Non-determinism and Uncertainty in the Situation Calculus

JAVIER PINTO
Departamento de Ciencia de la Computación
Pontificia Universidad Católica de Chile
Santiago 6904411, Chile
jpinto@ing.puc.cl

AMíLCAR SERNADAS, CRISTINA SERNADAS, and PAULO MATEUS
CMA, Departamento de Matemática
Instituto Superior Técnico
1096 Lisboa, Portugal
{acs,css,pmat}@math.ist.utl.pt

The purpose of this article is to extend the situation calculus, a logical framework for the specification of theories of action and change, with actions that have a non-deterministic or uncertain nature. Our approach is based upon the idea that actions may have a *deterministic* component, and a *probabilistic* component. For example, the act of flipping a coin has a deterministic component (the actual coin toss) and an uncertain component (the outcome). We extend the language of the situation calculus in order to make explicit this distinction between this two action components. Furthermore, we provide means to reason about the outcomes of processes specified only in terms of deterministic action components (which we call behaviors). In particular, we show how one can compute the probabilty that some fluent will hold after specific behavior is realized. An important feature of our approach is that the syntactic and semantic structure of actions and situations is independent of the decomposition of actions into deterministic and uncertain components. Thus, we inherit solutions to the frame problem, ramification problem, etc.

Keywords: Theories of Action; Situation Calculus; Uncertainty Handling.

1. Introduction and Motivation

When agents act in the world they encounter situations where the actions that they perform seem to have uncertain effects. For example, throwing a dart at a dart-board might result in the dart being in several possible places, depending on the accuracy and intentions of the thrower. For example, picture a novice player of darts in an Irish pub after drinking a couple of pints of Guinness. It is quite likely that this dart player hits any point on the dartboard, or on the wall, without much control on his part. On the other hand, picture a professional darts player with the ability to direct her throw with the utmost accuracy. The novice darts player is performing an indeterminate action, whose results cannot be accurately predicted; whereas, the pro performs a nearly determinate action. In the light of this example, we might ask "what is the source of indeterminacy?" The point of the example is to suggest that primitive actions are not indeterminate. One can easily imagine a darts player so good that he can always throw the dart wherever she wants. Thus, a dart throw is not in and of itself an indeterminate action. What happens is that the agent might not have the ability to execute the action that she or he wants to perform.

In a previous article[1], a proposal to view indeterminacy as arising from an indeterminate choice between primitive completely determinate actions was advanced. I.e., actions do not have indeterminate effects. Indeterminate actions were modeled as sets of primitive, determinate actions. There is one crucial advantage of this view of indeterminacy: it is not necessary to deal with the specification of the effects of indeterminate primitive actions. The effects of indeterminate actions are derived from the effects of the individual primitive actions. Therefore, the frame problem, the ramification problem, the qualification problem, etc. are orthogonal to the issue of indeterminacy in theories of actions in this style. Therefore, solutions to these problems in completely determinate worlds would carry over to worlds in which agents might perform indeterminate actions (viewed as choice among a set of primitive, determinate actions).

In this article, we depart from the proposal mentioned before, and follow another route. Instead of appealing to sets in order to model indetermination, we model indeterminate actions by decomposing them into a *non-deterministic* part and its possible *outcomes*. For example, flipping a coin is modeled as the non-deterministic flipping along with the possible outcomes *heads* and *tails*. Therefore, one action is *flip-heads*, and the other is *flip-tails*. Also, we associate a probability distribution to the *outcomes*, given the *non-deterministic* action part. Notice, however, that the action *flip-heads* remains deterministic. Its effects are described with standard effect axioms. Therefore[1], the solutions to the *frame* and related problems still apply.

Another example, discussed in the literature, has to do with the motion control of a wheeled robot. Imagine a robot that moves on a Cartesian space. The control action of the robot might be to advance forward by 20 cms.; but, due to slippage, the robot might end up moving only 17 cms. The robot's control operation is not indeterminate, it is perfectly precise. The outcome obtained from the operation, however, has an indeterministic component, which arises from the interaction of the robot's control mechanisms and its environment.

Our view of indeterminacy is independent of the formalism in which one chooses to write theories of action and change. Thus, we believe, one should be able to formalize indeterminacy with this view by extending any of the logical languages that have been proposed to write theories of (determinate) action and change. However, our technical proposal has been developed within the framework of the Situation Calculus [2,3].

Lately, a number of proposals have been advanced in order to deal with these issues. For instance, approaches based on fuzzy logics[4,5], and closer approaches also based upon the situation calculus and probabilities[6,7].

The article is organized as follows. First we present the logical framework that constitutes the basis for our technical proposal. Then, we extend the language presented in order to accommodate indeterminate actions (as pairs), and to give probabilities to the outcomes. Afterwards, we deal with ordered sequences of indeterminate actions. Later, we model the *alternating bit protocol*, which is used in order to ensure that a set of messages are correctly communicated (in order) from a sender to a receiver. The non-determinism that arises is due to losses in the communication channels. Later, we compare the main features of our approach with other approaches to deal with similar issues, also based on the situation calculus. Finally we present our conclusions and discuss our future work on this subject.

2. Standard Situation Calculus

The situation calculus originated from the pioneering work of John McCarthy[8,2] who proposed the logical language as a means to represent knowledge about actions and the change they provoke on the world. In the last decade, the situation calculus has received much attention in the theories of action community. In particular, due to Raymond Reiter's work on the frame problem[3]. Our work is based on Reiter's solution to the frame problem, along with extensions to deal with the ramification problem[9,10]. Although the situation calculus that we use owes much to the original language devised by McCarthy, there are some conceptual differences. In particular, the notion of *situation* differs in some ways. Our view of situations is that they are what results from the execution of sequences of actions in some initial situation. On the other hand, McCarthy viewed situations as *snapshots* of the world. Thus, McCarthy's view of situations seems closer to the notion of *state*. In this section, we describe the dialect of the situation calculus used in this paper.

