Non-instantaneous Actions and Concurrency in the Situation Calculus (Extended Abstract)

Jorge A. Baier and Javier A. Pinto*
Departamento de Ciencia de la Computación, Escuela de Ingeniería,
Pontificia Universidad Católica de Chile, Casilla 306, Santiago, CHILE
{jabaier, jpinto}@ing.puc.cl

1 Introduction

When representing knowledge about actions and change, it is essential to have the ability to describe actions that have temporal duration. In some languages in which there is a linear view of time (e.g., Allen's temporal logic [1, 2]) actions with durations are elementary components of the underlying ontology. In contrast, in languages where time can be seen as a branching structure, actions are usually considered to be instantaneous.

In this article, our goal is to extend the language of the situation calculus with non-instantaneous actions. This goal was already pursued by Gelfond, Lifschitz and Rabinov in [3]. We develop on ideas proposed by Reiter in [9] and Pinto and Reiter in [7]. Our main contribution is to deal with this ontological extension to the situation calculus within the framework of Reiter's solution to the frame problem [8].

2 Theoretical Framework

We use the Situation Calculus [4] with the sorts $\mathcal{A}$, $\mathcal{C}$, $\mathcal{A}_d$, $\mathcal{S}$, and $\mathcal{T}$, for primitive actions, concurrent actions, actions with duration, situations and time. The sorts of the variables a, c, a_d, s and t respectively correspond to the ones introduced. The free variables in all formulas are implicitly universally prenex quantified. We assume that the sort $\mathcal{T}$ is interpreted to be the positive real numbers, which we do not axiomatize.

There is a distinguished constant S_0 which denotes the initial situation. The function symbol $do : \mathcal{C} \times \mathcal{S} \rightarrow \mathcal{S}$ defines successor situations, such that $do(c, s)$ is the resulting situation from executing the concurrent action c in s. The function $start : \mathcal{S} \rightarrow \mathcal{T}$ defines the start of situations, that is, the point in time in which the situation begins. The relation $\prec \subseteq \mathcal{S} \times \mathcal{S}$ defines

*This work has been partly funded by the Fondecyt project number 197-1304.

a structural ordering between situations (is used with an infix notation). The predicate $Poss \subseteq (\mathcal{A} \cup \mathcal{C}) \times \mathcal{S}$ is such that $Poss(c, s)$ holds when a concurrent action c is possible in the situation s. This predicate can also take a primitive action as first argument.

The foundational axioms for the concurrent Situation Calculus are:

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

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

$$\neg s \prec S_0, \tag{3}$$

$$s \prec do(c, s') \equiv s \preceq s', \tag{4}$$

$$start(s) < start(do(c, s)), \tag{5}$$

$$start(S_0) = 0. \tag{6}$$

Here, we use $s \preceq s'$ as a shorthand for $s \prec s' \vee s = s'$. The start of S_0 has been assigned arbitrarily. A concurrent action is a set of primitive actions. We will consider that a concurrent action is possible if all the primitive actions which constitute it are possible. Furthermore, the empty action is never possible.

$$Poss(c, s) \equiv [(\forall a).(a \in c \supset Poss(a, s)) \wedge c \neq \{\}]. \tag{7}$$

3 Definition of Occurrences

We will make use of some occurrence predicates defined in [6]. A central observation in [6] is that situations can be regarded as histories, which include all the actions that have taken place since the start of time (S_0). In the rest of this article, elements of sort $\mathcal{S}$ will be referred to as situations as well as histories.

A concurrent action c has occurred in a history s at time t if there exists a situation in the past of s where c was executed and its start is equal to t.

$$hasOccurred(c, t, s) \equiv (\exists s')\, do(c, s') \preceq s \wedge start(do(c, s')) = t. \tag{8}$$

The predicate can be extended for simple actions in the following way:

$$hasOccurred(a, t, s) \equiv (\exists\, c)\ (hasOccurred(c, t, s) \wedge a \in c). \tag{9}$$

The predicate $Sit(t, s, s_h)$ holds when situation s is in the history of s_h and t is between the start and end of s.

$$Sit(t, s, s_h) \equiv (\exists c)\ do(c, s) \preceq s_h \wedge start(s) < t \wedge t \leq start(do(c, s)). \tag{10}$$

A situation is said to be $legal_{poss}$ if its history only contains actions that are performed in situations in which they are possible.

$$legal_{poss}(s) \equiv s = S_0 \vee (\forall\, c, s)\, do(c, s') \preceq s \supset Poss(c, s'). \tag{11}$$

