Non-instantaneous Actions and Concurrency in the Situation Calculus
Preliminary Draft

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]. 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].

There are several issues that one needs to address when introducing non-instantaneous actions. First, durations are measured in explicit time.

2 The Language

We use the Situation Calculus [5] 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 already introduced.

The foundational axioms for the concurrent Situation Calculus are:

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

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

$$(\forall s)\ \neg s < S_0, \tag{3}$$

$$(\forall c, s, s')\ s < do(c, s') \equiv Poss(c, s') \wedge s \leq s', \tag{4}$$

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

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

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

A concurrent action is a set of simple actions. We will consider that a concurrent action is possible if all the simple actions which constitutes 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 Occurrence predicates

We will make use of some occurrence predicates defined by Pinto [7].

A concurrent action c has occurred in a history s at time t if there exist 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:

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

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 it is conformed by the execution of a sequence of possible actions. A situation s is legal if there exist a situation in its future which is $proper$.

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

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

The $proper$ predicate *prunes* the branching structure of the Situation Calculus. For instance, if some event has occurred non preventably in some time t then all the situations in which the event has occurred at time t are proper.

We add the axiom $occurs_{to}(a, t, s)$ if we want an action a to preventably occur at some time t, i.e. occur at time t only if it is possible, all the histories having s in their past. If a situation s is $proper$ then all actions that occur at some time t must be in the history s if they were possible. The following axiom adds a relationship that all proper situations must satisfy:

$$proper(s) \supset [occurs_{to}(a, t, s') \wedge t < start(s) \wedge s' \prec s \supset hasOccurred(a, t, s_h)] \tag{13}$$

An action a_1 triggers a_2 in some time δ if, after δ units of time from execution time of a_1, a_2 must be executed.

$$trigger(a_1, a_2, \delta) \equiv [(\forall c, s)\, a_1 \in c \supset occurs_{to}(a_2, start(do(c, s)) + \delta, do(c, s))] \tag{14}$$

A preventable trigger can be defined analogously using the predicate $occurs_{ct} \subseteq \mathcal{A} \times \mathcal{T} \times \mathcal{S}$. $occurs_{ct}(a, t, s)$ holds when action a is triggered at time t in the future of s. The constraint for proper situations is:

$$\begin{aligned} proper(s) \supset [&occurs_{po}(a, t, s) \wedge t < start(s) \wedge s' \prec s \wedge \\ &Sit(t, s_t, s) \wedge Poss(a, s_t) \supset hasOccurred(a, t, s)] \end{aligned} \tag{15}$$

There also will be the possibility of incorporating natural actions (executed by nature due to physical phenomena). These actions can be either primitive or with duration and occur always when they are possible.

$$\begin{aligned} proper(s) \supset [&(\forall a, c, s')\, do(c, s') \preceq s \wedge natural(a) \wedge \\ &Poss(a, s) \supset a \in c] \end{aligned} \tag{16}$$

3.1 Extended triggers

In this section we add more trigger predicates that will be useful when dealing with actions with duration.

Some actions may preventably trigger other actions. For example the action $Drop$ will preventably trigger $HitGround$ in a time T after it is performed if there exist an action $Catch$. If catch is executed before T units of time, $HitGround$ will not necessarily occur. Moreover, we do not want $HitGround$ to occur before T units of time.

We introduce the predicate $trigger_{nb}$

4 Modeling actions with duration

Actions with duration can be modeled as a pair of instantaneous actions. One action initiates the execution of the action with duration and the other 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 executed along the path determined by s_1 and s_2, and that its duration is $start(s_2) - start(s_1)$.

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.

In general, an action with duration consists of three elements: an action name $ad(\bar{x})$, a formula φ^e_{ad}, simple in s, which is always true during the execution of the action, and φ^f_{ad}, a formula simple in s which becomes true once the execution of the action has finished.