Our dialect of the situation calculus is a many sorted second order language. The sorts are $\mathcal{A}$ for primitive actions, $\mathcal{S}$ for situations, and $\mathcal{F}$ for fluents. Notice that fluents are reified, in contrast with the language used by Reiter[3], in which fluents are predicates that take exactly one argument of type situation. We include a sort $\mathcal{D}$ for domain objects. Also, we have the following distinguished symbols: The constant S_0 for the initial situation; the function $do : \mathcal{A} \times \mathcal{S} \rightarrow \mathcal{S}$; the predicate $holds \subseteq \mathcal{F} \times \mathcal{S}$; the predicate $Poss \subseteq \mathcal{A} \times \mathcal{S}$. Furthermore, we make use of $\mathcal{R}_0^+$, the positive reals, as an interpreted sort.

The following are foundational axioms for the situation calculus:

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

$$(\forall\, a_1, a_2, s_1, s_2).\ do(a_1, s_1) = do(a_2, s_2) \supset a_1 = a_2, \tag{2}$$

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

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

Axiom (1) is an induction axiom, inspired by Peano's induction axiom for the natural numbers. φ is taken to be a one place predicate variable for objects of type $\mathcal{S}$. In a first order axiomatization, we take the axiom to be an induction schema and work with the *standard models* of the theory.

3. Introducing Probabilistic Actions

The language of the situation calculus is extended by defining new sorts $\mathcal{B}$, $\mathcal{T}$ and $\mathcal{E}$, corresponding to *non-deterministic actions*, *transitions*, and *probabilistic outcomes* respectively. The non-deterministic actions correspond to the choices that can be made by an agent. For instance, a non-deterministic action could be a *coin-toss*. A transition corresponds to a probabilistic choice point, which is reached after choosing a non-deterministic action in a given situation. An outcome corresponds to one probabilistic alternative in a transition. For instance, the outcomes could be *heads* or *tails*, which might correspond to the alternatives available in a transition resulting from a coin-toss. A situation, in our extended situation calculus, is a choice point for the agent, where she chooses (non-deterministically) an action to perform. A transition, on the other hand, corresponds to a choice point for *nature*, thus, given a situation and an agent action, *nature* chooses an event (probabilistically) that leads to the successor situation.

Our way of modeling probabilistic actions is related to the separation between probabilistic and non-deterministic choice made in the literature on specification and verification of randomized distributed systems[11]. To make a parallel with the extended situation calculus presented here, non-determinism corresponds to the choice of behavior made by the agent (the *non-deterministic action*), whereas the probabilistic choice corresponds to the event that results of a choice made by *nature*. On the other hand, the modeling of probabilistic actions, and its integration with the situation structure of the situation calculus, induces a tree which can be viewed as a probabilistic decision tree. The situations in the extended situation calculus correspond to what Shafer[12] calls *decision situation* and transitions are *chance situations*.

We will assume that the sort $\mathcal{E}$ of outcomes is finite. We restrict the language to include a finite set of constants of type $\mathcal{E}$; and we assume domain closure and uniqueness of names for the outcomes. This could be extended to countable domains of outcomes, whenever there are free generators for $\mathcal{E}$. This limitation is shared by other approaches to handling uncertainty within the situation calculus described in the literature[6,7]. In current research [13], we are considering a different approach, in which transitions may have a continuous set of possible outcomes.

We introduce the new function symbols $do_{[} : \mathcal{B} \times \mathcal{S} \rightarrow \mathcal{T}$, and $do_{]} : \mathcal{E} \times \mathcal{T} \rightarrow \mathcal{S}$ and the new predicate symbols $Poss_{[} \subseteq \mathcal{B} \times \mathcal{S}$, and $Poss_{]} \subseteq \mathcal{E} \times \mathcal{T}$.

Given a non-deterministic action b and an outcome e, we represent the action that results from the non-deterministic action and outcome using the function $\langle\rangle$: $\mathcal{B} \times \mathcal{E} \rightarrow \mathcal{A}$. We use the notation $\langle b, e\rangle$ as syntactic sugar for $\langle\rangle(b, e)$. If one wants all the actions to have a probabilistic component, we can use the axiom:

$$(\forall\, a)\ (\exists\, b, e)\ a = \langle b, e\rangle.$$

Thus, obtaining $\mathcal{A} = \mathcal{B} \times \mathcal{E}$. Also, in order to preserve the tree structure (with the now embedded transitions), we need the following uniqueness of names axiom*:

$$\langle b, e\rangle = \langle b', e'\rangle \supset b = b' \wedge e = e'. \tag{5}$$

We also need the structural axioms:

$$do(\langle b, e\rangle, s) = do_{]}(e, do_{[}(b, s)), \tag{6}$$

$$(\forall\, t)\ (\exists\, b, s)\ t = do_{[}(b, s). \tag{7}$$

These axioms force the tree structure to be well formed. In particular, it disallows the existence of transitions that don't arise from executing behaviors in existing situations.

Remark 1 From the foundational axioms (1)-(3), and axioms (5)-(7), it follows that:

$$do_{[}(b, s) = do_{[}(b', s') \supset b = b' \wedge s = s',$$

and

$$do_{]}(e, t) = do_{]}(e', t') \supset e = e' \wedge t = t',$$

For $Poss$ we have:

$$Poss(\langle b, e\rangle, s) \equiv Poss_{[}(b, s) \wedge Poss_{]}(e, do_{[}(b, s)). \tag{8}$$

For each transition in $\mathcal{T}$, we introduce a probability presentation for the set of outcomes (which we assume to be finite) as a function $Pr : \mathcal{E} \times \mathcal{T} \rightarrow \mathcal{R}_0^+$, with the following constraints:

$$Poss_{[}(b, s) \supset \sum_{e \in \mathcal{E}} Pr(e, do_{[}(b, s)) = 1.0, \tag{9}$$

$$\neg Poss_{]}(e, t) \supset Pr(e, t) = 0.0. \tag{10}$$

Notice that (9) is not a proper formula of the language. This formula stands for a finite sum over all the $\mathcal{E}$ terms of the language. However, we need to assume that we have a finite number of $\mathcal{E}$ terms plus a domain closure assumption.