On the other hand, further restrictions on the tree of situations are required. The restrictions imposed by $legal_{poss}$ are not sufficient and need to be strengthened. To this end, we define the predicate *legal*. A situation s is *legal* if there exists a situation in its future which is *proper*.

$$legal(s) \equiv ((\forall t)\, start(s) < t \supset (\exists s')s \prec s' \wedge t < start(s') \wedge proper(s')). \tag{12}$$

A situation s is legal if there exists a situation in its future which is *proper*. The *proper* predicate *prunes* the branching structure of the Situation Calculus. For instance, if a certain event occurs at some time t then all the situations including t in their history, in which the event has occurred at time t are proper. In this, and the following sections, we will define several necessary conditions for *proper*. In order to be able to conclude that some situation is *proper* we will assume that all necessary conditions are sufficient. All proper situations must be $legal_{poss}$:

$$proper(s) \supset legal_{poss}(s). \tag{13}$$

Action occurrences are modeled with occurrence predicates. For instance, some actions may *preventably trigger* other actions. Consider, for example, an action *Drop*, which will preventably trigger *HitGround* in a time T after it is performed. If there exists an action *Catch*, and it is executed before T units of time, *HitGround* will not necessarily occur. Thus, the notion of prevention. Moreover, we do not want *HitGround* to occur before T units of time. To model this kind of behavior we extend the language with the predicate $occurs_{nb} \subseteq \mathcal{A} \times \mathcal{T} \times \mathcal{S}$. When $occurs_{nb}(a, t, s)$ is true, a occurs at time t in the future of s only if it is possible, but not before t.

$$\begin{aligned} proper(s) \supset [&occurs_{nb}(a, t, s') \wedge t < start(s) \wedge s' \prec s_t \wedge \\ &Sit(t, s_t, s) \wedge Poss(a, s_t) \supset \\ &\quad hasOccurred(a, t, s) \wedge \\ &\quad ((\forall t')start(s') < t' < t \supset \neg hasOccurred(a, t', s))]. \end{aligned} \tag{14}$$

Now we can define $trigger_{nb} : \mathcal{A} \times \mathcal{A} \times \mathcal{T}$, such that $trigger_{nb}(a_1, a_2, \delta)$ holds when a_1 triggers a_2 only after δ units of time.

$$trigger_{nb}(a_1, a_2, \delta) \equiv [(\forall c, s)\, a_1 \in c \supset occurs_{nb}(a_2, [start(do(c, s)) + \delta], do(c, s))]. \tag{15}$$

4 Modeling Actions with Duration

Actions with duration can be modeled as pairs of instantaneous actions. The first element of the pair initiates the execution of the action with duration and the other one finishes it. If the *beginning action* is executed in a situation s_1, and the *finishing action* is executed in s_2 and $s_1 \preceq s_2$ we say that the action was performed along the path determined by s_1 and s_2, and that its duration is $start(s_2) - start(s_1)$. In contrast to [3], the duration of the action does not determine the duration of the situation in which the action with duration started.

As soon as the execution of an action is started, certain fluents become true, e.g. *the action is being executed*, and others false as a direct effect of the *beginning action*. Usually, the truth values of those fluents will remain constant along the execution of the action.

The action $begin(a_d)$ will begin the execution of the action a_d, and $finish(a_d)$ will finish it, both are instantaneous.

The execution of an action, with or without duration, might depend on whether another action with duration is being carried out. Then, we would need to know if there are actions being executed.

For this purpose, we introduce the propositional fluent $executing(a_d, s)$ intended to hold when the execution of the action a_d is in progress during the situation s. The effect axioms for this fluent are:

$$Poss(c, s) \wedge begin(a_d) \in c \supset executing(a_d, do(c, s)). \tag{16}$$
$$Poss(c, s) \wedge finish(a_d) \in c \supset \neg executing(a_d, do(c, s)). \tag{17}$$

We do not want to begin the execution of an action if it is already in progress or finish the execution of another if it is not being executed, that is to say:

$$Poss(begin(a_d), s) \supset \neg executing(a_d, s) \tag{18}$$
$$Poss(finish(a_d), s) \supset executing(a_d, s) \tag{19}$$

Now we can use completion to derive necessary and sufficient conditions for $Poss(begin(A_d(\bar{x})), s)$ and $Poss(finish(A_d(\bar{x})), s)$.