4.1 Execution of an action

The action $begin(ad(\bar{x}))$ will begin the execution of the action $ad(\bar{x})$, and $finish(ad(\bar{x}))$ will finish it. $begin$ and end are instantaneous actions and thus they are part of the sort $\mathcal{A}$

$$begin(a_d) \in \mathcal{A} \tag{17}$$

$$finish(a_d) \in \mathcal{A} \tag{18}$$

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 is in progress during the situation s. The effect axioms for this fluent are:

$$Poss(c, s) \wedge begin(a_d) \in c \supset executing(a_d, do(c, s)). \tag{19}$$

$$Poss(c, s) \wedge finish(a_d) \in c \supset \neg executing(a_d, do(c, s)). \tag{20}$$

4.2 Preconditions of $begin$ and $finish$

For a particular action $A_d(\bar{x})$, both of these actions have preconditions of the form:

$$Poss(begin(A_d(\bar{x})), s) \supset \Pi^b_{A_d}(\bar{x}, s), \tag{21}$$

$$Poss(finish(A_d(\bar{x})), s) \supset \Pi^f_{A_d}(\bar{x}, s), \tag{22}$$

where $\Pi^b_{A_d}$ and $\Pi^f_{A_d}$ are disjunctions of fluents.

Depending on the problem one wants to model, the concurrent execution of $begin$ and $finish$ will or will not be possible. In the latter case, the actions with duration cannot be executed instantaneously.

We will assume the execution of $begin$ and $finish$ will not be concurrently possible for the same action. Additionally, 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:

$$Poss(begin(A_d(\bar{x})), s) \supset \neg executing(A_d(\bar{x}), s) \tag{23}$$

$$Poss(finish(A_d(\bar{x})), s) \supset executing(A_d(\bar{x}), s) \tag{24}$$

Using predicate completion, we derive:

$$Poss(begin(A_d(\bar{x})), s) \equiv \Pi^b_{A_d}(\bar{x}, s) \wedge \neg executing(A_d(\bar{x}), s) \tag{25}$$

$$Poss(finish(A_d(\bar{x})), s) \equiv \Pi^f_{A_d}(\bar{x}, s) \wedge executing(A_d(\bar{x}), s) \tag{26}$$

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* (23) *and* (24)*:*

□

4.3 Occurrence of $finish$

It is necessary to guarantee that any action that has started will end. Thus, the beginning of an action will *trigger* its end.

To mean that an action Ad will have duration δ we can use the $trigger$ predicate:

$$trigger(begin(Ad), finish(Ad), \delta)$$

4.4 Properties of the paths of action executions

As mentioned before, there are some properties that hold during and after the execution of an action with duration. These properties are necessary conditions for the action to be in execution.

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:

$$executing(A_d(\bar{x}), s) \supset \varphi^e_{A_d}(A_d(\bar{x}), s), \tag{27}$$

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 may deduce that A_d has finished.

Following [6], we can generate new effect axioms for all the fluents in $\varphi^e_{a_d}$ and for $executing$.

This is not sufficient to guarantee that the action $finish(a_d)$ will always occur as soon as executing becomes false. For this purpose, we add the following constraint for legal situations:

$$\begin{aligned} proper(s_h) \supset [do(c, s) \preceq s_h \wedge executing(a_d, s) \wedge \\ \neg executing(a_d, do(c, s)) \supset finish(a_d) \in c]. \end{aligned} \tag{28}$$

If $\varphi^f_{A_d}(\bar{x}, s) = \bigwedge_{i=1}^n l_i(\bar{x}, s)$, for each fluent literal l_i we write the following effect axiom:

$$Poss(c, s) \wedge finish(A_d(\bar{x})) \in c \supset l_i(\bar{x}, do(c, s)) \tag{29}$$

Al these effect axioms describe when an action changes its truth value from one situation to its successor but it does 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 *frame axioms* which solve this issue.

4.5 Some useful predicates

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 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,

$$\begin{aligned} during(a_d, a, s_h) \equiv & (\exists s, s', c, c').\, do(c, s) \prec do(c', s') \preceq s_h \wedge begin(a_d) \in c \wedge a \in c' \wedge \\ & \neg(\exists c'', s'')\, do(c, s) \prec do(c'', s'') \prec do(c', s') \wedge finish(a_d) \in c''. \end{aligned} \tag{30}$$

One may also want to say that in a history s an action with duration started at some point in time and ended at another. With this intention we introduce $hasOccurred \subseteq \mathcal{A}_d \times \mathcal{T} \times \mathcal{T} \times \mathcal{S}$, where $hasOccurred(a_d, t, t', s)$ is true when the action with duration a_d started at time t and ended at t' in the history s. The formal definition is:

$$\begin{aligned} hasOccurred(a_d, t, t', s) \equiv & [hasOccurred(begin(a_d), t, s) \wedge \\ & hasOccurred(finish(a_d), t', s) \wedge \\ & \neg(\exists t'')\, t < t'' < t' \wedge hasOccurred(finish(a_d), t'', s)] \end{aligned} \tag{31}$$

Example 1 In this example we have a door designed so that it requires both hands to open it. It has a spring lock that must be held open with one hand while the door is pulled with the other. The door is initially closed and locked.