*In the rest of the article, all free variables that appear in formulas are assumed to be universally quantified with maximum scope.

Remark 2 From (9) and (10), and the probability axioms, it follows that:

$$Poss_{[}(b,s) \supset (\exists e)\ Poss(\langle b,e\rangle,s).$$

Thus, if some behavior is possible in a situation, then there must be a possible outcome in the transition that results from the execution of the behavior in the resulting transition.

3.1. *Example: A coin toss*

Consider the simple example of the coin toss. There is only one possible non-deterministic action, the toss. We have a single constant *Toss* of sort $\mathcal{B}$. The outcomes can be heads or tails, modeled with the constants *Heads* and *Tails* of sort $\mathcal{E}$.

Precondition axioms

$$\begin{aligned}&Poss(\langle Toss, Heads\rangle, s),\\&Poss(\langle Toss, Tails\rangle, s).\end{aligned}$$

Effect axioms

$$\begin{aligned}&Poss(\langle Toss, Heads\rangle, s) \supset\\&\qquad holds(Head\text{-}Up, do(\langle Toss, Heads\rangle, s)),\\&Poss(\langle Toss, Tails\rangle, s) \supset\\&\qquad holds(Tail\text{-}Up, do(\langle Toss, Tails\rangle, s)).\end{aligned}$$

State constraints

$$holds(Head\text{-}Up, s) \equiv \neg holds(Tail\text{-}Up, s).$$

Possible outcomes axioms

$$Poss_{]}(e, do_{[}(Toss, s)) \supset e = Heads \vee e = Tails.$$

Probability assignments

$$Pr(Heads, do_{[}(Toss, s)) = Pr(Tails, do_{[}(Toss, s)).$$

4. Behavioral Sequences

Each node in the situation/transition structure, identifies a unique finite path leading from the root (S_0) to the node. Each path in this tree should be interpreted as a possible outcome obtained by following a specific *behavioral sequence*. For example, the path identified by the situation:

$$do(\langle Toss, Heads\rangle, do(\langle Toss, Tails\rangle, S_0)),$$

corresponds to one possible outcome of the behavioral sequence *Toss*, *Toss*. We find it convenient to introduce explicitly the notion of behavioral sequence. We introduce the sort $\mathcal{B}^*$ for behavioral sequences. We use the constant term [] to denote an empty sequence. Also, we use $\vec{b}$ as a variable of sort $\mathcal{B}^*$. We use the operator $| : \mathcal{B} \times \mathcal{B}^* \rightarrow \mathcal{B}^*$ to build a new sequence out of an element and a sequence (as the Lisp *cons* function). We write $[b|\vec{b}]$ as syntactic sugar for $|(b, \vec{b})$. We use $first$ and $tail$. We take $tail$ of empty sequences to be the empty sequence as well. The first of an empty sequence is taken to be some illegal value, which we also denote with [], along with the requirement that [] is never possible:

$$\neg Poss_{[}([], s). \tag{11}$$

Therefore, if $\vec{b}$ is a non-empty sequence, $first(\vec{b})$ denotes the first element of the sequence, and $tail(\vec{b})$ denotes the same sequence minus the first element. The axioms to characterize sequences are straightforward and are omitted.

A behavioral sequence is a finite ordered sequence of non-deterministic actions $b_1, \ldots, b_n$. A behavioral sequence induces a finite tree which is a subtree of the situation calculus. An important reasoning task that we have in mind when modeling uncertain actions is *plan verification*. A behavioral sequence can be seen as a plan of action, and we would like to be able to determine whether the plan yields the expected results. Given that we work with probabilistic outcomes, we should be able to answer the following questions:

- *What is the probability that a behavioral sequence can be performed successfully in a given situation?* By successfully, we mean that the behavioral sequence can be completed in a given situation, as defined formally later.
- *What is the probability that some fluent holds after a behavioral sequence is completed successfully in some situation?*

We define a reachability relation $\rightarrow : \mathcal{S} \times \mathcal{B}^* \times \mathcal{S}$, such that $\rightarrow (s, \vec{b}, s')$ is true if s' can be reached, probabilistically, from s by performing the behavioral sequence $\vec{b}$. As syntactic sugar, we write $s \xrightarrow{\vec{b}} s'$. One possible definition for the reachability relation is as follows:

$$\begin{aligned} s^0 \xrightarrow{b_1, \ldots, b_n} s^n \equiv (\exists\, e_1, \ldots, e_n) \\ s^n = do([\langle b_1, e_1 \rangle, \ldots, \langle b_n, e_n \rangle], s^0) \wedge s^0 \leq s^n. \end{aligned}$$

Thus, if $s \xrightarrow{\vec{b}} s'$ holds, then we have a sequence of actions built from the behavioral sequence that leads from s to s'. The literal $s^0 \leq s^n$, forces the reachability relation to consider only sequences that can be realized by performing possible actions. The previous definition for the relation $\rightarrow$ is realized in the recursive definition (12) and (13) below.

$$s \xrightarrow{\vec{b}} s \equiv \vec{b} = []. \tag{12}$$

This part of the definition states that there is a transition from a situation s onto itself if the behavioral sequence is empty. The rest of the definition for $\rightarrow$ is:

$$\begin{aligned} \vec{b} \neq [] \supset [s \xrightarrow{\vec{b}} s' \equiv s < s' \wedge (\exists s'', e) \\ s'' = do(\langle first(\vec{b}), e\rangle, s) \wedge s'' \xrightarrow{tail(\vec{b})} s']. \end{aligned} \tag{13}$$

Given a behavioral sequence, the reachability relation between situations characterizes the notion that a *behavioral sequence can be performed successfully.* Thus, we can define this notion with the abbreviation:

$$successful(\vec{b}, s) =_{def} (\exists s')\ s \xrightarrow{\vec{b}} s'.$$

Starting from the transition probabilities introduced in the previous section, we can now define the following probabilities:

- $Prx : \mathcal{S} \times \mathcal{B}^* \times \mathcal{S} \rightarrow \mathcal{R}_0^+$. We write $Prx(s, \vec{b}, s')$ to denote the probability of reaching s' from s, by performing $\vec{b}$. These probabilities, Prx, are characterized with the following axioms:

$$\neg s \xrightarrow{\vec{b}} s' \supset Prx(s, \vec{b}, s') = 0.0, \tag{14}$$

$$Prx(s, [], s) = 1.0, \tag{15}$$

$$\begin{aligned} & do(\langle b, e\rangle, s) \leq s' \wedge s \xrightarrow{[b|\vec{b}]} s' \supset \\ & \quad Prx(s, [b|\vec{b}], s') = Pr(e, do_{[}(b, s)) \times Prx(do(\langle b, e\rangle, s), \vec{b}, s'). \end{aligned} \tag{16}$$

- $Prb : \mathcal{B}^* \times \mathcal{S} \rightarrow \mathcal{R}_0^+$. We write $Prb(\vec{b}, s)$ to denote the probability that it is possible that the non-deterministic action $\vec{b}$ be performed successfully in s.

$$Prb(\vec{b}, s) = \sum_{s' \in S} Prx(s, \vec{b}, s'). \tag{17}$$

  Notice that, as with (9), this is not a proper formula of our logical language. As before, we take this formula to be a shorthand for a finite sum over $\mathcal{R}_0^+$ terms. This formula can only be expressed when $\vec{b}$ is a fixed and finite sequence of $\mathcal{B}$ terms. The same restrictions apply to (18) and (21) below. Alternatively, this summation can be expressed by using a recursive specification.

- $Prf : \mathcal{F} \times \mathcal{B}^* \times \mathcal{S} \rightarrow \mathcal{R}_0^+$. We write $Prf(f, \vec{b}, s)$ to denote the probability that fluent f is true given that the sequence $\vec{b}$ is attempted in s.

$$Prf(f, \vec{b}, s) = \sum_{s' \in \mathcal{S}} Prx(s, \vec{b}, s') \times \overline{holds}(f, s'), \tag{18}$$

  where we have used the function symbol $\overline{holds} : \mathcal{F} \times \mathcal{S} \rightarrow \mathcal{R}_0^+$, which is simply defined to be equal to 1.0 for a fluent f and situation s, iff the fluent f holds in s, and is defined to be 0.0 otherwise. Thus, we write:

$$\overline{holds}(f, s) = 1 \equiv holds(f, s), \tag{19}$$
$$\overline{holds}(f, s) = 0 \equiv \neg holds(f, s). \tag{20}$$

From (18), it follows that probability $Prf(f, \vec{b}, s)$ is 0 if $\vec{b}$ may not be performed in s. Notice that (18) corresponds to the utility obtained from executing the behavior sequence $\vec{b}$, assuming that the utility is 1 if f holds and 0 otherwise.

Also, we define $Prnf : \mathcal{F} \times \mathcal{B}^* \times \mathcal{S} \rightarrow \mathcal{R}_0^+$. We write $Prnf(f, \vec{b}, s)$ to denote the probability that fluent f is false given that the sequence $\vec{b}$ is attempted in s.

$$Prnf(f, \vec{b}, s) = \sum_{s' \in \mathcal{S}} Prx(s, \vec{b}, s') \times (1 - \overline{holds}(f, s')). \tag{21}$$

We have:

Remark 3

$$Prf(f, \vec{b}, s) + Prnf(f, \vec{b}, s) = Prb(\vec{b}, s).$$

- The probability that a fluent f is true given that the sequence $\vec{b}$ was successfully performed in s, denoted $CPrf(f, \vec{b}, s)$, is given by:

$$CPrf(f, \vec{b}, s) = \frac{Prf(f, \vec{b}, s)}{Prb(\vec{b}, s)}. \tag{22}$$

4.1. *Example: Russian Roulette*

Consider a gambler in a game of Russian roulette. In this *game*, a player takes a revolver with 6 bullet chambers in the barrel and loads one of the chambers with a bullet, leaving the other 5 empty. The player rotates the barrel and pulls the trigger while pointing at himself. The probability that the player gets shot is assumed to be 1/6. To simplify the example, we consider the player's move to be a single non-deterministic action *Rshoot* for rotate and shoot (of sort $\mathcal{B}$), with two possible outcomes *Die* and *Survive* of sort $\mathcal{E}$. We want to determine what is the probability of being alive after playing at the roulette twice.

We need to write the theory of action that describes this domain. The constant symbols *Rshoot*, *Die* and *Survive* were specified above. We also have the single fluent constant *Alive*. Again, for simplicity, we assume that the revolver is always properly loaded, and don't characterize its behavior.

Action precondition axioms

$$Poss(\langle Rshoot, Die\rangle, s) \equiv holds(Alive, s),$$
$$Poss(\langle Rshoot, Survive\rangle, s) \equiv holds(Alive, s).$$

Effect and successor state axioms Reiter's solution to the frame problem specifies a mechanism to derive a set of successor state axioms from a set of effect axioms. There is only one effect axiom in this example:

$$Poss(\langle Rshoot, Die\rangle, s) \supset \neg holds(Alive, do(\langle Rshoot, Die\rangle, s)).$$

From this effect axiom, we obtain the following successor state axiom:

$$Poss(a, s) \supset [holds(Alive, do(a, s)) \equiv holds(Alive, s) \wedge a \neq \langle Rshoot, Die\rangle]. \quad (23)$$

Initial situation The previous specification characterizes the behavior of a completely determinate system. We assume that in the initial situation:

$$holds(Alive, S_0).$$

Possible outcomes axiom

$$Poss_{]}(e, do_{[}(Rshoot, s)) \supset e = Die \vee e = Survive.$$

Probability assignments

$$Pr(Die, do_{[}(Rshoot, s)) = 1/6,$$
$$Pr(Survive, do_{[}(Rshoot, s)) = 5/6.$$

Now, we would like to determine what is the probability that the gambler will survive after two rounds of shootings:

$$Prf(Alive, [Rshoot, Rshoot], S_0).$$

Obviously, the answer should be $5/6 \times 5/6$. Let us see if our axiomatization sanctions this result.