Theorem 1 *If an action c is possible then $begin(a_d)$ and $finish(a_d)$ cannot be part of c. Thus,*

$$Poss(c,s) \supset \neg(begin(a_d) \in c \wedge finish(a_d) \in c).$$

PROOF: *By cases using* (18) *and* (19). $\square$

It is necessary to guarantee that any action that has started will end. Thus, the beginning of an action will *trigger* its end. The specification may change, depending on the type of action being modeled. Consider this example from [3]: *"When you press the button of the hand-drying machine, it emits hot air for 30 seconds"*. The action $BlowAir$ is in this case an action with duration that takes exactly 30 seconds of execution. To assert this fact, we may add the following axiom:

$$trigger_{nb}(begin(BlowAir), finish(BlowAir), 30sec),$$

i.e. the $finish(BlowAir)$ is triggered by $begin(BlowAir)$ exactly 30 seconds after the latter action is executed.

4.1 Properties of the paths of action executions

Let $\varphi^e_{a_d}$ be a formula which determines the necessary conditions for an action a_d. We will suppose this formula is simple in s, and that it is a conjunction of fluents. Then, for a particular action A_d we add the following integrity constraint:

$$executing(A_d(\bar{x}), s) \supset \varphi^e_{A_d}(A_d(\bar{x}), s), \tag{20}$$

to express the fact that $\varphi^e_{A_d}(A_d(\bar{x}), s)$ must be true along the execution of $A_d(\bar{x})$. If for some reason $\varphi^e_{A_d}$ ceases to hold we can deduce that A_d has finished.

This is not sufficient to guarantee that the action $finish(a_d)$ will always occur as soon as executing becomes false. We need this because as a consequence of executing $finish$, the truth values of some fluents may change. For this purpose, we add the following constraint for proper situations:

$$\begin{aligned} proper(s_h) \supset [do(c,s) \preceq s_h \wedge executing(a_d, s) \wedge \\ \neg executing(a_d, do(c,s)) \supset finish(a_d) \in c]. \end{aligned} \tag{21}$$

Following [5], we can generate new effect axioms for all the fluents in $\varphi^e_{a_d}$ and for $executing$.

All these effect axioms describe when an action changes a fluent's truth value from one situation to the successor one but they do not describe when its truth value does not change. Fortunately, we can use Reiter's simple solution to the frame problem [8] to automatically derive *successor state axioms* to solve this problem.

One may want to have predicates to express relationships between actions without explicitly making reference to the beginning and finish of an action. For example, one may want to say that an action (instantaneous or with duration) was executed during the execution of an action with duration. We introduce the predicate $during \subseteq \mathcal{A}_d \times \mathcal{A} \times \mathcal{S}$, where $during(a_d, a, s)$ means an instantaneous action a was executed amid the execution of a_d in a history s. The definition follows,

$$\begin{aligned} during(a_d, a, s_h) \equiv & (\exists s, s', c, c').\, do(c, s) \prec do(c', s') \preceq s_h \wedge begin(a_d) \in c \wedge a \in c' \wedge \\ & \neg(\exists c'', s'')\, do(c, s) \prec do(c'', s'') \prec do(c', s') \wedge finish(a_d) \in c''. \end{aligned} \tag{22}$$

Example 1 This example is a variation of one presented in [1]. There is a door designed in a way that it requires to be unlocked before opening it. It has a lock that must be held open while the door is pulled. The door is initially closed and locked. Additionally, the door automatically locks the door 1 minute after it has been unlocked.

There is one action with duration: $HoldLock$ and an instantaneous action $PullDoor$. We will consider that the beginning and finish of the action $HoldLock$ have additional preconditions. Thus, the precondition for the start and end of $HoldLock$ are,

$$Poss(begin(HoldLock), s) \equiv \neg executing(HoldLock, s), \tag{23}$$

$$Poss(finish(HoldLock), s) \equiv executing(HoldLock, s). \tag{24}$$

The precondition to open the door is that it must be *unlocked*:

$$Poss(PullDoor, s) \equiv unlocked(s). \tag{25}$$

For $PullDoor$ we have the following effect axiom:

$$Poss(c, s) \wedge PullDoor \in c \supset open(s). \tag{26}$$

Once $HoldLock$ is finished, the door becomes *unlocked*:

$$Poss(c, s) \wedge finish(HoldLock) \in c \supset \neg unlocked(do(c, s)). \tag{27}$$

The necessary conditions $\varphi^e_{HoldLock}$ are defined as follows:

$$\varphi^e_{HoldLock} =_{def} unlocked(s). \tag{28}$$

From there, we define the ramification constraint,

$$executing(HoldLock, s) \supset unlocked(s) \tag{29}$$

Following [5], we generate the following effect axioms for this constraint:

$$Poss(c, s) \wedge begin(HoldLock) \in c \supset unlocked(do(c, s)), \tag{30}$$

$$Poss(c, s) \wedge finish(HoldLock) \in c \supset \neg executing(HoldLock, do(c, s)). \tag{31}$$