There is one action with duration: $HoldLock$ and an instantaneous action $PullDoor$.

We will consider the action the beginning and finish of the action $HoldLock$ has no preconditions.

$$Poss(begin(HoldLock), s) \equiv \neg executing(HoldLock, s), \tag{32}$$

$$Poss(finish(HoldLock), s) \equiv executing(HoldLock, s). \tag{33}$$

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

$$Poss(PullDoor, s) \equiv unlocked(s). \tag{34}$$

For $PullDoor$ we have the following effect axiom:

$$Poss(c, s) \wedge PullDoor \in c \supset open(s). \tag{35}$$

φ^e and φ^f for $HoldLock$ are defined as follows:

$$\varphi^e_{HoldLock} =_{def} unlocked(s) \tag{36}$$

$$\varphi^f_{HoldLock} =_{def} \neg unlocked(s) \tag{37}$$

Following our procedure, we generate the following effect axioms:

$$Poss(c, s) \wedge begin(HoldLock) \in c \supset unlocked(do(c, s)) \tag{38}$$

$$Poss(c, s) \wedge finish(HoldLock) \in c \supset \neg unlocked(do(c, s)) \tag{39}$$

Using Reiter's method we derive the following successor state axioms:

$$\begin{aligned} Poss(c, s) \supset [unlocked(do(c, s)) \equiv & begin(HoldLock) \in c \vee \\ & unlocked(s) \wedge finish(HoldLock) \in c] \end{aligned} \tag{40}$$

$$Poss(c, s) \supset [open(do(c, s)) \equiv PullDoor \in c \vee open(s)] \tag{41}$$

The initial situation is characterized with:

$$\neg open(S_0) \wedge \neg unlocked(S_0). \tag{42}$$

Proposition 1 *Let Σ be a theory containing all the precondition, frame and initial description 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 you must have open the door while holding the lock.
PROOF: *We first demonstrate*

$$open(s) \supset (\exists s')do(c, s') \preceq s \wedge PullDoor \in c \wedge Poss(c, s') \tag{43}$$

By contradiction using (1) *on:*

$$\varphi(s) \equiv (\forall s).\ (\forall s')(do(c, s') \preceq s \wedge Poss(c, s') \supset PullDoor \notin c) \supset \neg open(s)$$

But, by (34)*:*

$$Poss(c, s') \wedge PullDoor \in c \supset unlocked(s') \tag{44}$$

Now, again by contradiction we prove:

$$unlocked(s') \supset (\exists s'')do(c, s'') \preceq s' \wedge begin(HoldLock) \in c \tag{45}$$

using (1) *on*

$$\varphi(s') \equiv (\forall s').\ (\forall s'')(do(c, s'') \preceq s' \wedge Poss(c, s'') \supset begin(HoldLock) \notin c) \supset \neg unlocked(s')$$

It follows from (43) *and* (45) *that:*

$$\begin{aligned} &legal(s) \wedge open(s) \supset \\ &\qquad (\exists s', s'', c', c'')do(c'', s'') \prec do(c', s') \preceq s \wedge begin(HoldLock) \in c'' \wedge PullDoor \in c. \end{aligned} \tag{46}$$

Without loss of generality we can assume that:

$$\begin{aligned} &(\forall s, s', s'', c', c')do(c'', s'') \prec do(c', s') \preceq s \wedge begin(HoldLock) \in c'' \wedge PullDoor \in c \wedge \\ &\qquad (\forall c''', s''')do(c'', s'') \prec do(c''', s''') \prec do(c', s') \end{aligned} \tag{47}$$

Then, using (30) *the theorem is proved.* □

Example 2 This is an extension to example 1. Here, to unlock the door it is necessary to turn the lock first. The door does not get unlocked as a result of turning the lock but as result of an exogenous action, due to a physical phenomenon, $unlock$. This action is triggered after some time of the beginning of $TurnLock$. There is also a natural action $lock$, that occurs when the door is unlocked and the lock not held.