Using the foundational axioms, the action precondition axioms, and (12) and (13) it is easy to show that:

$$S_0 \overset{[Rshoot]}{\longrightarrow} s \equiv s = do(\langle Rshoot, Die\rangle, S_0) \vee s = do(\langle Rshoot, Survive\rangle, S_0).$$

Also,

$$\neg(do(\langle Rshoot, Die\rangle, S_0) \overset{[Rshoot]}{\longrightarrow} s),$$

and

$$
\begin{aligned}
&do(\langle Rshoot, Survive\rangle, S_0) \overset{[Rshoot]}{\longrightarrow} s \equiv \\
&s = do(\langle Rshoot, Die\rangle, do(\langle Rshoot, Survive\rangle, S_0)) \vee \\
&s = do(\langle Rshoot, Survive\rangle, do(\langle Rshoot, Survive\rangle, S_0)).
\end{aligned}
$$

From where we obtain:

$$
\begin{aligned}
&S_0 \overset{[Rshoot,Rshoot]}{\longrightarrow} s \equiv \\
&s = do(\langle Rshoot, Die\rangle, do(\langle Rshoot, Survive\rangle, S_0)) \vee \\
&s = do(\langle Rshoot, Survive\rangle, do(\langle Rshoot, Survive\rangle, S_0)).
\end{aligned}
$$

Thus, the previous sentence characterizes all situations that one can get to by following the behavioral sequence $[Rshoot, Rshoot]$.

By applying (18) and the successor state axiom (23), it follows that:

$$
\begin{aligned}
&Prf(Alive, [Rshoot, Rshoot], S_0) = \\
&\quad Prx(S_0, [Rshoot, Rshoot], \\
&\qquad do(\langle Rshoot, Survive\rangle, \\
&\qquad\quad do(\langle Rshoot, Survive\rangle, S_0))).
\end{aligned}
\tag{24}
$$

Using the probability assignments, along with (15)-(16), one obtains the expected result.

5. The Alternating Bit Protocol

This protocol has been designed to communicate a sender with a receiver. The sender has an ordered sequence of messages that need to be transmitted, in the correct order, to the receiver. The communication line can be faulty, and some transmissions be lost. To ensure that messages are received correctly, the sender and receiver use one bit to specify the status of the communication. The receiver starts with a bit, say b, and the sender starts with its negation, $\underline{b}$. The sender, sends the current message along with his own bit. When the receiver gets a message, he exchanges his bit by the one contained in the message from the sender. The receiver always acknowledges to the sender with his own bit. When the sender receives a bit from the receiver, he checks whether the bit is the same he has. If so, he interprets that as a confirmation that the message he is sending has been delivered. In that case, the sender changes his bit and proceeds with the next message.

Below, we formalize the behavior of the sender and receiver, and prove certain properties of the protocol.

5.1. *The Language*

In order to model the alternating bit protocol we specify the actions that are available to the sender and receiver in the style proposed by Reiter. For each agent (sender and receiver) we specify how their actions affect the world. Messages are taken to be of sort $\mathcal{D}$, and variables $m, m', \ldots$ are used to denote objects of this sort. The fluents are as follows:

- $sbit, rbit : \{0,1\} \to \mathcal{F}$: The fluent *sbit* is used to identify the bit that the sender is currently transmitting, along with a message, to the sender. The fluent *rbit* is similarly used to identify the bit that the receiver is currently acknowledging.
- $sent : \mathcal{D} \to \mathcal{F}$: This fluent holds when a given message has been successfully delivered.

We have the following non-deterministic actions:

- $send : \mathcal{D} \times \{0,1\} \to \mathcal{B}$: It transmits the current message and bit to the receiver.
- $ack : \{0,1\} \to \mathcal{B}$: It transmits the current acknowledge bit to the sender.

Also, we have the following outcome constants:

- *Loss*: The transmitted message is lost.
- *Dlvrd*: The transmitted message is delivered.

Thus, there are four types of actions in this example, these actions correspond to the message sending and acknowledgments followed by either a loss or a successful delivery.

State Constraints The following constraints establish that the sender and receiver can hold a single bit. Thus, *sbit* can hold for 0 or 1, but not both; the same is true for *rbit*.

$$holds(sbit(0), s) \equiv \neg holds(sbit(1), s), \tag{25}$$

$$holds(rbit(0), s) \equiv \neg holds(rbit(1), s). \tag{26}$$

Initial Situation Initially, the sender's bit is 0 and the receiver's bit is 1. Furthermore, no messages have been sent.

$$holds(sbit(0), S_0), \tag{27}$$

$$holds(rbit(1), S_0), \tag{28}$$

$$\neg holds(sent(m, b), S_0). \tag{29}$$

Effect Axioms After a successful delivery by the sender (execution of $\langle send(m,b), Dlvrd\rangle$), the receiver's bit becomes the same that is contained in the delivered message.

$$\begin{aligned} &Poss(\langle send(m,b), Dlvrd\rangle, s) \wedge holds(sbit(b), s) \supset \\ &\quad holds(rbit(b), do(\langle send(m,b), Dlvrd\rangle, s)), \end{aligned}$$

After a successful delivery of a message by the sender, the message is recorded as having been sent.

$$\begin{aligned} &Poss(\langle send(m,b), Dlvrd\rangle, s) \supset \\ &\quad holds(sent(m,b), do(\langle send(m,b), Dlvrd\rangle, s)), \end{aligned}$$

After an acknowledgement message from the receiver is successfully sent to the sender, the sender's bit changes.

$$Poss(\langle ack(b), Dlvrd\rangle, s) \wedge holds(rbit(b), s) \supset$$
$$\neg holds(sbit(b), do(\langle ack(b), Dlvrd\rangle, s)).$$

Action Precondition Axioms A message m and bit b can only be sent by the sender if the sender's bit is b.

$$Poss_{[}(send(m, b), s) \supset holds(sbit(b), s).$$

The receiver can only acknowledge with a bit that agrees with its own bit.

$$Poss_{[}(ack(b), s) \supset holds(rbit(b), s),$$

5.2. *The Theory of Action*

The theory of action results from applying Reiter's methodology to solve the frame problem[3], along with Pinto's extension to handle binary state constraints[14]. Let Σ denote the resulting theory, which contains the foundational axioms (1)-(3) and the following sets of axioms.

