circumscription to Formalizing Commonsense Knowledge. *Artificial Intelligence 28* (1986), 89–116.

[21] MCCARTHY, J., AND HAYES, P. J. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In *Machine Intelligence* **4**, B. Meltzer and D. Michie, Eds. Edinburgh University Press, Edinburgh, Scotland, 1969, pp. 463–502.

[22] MCDERMOTT, D. A Temporal Logic for Reasoning About Processes and Plans. *Cognitive Science 6* (1982), 101–155.

[23] MILLER, R. A Case Study in Reasoning about Actions and Continuous Change. In *Proceedings ECAI* (1996), John Wiley & Sons.

[24] MILLER, R. Notes on deductive and abductive planning in the event calculus. URL = http://www.dcs.qmw.ac.uk/˜rsm/abstract14.html, July 1996.

[25] MILLER, R., AND SHANAHAN, M. Narratives in the Situation Calculus. *The Journal of Logic and Computation 4*, 5 (1994), 513–530.

[26] PINTO, J. *Temporal Reasoning in the Situation Calculus.* PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, Feb. 1994. URL = ftp://ftp.cs.toronto.edu/~cogrob/jpThesis.ps.Z.

[27] PINTO, J. Integrating Discrete and Continuous Change in a Logical Framework. *Computational Intelligence 14*, 1 (1997). To appear.

[28] PINTO, J. Ramification constraints revisited. In progress.
URL = ftp://lyrcc.ing.puc.cl/pub/jpinto/rcnew.ps.gz, 1997.

[29] PINTO, J., AND REITER, R. Temporal Reasoning in Logic Programming: A Case for the Situation Calculus. In *Proceedings of the Tenth International Conference on Logic Programming* (Budapest, June 1993), D. S. Warren, Ed., The MIT Press, pp. 203–221.

[30] PINTO, J., AND REITER, R. Reasoning about Time in the Situation Calculus. *Annals of Mathematics and Artificial Intelligence 14*, 2-4 (September 1995), 251–268.

[31] PINTO, J. A. On the Existence and Formalization of Natural Events. In *Proceedings of the IJCAI Workshop on Nonmonotonic Reasoning, Action and Change.* (Aug. 1995). URL = ftp://lyrcc.ing.puc.cl/pub/jpinto/ijcwrk.ps.gz.

[32] PROVETTI, A. Hypothetical Reasoning about Actions: From Situation Calculus to Event Calculus. *Computational Intelligence 12*, 2 (1995).

[33] REITER, R. *The Frame Problem in the Situation Calculus: A Simple Solution (sometimes) and a completeness result for goal regression*. Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy. Academic Press, San Diego, CA, 1991, pp. 359–380.

[34] REITER, R. Formalizing database evolution in the situation calculus. In *Proceedings of the International Conference on Fifth Generation Computer Systems* (June 1992), pp. 600–609.

[35] REITER, R. Proving Properties of States in the Situation Calculus. *Artificial Intelligence 64*, 2 (Dec. 1993), 337–351.

[36] REITER, R. Natural Actions, Concurrency and Continuous Time in the Situation Calculus. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference(KR'96)* (Cambridge, Massachussetts, U.S.A., Nov. 1996).

[37] SCHERL, R., AND LEVESQUE, H. The Frame Problem and Knowledge Producing Actions. In *Proceedings AAAI-93* (Washington, D.C., July 1993), AAAI, pp. 689–695.

[38] SCHUBERT, L. Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions. In *Knowledge Representation and Defeasible Reasoning*, H. Kyberg, R. Loui, and G. Carlson, Eds. Kluwer Academic Press, 1990, pp. 23–67.

[39] SHANAHAN, M. A Circumscriptive Calculus of Events. *Artificial Intelligence 77*, 2 (September 1995), 249–284.

[40] SHANAHAN, M. P. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. MIT Press, 1996. To appear.

[41] VAN BELLEGHEM, K., DENECKER, M., AND SCHREYE, D. D. Combining Situation Calculus and Event Calculus. In *Proceedings of the Twelfth International Conference on Logic Programming* (1995), pp. 83–97.

[42] WIDOM, J., AND CERI, S. *Active Database Systems, Triggers and Rules for Advanced Database Processing*. Morgan Kaufmann Publishers, Inc., 1996.

[43] ZANIOLO, C. Active database rules with transaction-conscious stable-model semantics. In *Lecture Notes in Computer Science: Deductive and Object-Oriented Databases* (Dec. 1995), T. W. Ling, A. Mendelzon, and L. Vieille, Eds., Springer Verlag.