There are two actions with duration: $TurnLock$ and $PullDoor$, two instantaneous actions $LetGo$ and $unlock$, and one instantaneous natural action $lock$. Additionally, there is an action $unlock$, that is not executable by the agent, but triggered by the beginning of $TurnLock$.

We will suppose that it is always possible to begin and finish the execution of $TurnLock$. It is possible to open the door if it is $unlocked$. It is possible to let the lock go if it is held. Applying predicate completion, we obtain:

$$Poss(begin(TurnLock), s) \equiv \neg executing(TurnLock, s) \tag{48}$$
$$Poss(finish(TurnLock), s) \equiv executing(TurnLock, s) \tag{49}$$
$$Poss(begin(PullDoor), s) \equiv unlocked(s) \wedge \neg executing(PullDoor, s) \tag{50}$$
$$Poss(finish(PullDoor), s) \equiv executing(PullDoor, s) \tag{51}$$
$$Poss(LetGo, s) \equiv held(s). \tag{52}$$

$lock$ is a natural action

$$natural(lock). \tag{53}$$

The precondition axioms for $lock$ and $unlock$ are:

$$Poss(lock, s) \equiv \neg held(s), \tag{54}$$
$$Poss(unlock, s) \equiv True. \tag{55}$$

And their effect axioms:

$$Poss(c, s) \wedge lock \in c \supset \neg unlocked(do(c, s)) \tag{56}$$
$$Poss(c, s) \wedge unlock \in c \supset unlocked(do(c, s)) \tag{57}$$

Both $lock$ and $unlock$ are not supposed to be executed by the agent but by nature as a consequence of an underlying physic phenomenon.

For the action $LetGo$ we have the following effect axiom:

$$Poss(c, s) \wedge LetGo \in c \supset \neg held(do(c, s)). \tag{58}$$

In this version of the example the action $begin(PullDoor)$ will have no direct effect, except that the action $PullDoor$ is being executed. When the action $PullDoor$ is ended, the door is $open$. We define now φ^e and φ^f for the actions with duration:

$$\varphi^e_{TurnLock}(s) =_{def} held(s). \tag{59}$$
$$\varphi^f_{PullDoor}(s) =_{def} open(s). \tag{60}$$

Using (27) we add the ramification constraint:

$$executing(TurnLock, s) \supset held(s), \tag{61}$$

from where we derive the following effect axioms:

$$Poss(c, s) \wedge begin(TurnLock) \in c \supset held(do(c, s)), \tag{62}$$
$$Poss(c, s) \wedge LetGo \in c \supset \neg executing(TurnLock, do(c, s)). \tag{63}$$

We add effect axioms for $finish(PullDoor)$ as indicated by (29):

$$Poss(c, s) \wedge finish(PullDoor) \in c \supset open(s). \tag{64}$$

Using Reiter's method, we can automatically generate successor state axioms for $open$, $held$, and $unlocked$.

It is necessary to say that $PullDoor$ must finish after it has begun. Let us suppose its duration is δ. Then, we add:

$$trigger(begin(PullDoor), finish(PullDoor), \delta) \tag{65}$$

The natural action $unlock$ is *triggered* by $begin(TurnLock)$ after some time. Let us suppose it takes ρ units of time turning the lock to unlock it. To indicate this we add

$$trigger(begin(TurnLock), unlock, \rho). \tag{66}$$

You cannot turn the lock for ever, sooner or later you will not be able to continue. Suppose $TurnLock$ cannot last more than ρ' units of time. Then,

$$trigger(begin(TurnLock), end(TurnLock), \rho'). \tag{67}$$

The initial situation S_0 is characterized with:

$$\neg unlocked(S_0) \wedge \neg open(S_0). \tag{68}$$

Proposition 2 *Let Σ be a theory containing all the precondition, frame and initial description axioms. Then,*

$$\begin{aligned}\Sigma \models (\forall s).legal(s) \wedge open(s) \supset (\exists t_1, t_1', t_2, t_2')\, & hasOccurred(TurnLock, t_1, t_1', s) \wedge \\ & hasOccurred(PullDoor, t_2, t_2', s) \wedge t_2 - t_1 > \rho\end{aligned}$$

I.e., if there exists a situation where the door is opened, then in some moment in the past of s you must have held the lock and open the door at least ρ units of time later.