Axioms About the Initial Situation: The set T_{S_0}, with axioms (27)-(29), along with the state constraints (25) applied to S_0.

Successor State Axioms: These axioms are obtained by using the *explanation closure assumption*. They characterize the sufficient and necessary conditions for fluents to change after actions are performed in situations reached by performing legal actions; i.e., actions that are performed in situations that meet the action preconditions. The resulting axioms are†:

$$\begin{aligned}
&Poss(a, s) \supset holds(sbit(b), do(a, s)) \equiv \\
&\quad [\neg holds(rbit(b), s) \wedge a = \langle ack(\underline{b}), Dlvrd\rangle \vee \\
&\qquad holds(sbit(b), s) \wedge \neg(\exists\, b')\ a = \langle ack(b'), Dlvrd\rangle],
\end{aligned}$$

$$\begin{aligned}
&Poss(a, s) \supset holds(rbit(b), do(a, s)) \equiv \\
&\quad [holds(sbit(b), s) \wedge ((\exists\, m)\ a = \langle send(m, b), Dlvrd\rangle) \vee \\
&\qquad holds(rbit(b), s) \wedge \neg((\exists\, m, b)\ a = \langle send(m, b), Dlvrd\rangle)],
\end{aligned}$$

$$\begin{aligned}
&Poss(a, s) \supset \\
&\quad holds(sent(m, b), do(a, s)) \equiv \\
&\qquad a = \langle send(m, b), Dlvrd\rangle \vee holds(sent(m, b), s).
\end{aligned}$$

†If b is a variable with domain $\{0, 1\}$, we write $\underline{b}$ to denote the complement of b.

Action Precondition Axioms These axioms result from (8), and assuming that the necessary conditions for the performance of actions are also sufficient. This leads to:

$$
\begin{aligned}
&Poss(\langle send(m,b), Dlvrd\rangle, s) \equiv holds(sbit(b), s),\\
&Poss(\langle send(m,b), Loss\rangle, s) \equiv holds(sbit(b), s),\\
&Poss(\langle ack(b), Dlvrd\rangle, s) \equiv holds(rbit(b), s),\\
&Poss(\langle ack(b), Loss\rangle, s) \equiv holds(rbit(b), s).
\end{aligned}
$$

Possible Outcomes Axioms

$$Poss_{]}(e,t) \supset e = Loss \vee e = Dlvrd.$$

Probability Assignments

$$
\begin{aligned}
&Pr(Loss, do_{[}(send(m,b), s)) = K_1,\\
&Pr(Loss, do_{[}(ack(b), s)) = K_2.
\end{aligned}
$$

Thus, we consider that the probabilities of losing messages or acknowledgments are constant.

5.3. *Properties*

The sender will be ready to send a new message, whenever the sender bit is different from the receiver's. To identify these situations, we can introduce the *defined* fluent constant *Ready* with the following definition:

$$holds(Ready, s) =_{\mathrm{def}} [holds(sbit(0), s) \equiv holds(rbit(1), s)]$$

Remark 4 Let Σ be the theory of action, as described in the previous section.

- If we assume that we start with a message m that has not been sent, and in a legal state s where the bits for the sender and receiver are different, then, after making n attempts to deliver m, the delivery will have been successful in a later state s' if and only if s' is such that the bits of the sender and the receiver are the same.

  $$
  \begin{aligned}
  &\Sigma \models (\forall\, s, s', m, b, n).\\
  &\quad [s \geq S_0 \wedge \neg sent(m,s) \wedge holds(Ready, s) \wedge s \stackrel{send^n(m,b)}{\longrightarrow} s'] \supset\\
  &\qquad\qquad\qquad (sent(m,s') \equiv \neg holds(Ready, s')).
  \end{aligned}
  $$

  In other words, the sender can tell whether the message has been successfully transmitted, by checking the receiver's bit. In the communication protocol, this is done with the acknowledgment received by the sender from the receiver. After a successful acknowledgment, the sender will be ready for the next message.

- If the behavioral sequence $send^n(m,b)$ is performed in a situation where the message has not been sent, and the sender is ready, then the probability that the message m is successfully delivered is characterized with:

$$\begin{aligned}
&\Sigma \models (\forall m, n, b, s).\\
&\quad s \geq S_0 \wedge holds(Ready, s) \wedge \neg sent(m, s) \supset\\
&\qquad Prf(sent(m), send^n(m,b), s)\\
&\qquad = 1 - Prnf(sent(m), send^n(m,b), s)\\
&\qquad = 1 - Prx(s, send^n(m,b),\\
&\qquad\qquad do(\langle send, Loss\rangle^n(m,b), s))\\
&\qquad = 1 - [Poss_{]}(Loss, do_{[}(send(m,b), s))]^n.
\end{aligned}$$

- If the receiver sends an acknowledgment, this acknowledgment will have no effect, unless it is delivered in a state where *Ready* does not hold:

$$\begin{aligned}
&\Sigma \models (\forall s).\ s \geq S_0 \wedge holds(Ready, s) \wedge\\
&\quad Poss(\langle ack(b), Dlvrd\rangle, s) \supset\\
&\qquad [holds(f, s) \equiv holds(f, do(\langle ack(b), Dlvrd\rangle, s))],
\end{aligned}$$

$$\begin{aligned}
&\Sigma \models (\forall s).\ s \geq S_0 \wedge \neg holds(Ready, s) \wedge\\
&\quad Poss(\langle ack(b), Dlvrd\rangle, s) \supset\\
&\qquad holds(Ready, do(\langle ack(b), Dlvrd\rangle, s)).
\end{aligned}$$

Obviously, if the acknowledgment is lost, there is no change in the fluents.