Using [8] we derive the following successor state axioms:

$$\begin{aligned} Poss(c, s) \supset [unlocked(do(c, s)) \equiv & begin(HoldLock) \in c \vee \\ & unlocked(s) \wedge \neg finish(HoldLock) \in c] \end{aligned} \tag{32}$$

$$Poss(c, s) \supset [open(do(c, s)) \equiv PullDoor \in c \vee open(s)] \tag{33}$$

$$\begin{aligned} Poss(c, s) \supset [& executing(HoldLock, do(c, s)) \equiv \\ & begin(HoldLock) \in c \vee \\ & executing(HoldLock, s) \wedge \neg finish(HoldLock, s) \in c] \end{aligned} \tag{34}$$

The door automatically unlocks after 1 min.:

$$trigger_{pnb}(begin(HoldLock), finish(HoldLock), 1\text{min}). \tag{35}$$

The initial situation is characterized with:

$$\neg open(S_0) \wedge \neg unlocked(S_0) \wedge \neg executing(HoldLock, S_0). \tag{36}$$

The following are unique name axioms for simple actions:

$$begin(a_d) \neq PullDoor \tag{37}$$

$$finish(a_d) \neq PullDoor \tag{38}$$

Proposition 1 *Let Σ be a theory containing all the precondition, frame, legality, unique name, initial description and foundational axioms. Then,*

$$\Sigma \models (\forall s).legal(s) \wedge open(s) \supset during(HoldLock, PullDoor, s).$$

I.e., if there exists a situation where the door is opened, then in some moment in the past of s the door was opened wile holding the the lock.

PROOF: *Omitted due to lack of space.*

5 Narratives and Non-instantaneous Actions

In this section we explore the specification of narratives which involve actions with duration. A narrative is a possibly incomplete description of the evolution of the world. It comprises a specification of actions or events known to occur at some point in time.

Consider the following narrative *"Today I began to drink coffee at 7:00 am. While I was drinking, the telephone rang at 7:05 am"*.

To model this kind of narratives it is useful to introduce the predicate $when \subseteq \mathcal{A}_d \times \mathcal{A} \times \mathcal{T}$, such that $when(a_d, a, t)$ means that a occurred at time t during the execution of the action with duration a_d.

In any legal history, if $when(a_d, a, t)$ then a must have occurred at time t during the execution of a_d.

$$\begin{aligned} proper(s) \supset [when(a_d, a, t) \wedge t < start(s) \supset \\ & hasOccurred(a, t, s) \wedge \\ & (\exists t').\, hasOccurred(begin(a_d), t', s) \wedge t' < t \wedge \\ & \neg(\exists t'') hasOccurred(finish(a_d), t'', s) \wedge t' < t'' < t] \end{aligned} \tag{39}$$

In the example, the narrative can be defined by $occurs_{nb}(DrinkCoffee, 7:00am) \wedge when(DrinkCoffee, Ring, 7:05am)$.

For actions with duration that occur during actions with duration, i.e. that can overlap, we can extend the while predicate such that $when \subseteq \mathcal{A}_d \times (\mathcal{A} \cup \mathcal{A}_d) \times \mathcal{T}$.

$$when(a_d, a'_d, t) \equiv when(a_d, begin(a'_d), t) \tag{40}$$

6 Conclusions

In this work we have proposed a method to deal with actions with duration. We have given trigger definitions that can be applied to a great number of problems in which non instantaneous actions appear. Additionally we have shown how we can define narratives involving these actions.

The main difference between our work and that of [3] is the ability to simply represent concurrency of actions. In our approach, the duration of the action is not inherent to the action itself but to the behavior of the agent or of the world. The determination of the truth values of fluents at a certain point in time in a history is straightforward, no matter the amount of actions being executed. The properties that hold during the execution of the action are treated exclusively using successor state axioms. There is no need to add extra actions to solve the problem of overlapping actions.

Within the extensions to this work, trigger and occurrence definitions can be augmented to cover a wider variety of cases of triggers and narratives. Finally, the axiomatization for actions with duration can be used for the implementation of a temporal planner in Prolog. This planner is currently under construction.

References

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

[2] Allen, J. F., and Hayes, P. J. Moments and Points in an Interval-Based Temporal Logic. *Computational Intelligence 5* (Dec. 1989), 225–238.

[3] 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.

[4] 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.

[5] 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.

[6] Pinto, J. Occurrences and Narratives as Constraints in the Branching Structure of the Situation Calculus. Submitted to the Journal of Logic and Computation
URL = ftp://lyrcc.ing.puc.cl/pub/jpinto/jlc.ps.gz, 1997.

[7] 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.

[8] 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.

[9] 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).