5 Why the use of actions with duration may help

When dealing with concurrency of actions, two difficult problems immediately arise: the *precondition interaction* problem and *effect interaction problem* [6]. The first one arises when the possibility of execution of an action depend on whether other actions are performed concurrently. The second has to do with the possible interactions between the effect of the primitive actions when performed concurrently. In this section we will focus on the first problem.

The precondition interaction problem is normally due to incompatibility of the use of certain physical resources. Nevertheless, there are other situations in which this problem appears because of inappropriately modeling actions that in the real world have duration as instantaneous actions. That is the case of the following example.

Example 3 In this example we have a box for eggs. The box has four holes for putting the eggs and two eggs. You can move eggs from one hole to another when the destination hole has no egg over it. The axiomatization for the problem using instantaneous actions follows:

$$egg(E1) \wedge egg(E2) \wedge hole(H1) \wedge hole(H2) \wedge hole(H3) \wedge hole(H4) \tag{69}$$

The fluent for describing the position of the eggs is $over(e, h, s)$, which holds when e is over h.

The only possible actions is $move(e, h_1, h_2)$ which moves the egg e from hole h_1 to hole h_2 and that is possible when there is no other egg over h_2.

$$Poss(move(e, h_1, h_2), s) \supset egg(e) \wedge hole(h_1) \wedge hole(h_2) \wedge \neg(\exists e')\, over(e', h_2) \tag{70}$$

The successor state axiom for $over$ is:

$$Poss(c,s) \supset [over(e,h,do(c,s)) \equiv (\exists h')hole(h') \wedge move(e,h',h) \in c) \vee \\ over(e,h,do(c,s)) \wedge \neg(\exists h'')move(e,h,h'') \in c \wedge h \neq h''] \tag{71}$$

If the initial situation is characterized with:

$$over(E1, H1, S_0) \wedge over(E2, H2, S_0). \tag{72}$$

Suppose we wanted a sequence of actions to have $E1$ over $H2$ and $E2$ over $H1$. The most simple and cheapest plan should be:

$$do(\{move(E1, H1, H2), move(E2, H2, H1)\}, S_0) \tag{73}$$

However the concurrent action $\{move(E1, H1, H2), move(E2, H2, H1)\}$ is not possible because of (70) and (7).

The problem of this example is not the axiom (7) but the improper specification of $move$ as an instantaneous. In the real world this is action is not instantaneous.

The problem is solved if we view $move$ as an action with duration with these characteristics:

Example 4 (cont'd) The preconditions for the beginning and finish of $move$ are:

$$Poss(begin(move(e, h_1, h_2)), s) \supset egg(e) \wedge hole(h_1) \wedge hole(h_2) \tag{74}$$

$$Poss(finish(move(e, h_1, h_2)), s) \supset \neg(\exists e')\ over(e', h_2) \tag{75}$$

The effect axioms are:

$$Poss(c, s) \wedge begin(move(e, h_1, h_2)) \in c \supset \neg over(e, h_1, do(c, s)) \tag{76}$$

$$Poss(c, s) \wedge finish(move(e, h_1, h_2)) \in c \supset over(e, h_2, do(c, s)). \tag{77}$$

Proposition 3 *With this axioms the situation:*

$$S_1 = do(\{finish(move(E1, H1, H2)), finish(move(E2, H2, H1))\}, \\ do(\{begin(move(E1, H1, H2)), begin(move(E2, H2, H1))\}, S_0)) \tag{78}$$

is such that $legal_{poss}(S_1) \wedge over(E1, H1, S_1) \wedge over(E2, H2, S_1)$
PROOF: *Follows directly from the axiomatization.*

6 Narratives involving non instantaneous actions

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

6.1 Non preventable occurrences

One may want to express the fact that an action starts at some particular time and ends at some other time. For example, consider the narrative:

> My watch beeps everyday form 7:30am to 7:31am.