- If the receiver has received a message that has not been successfully acknowledged (i.e., the sender is not ready), then the probability that the acknowledgment is finally delivered (resulting in a state where *Ready* holds) after n attempts is given by:

$$\begin{aligned}
&\Sigma \models (\forall m, n, b, s).\\
&\quad s \geq S_0 \wedge \neg holds(Ready, s) \wedge Poss_{[}(ack(b), s) \supset\\
&\qquad Prf(Ready, ack^n(b), s)\\
&\qquad\quad = 1 - Prnf(Ready, ack^n(b), S_0)\\
&\qquad\quad = 1 - Prx(s, ack^n(b),\\
&\qquad\qquad\quad do(\langle ack, Loss\rangle^n(b), s))\\
&\qquad\quad = 1 - [Poss_{]}(Loss, do_{[}(ack(b), s))]^n.
\end{aligned}$$

- Assuming that one starts in a situation where *Ready* holds, the probability that a message is successfully transmitted after n attempts, and that the message has been successfully acknowledged, after n' attempts, is $K_1^n \times K_2^{n'}$.

6. Comparison to Related Work

Lately, there has been a growing interest in the integration of uncertainty management and theories of action. In this section we compare our approach for the representation of indeterminism and uncertainty with two approaches that have been recently proposed in the literature[6,7]. These two approaches, as well as ours, are based upon the Situation Calculus.

Before we proceed with the actual comparison, let us say a few words about the semantic structure of our extended situation calculus. First, notice that the actual structure of situations is determined by the *foundational axioms* provided by Reiter[15]. This means that in any model of a situation calculus theory in our style, the situations that exist belong to a tree rooted at S_0. Every path in the tree corresponds to a possible development that includes behaviors (given by the agent that acts following the theoretical specification), and outcomes, represented by the events in the formalization. Given that the behaviors are given, one can compute the probability of being in a particular situation. An important aspect of the models of our probabilistic situation calculus, is that, given complete knowledge of the initial situation, and categorical knowledge about the domain, the situation calculus tree, as well as the state associated to each situation[‡], is completely determined.

Therefore, one distinguishing feature of the semantic structure of our probabilistic account of indetermination and uncertainty in the situation calculus, is that the probability space is defined over branches and paths in the situation tree. This is a major distinction between our approach and those discussed below. In each case, we will summarize the main conceptual differences, and advantages, if any, between the different views on how to integrate actions and probabilities.

6.1. *David Poole's Framework*

In David Poole's proposal[6] theories are written by integrating knowledge about actions and change, expressed in the situation calculus, with an *independent choice logic*, to model the probabilistic component. A theory, in this framework, is called an IC_{SC} theory. The main components of this theory are:

- A situation calculus theory of action written as an acyclic logic program[16] P. A subset of the rules in this program specifies sufficient conditions for fluents to be true after an action is performed. Each rule can be understood as an *effect axiom*, the only difference is that these axioms do not mention the *Poss* predicate. Actually, the *Poss* predicate is not present in Poole's framework and one can simply assume that this predicate is identically true. By performing predicate completion on the fluents mentioned on the head of these axioms, we obtain equivalences akin to Reiter's *successor state axioms*.

- A choice space C, which corresponds to a set of sets. Each element of C corresponds to a set of possible choices to be made by *nature*. Each choice is an

‡Here, we understand the state of a situation as the set of *fluents* that hold true in the situation.

atom (positive literal). The atoms in the choices do not unify with any of the heads of clauses of the acyclic program. For example, if we have coin tosses, we would have subsets $\{Heads(s), Tails(s)\}$ for each situation s in the choice space C. A particular development corresponds to a set of choices, where each choice is taken from one of the subsets of C. For instance, we might have the choices $Heads(S_0)$, $Tails(do(Toss, S_0))$, $Tails(do(Toss, do(Toss, S_0)))$, etc. Meaning that nature chose to give us heads in the first toss and two tails in sequence, the three tosses started in S_0. The requirement that the possible choices do not unify with the heads of the clauses means that these choices cannot be obtained by logical reasoning. Rather, the choices are given externally by *nature*.

A set of choices C_h from a choice space C corresponds to a mapping such that for each $c \in C$ exactly one element $c_h \in c$ is also a member of C_h. Given a set of choices C_h, and the acyclic logic program, the stable model of the program $P \cup C_h$ provides the model of a specification of the theory IC_{SC}. Each possible set of choices leads to another model of the theory.

In Poole's framework, each subset of the choice space is assigned a probability distribution. Furthermore, the choices are independent. Therefore, each set of choices is assigned a probability, which one can attach to the corresponding stable model. Thus, if one wants to compute the probability that some fluent is true after performing an action in S_0 one needs to sum the probabilities associated to all the models in which the fluent is true.

The main conceptual difference between Poole's approach and ours is in the structure of the models of the logical theory. As mentioned before, the behavior of the agent along with the events generated by nature, are represented in our models by single path in the situation tree structure. Associated with each path, there is a probability associated to the particular development it represents. A single model of our theories represents all possible developments associating a probability to each of them. On the other hand, in Poole's approach, the actual development is fixed in a any single model of the logical theory, and a probability is *externally* assigned, by the choices made by nature, to each of the models. From a conceptual point of view, this difference has important consequences:

- The most important difference is that, in our approach, one can reason within the logical language, about the probable outcome of a given behavior. Therefore, if one added some utility measure to the situations, one can, on the basis of probabilistic reasoning, decide what the best course of action might be. The important point, is that this can be done within the logical framework. Furthermore, one can also analyze the properties of agent programs by providing, externally, a fixed behavior and answering questions such as the "what is the probability that certain formula will be satisfied after the behavior is complete." In Poole's framework, any probabilistic reasoning must be done outside the logical theories. Therefore, one cannot specify the behavior of agents whose decisions are based on an analysis of the current situation.

- A drawback of Poole's framework has to do with the modeling of the choice space and the way in which the choices are made by nature. The problem is the assumption that the choices are independent of each other. This means, for instance, that the probability distributions on the outcomes of a given behavior cannot depend on the situation in which the behavior is actually performed. This is a severe limitation. For example, imagine a robot moving on a floor in which there is a rug. When the robot moves over the rug, the probability that his motion actions will not have the intended effects is different from the distribution associated to his motion over a tiled floor. In this case, the probability distributions of his errors should change depending upon the situation the robot reaches.