To this extent we incorporate the predicate $occurs_{bet} \subseteq \mathcal{A}_d \times \mathcal{T} \times \mathcal{T}$. $occurs_{bet}(a_d, t, t')$ holds when the action a_d started at time t and ended at time t'. If $occurs_{bet}(a_d, t, t')$ is true, then in any legal situation, $begin(a_d)$ must have occurred at time t and $finish(a_d)$ at time t'. Thus,

$$proper(s) \supset [occurs_{bet}(a_d, t, t') \wedge t < t' < start(s) \supset \\ hasOccurred(a_d, t, t', s)]. \tag{79}$$

In the example, the narrative can be modeled with the formula:

$$occurs_{bet}(Beep, \text{7:30am}, \text{7:31am}). \tag{80}$$

6.2 The $when$ predicate

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 these kinds 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. If one wants to assert that the action $Ring$ occurred at time T while $DrinkCoffee$ was in progress, then the following axiom should be added into the theory:

$$when(DrinkCoffee, Ring, T)$$

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

$$\begin{aligned} proper(s) \supset [when(a_d, a, t) \wedge t < start(s) \supset & \\ & hasOccurred(a, t, s) \wedge \\ & (\exists t').\, hasOccurred(begin(a_d), t', s) \wedge t' < t \wedge \\ & \neg(\exists t'')hasOccurred(finish(a_d), t'', s) \wedge t' < t'' < t] \end{aligned} \tag{81}$$

In the example the narrative can be defined by:

$$occurs_{np}(DrinkCoffee, 7:00am) \wedge when(DrinkCoffee, Ring, 7:05am)$$

For actions with duration that occur during actions with duration we can extend the while predicate such that $while \subseteq \mathcal{A}_d \times \mathcal{A}_d \times \mathcal{T}$. The constraint axiom for legal situations is:

$$when(a_d, a'_d, t) \equiv when(a_d, begin(a'_d), t) \tag{82}$$

Theorem 2 *For any legal situation, if $when(a_d, a, t)$ holds, then a is executed in a situation in which a_d is in progress.*

$$legal(s) \wedge when(a_d, a, t) \wedge t = start(s') \wedge s' \prec s \supset executing(a_d, s')$$

7 Temporal planning

In this section we show how can temporal planning be implemented in PROLOG.

7.1 Planning in Prolog

Following Green [4], planning in a theorem prover can be addressed by asking if there exist a situation s satisfy a goal description G.

The axioms of a theory in the traditional atemporal Situation Calculus can be written in PROLOG, as it will be shown later. A predicate `plan(S)` can be implemented in PROLOG such that is satisfies when `S` is the desired goal. This predicate can be syntactically written in a way to behave as a *breath-first*, *depth-first* or an *A* search.

Obtaining an atemporal plan is, then, somewhat simple, you just need to write the axiomatization, write a predicate `plan(S)` to achieve your goal, and ask prolog to satisfy that predicate.

7.2 Problems with temporal planning in the Situation Calculus

When you want to generate temporal plans in the presence of triggers, occurrences, etc. it is not that simple; several problems arise:

- When to execute an action? If a planner is at some point in time and decides to perform an action, when should this action be performed? This is a difficult question, the agent has, indeed, infinite possibilities to choose. If you look carefully to our axiomatization, there is not a way of deducing when to execute an action. Actually, there is no way of determining what value should the $start$ of a situation be equal to.
- Infinite plans may achieve the goal. This is because the problem described above. As the agent has many possibilities for choosing when to execute a certain action, the plans that can achieve the same goal can be infinite. One solution to this is to choose a granularity of time. This is not a bad idea if we think the agent may have limitations. For example, it will not be able to execute a large number actions in a very short time.
- What granularity to choose? The answer to this question will probably come from the problem being modeled, though it might be very difficult to determine (or even impossible?).

7.3 A naive solution

One solution to this problem is to plan forgetting that time exists, obtain an atemporal plan and then *project* this plan to a timeline and verify if the mapping situation is proper, i.e. a situation is such that all the things that had to happen occurred.

A projection of a situation s is a complete assignation of the $start$s for each situation in the history of s, including itself. For example,

```
project(do(A_1,do(A_2,S_0))) = \{ (do(A_1,do(A_2,S_0)),4), (do(A_2,S_0),3),
(S_0,0)\}
```

if and only if:

$$\begin{aligned} start(do(A_1, do(A_2, S_0))) &= 4 \\ start(do(A_2, S_0)) &= 3 \end{aligned}$$

7.3.1 Implementation

Aqui va lo que hay en el archivo scoc.pl.

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] 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.

[5] 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.

[6] 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.

[7] 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.

[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.