- Poole addresses some issues that we have not addressed in this article. In particular, issues related to measures of utilities and to the ability of sensing the environment. Incorporating measures of utilities in our framework should be straightforward. The issue of sensing will be addressed in the last section of the article.

6.2. *Bacchus, Halpern and Levesque's approach.*

Although the approach advanced by Bacchus, Halpern and Levesque[7] (BH&L), is much closer to ours than the approach of Poole's, there are substantive differences. BH&L's approach is also situation calculus based, and the most important differences stem from differing semantical interpretations of the situation structure.

The initial motivation for the different interpretation of the semantic structure of situations comes from the view that uncertainty should be understood as lack of knowledge on the reasoning agent's behalf. Thus, knowledge is at the core of their approach to the representation of uncertainty. The approach is based on ideas taken from the modal logics of knowledge and belief[17]. In these modal logics, there is a *possible worlds* semantics in which an agent's state of knowledge is represented as the set of worlds that the agent considers possible, given his knowledge. An agent knows something, if that something is true in every world she considers possible. Thus, the more worlds the agent considers possible, the lesser the agent knows. What the agent considers possible in a given world is represented by a binary accessibility relation between worlds. Scherl and Levesque[18], following a related approach by Moore[19], introduced a predicate K to represent the accessibility relation. From this point of view, a world would be a situation, and $K(S, S')$ means that, as far as the agent knows in situation S, she could be in situation S'. To integrate this idea into the situation calculus, one starts with a set of possible initial situations. Each such situation is the root of a situation tree structure (the trees are pairwise disjoint). There is a distinguished initial situation, S_0, which is regarded as the *real* initial situation. Every situation K-accessible from S_0 is the initial situation of some situation calculus tree. Scherl and Levesque introduce the notion that some actions *produce knowledge*.

For example, suppose that our agent does not know what the weather is like outside. For what she knows she could be in a sunny situation S, or in a rainy situation S'. If the agent looks out the window and sees that it is raining, she will come to know this fact. This will be reflected in the interpretation of the accessibility relation between $do(look, S)$ and $do(look, S')$. Thus, it might have been the case that $K(S, S')$ was true; but, as a consequence of looking, we would have $\neg K(do(look, S), do(look, S')$.

It can be stated that in BH&L's framework, uncertainty is *agent centered*. Indeed, there is one possible world, which is the real world. In this world, everything is certain. Other worlds are possible from the point of view of what the agent knows about it. If we take the set of situations that the agent considers possible, each one of these worlds is given a probability (actually, a *degree of belief*). Thus, in the example above, the agent might assign a probability of 0.5 to all the situations that she considers possible and in which it is raining, and 0.5 probability to all the other situations she considers possible. After looking, the probabilities assigned to the situations that result from looking are adjusted to reflect the fact that more knowledge is acquired.

The treatment given in BH&L to uncertain actions is in the same spirit as ours. Although, the realization is quite different. For instance, a coin toss, in both approaches, is represented by two separate actions. While we would have the actions as $\langle Toss, Heads\rangle$ and $\langle Toss, Tails\rangle$, in the syntax of BH&L one writes two terms such as $Toss(Heads)$ and $Toss(Tails)$, plus the fact that both actions are *indistinguishable*. The concept of *action indistinguishability* is central. Assume that S and S' are two situations that the agent considers possible. Furthermore, assume that A and A' are two indistinguishable actions. Then, after performing A (or A'), the situations $do(A, S)$, $do(A', S)$, $do(A, S')$ and $do(A', S')$, should all be considered possible. Also, the degrees of belief assigned to the situations that result from all situations considered possible, after performing an action or its indistinguishable counterparts, are adjusted according to rules compatible with the treatment of probabilities.

Thus, as should be clear from the above description, BH&L's framework is deeply intertwined with their treatment of knowledge. Ours, on the other hand, is completely orthogonal to this issue. The main consequences of this difference has to do with the form of reasoning that is supported in each approach. For instance, an agent implemented based on a specification written in our approach, would be able to evaluate the expected outcome of a given behavior. In BH&L's, the agent would need to reason about what its state of knowledge would be after performing the same behavior. Thus, in the second case, some form of introspection is necessary.

7. Limitations, Conclusions and Future Work

In this article we have presented a novel approach to modeling action and change in the presence of non-determinism. This approach has been inspired by the work on the modeling probabilistic processes[11]. Our main contribution is to integrate non-determinism to an existing language for knowledge representation. More specifically, in the area of the development of logical languages to represent action and change.

From a representational point of view, an important limitation of the approach that we have presented is the fact that we require that the space of *events* that can arise in response to a given *behavior* be finite. This requirement arises because our logical language does not incorporate a rich language for real numbers. For instance, axiom (9) would turn into an integral instead of a finite sum. This issue is of great relevance in many applications. For example, consider a wheeled robot that issues an order to its controllers to move ahead by 1 meter. The outcome of such a motion will not be precise and might have a continuous probability distribution that will depend on the context in which the order is issued. It appears, that in any application in which agents move in physical space will present the requirement to handle continuous properties. In our current research[§], we follow the semantic approach presented in this article and extend it in order to incorporate continuous change. This approach is completely general and integrates both, the possibility of discrete and non-discrete events. Furthermore, we are developing an approach to integrate logical reasoning and mathematical analysis in order to deal with real continuous functions, integrals, etc. The basic approach is to write theories of action in a logical language that integrates standard functions for the real numbers. In this language, we expect to be able to do logical reasoning by means of rewrite rules and mathematical analysis using the tools for reasoning about the reals (e.g., *mathematica*).

Another limitation stems from the basic assumption that we have prior knowledge of probability distributions of the outcomes of the transitions. In general this assumption might be incorrect. For instance, as one of the reviewers of this article pointed out, certain uncertain events are not random. For instance, whether or not a given individual is in his office or not, does not have a meaningful probability distribution. To handle this type of uncertainty, one needs to incorporate, as in BH&L's approach, knowledge. Thus, if we don't know if the subject is in his office, we might simply call and find out. However, on occasion, we might be forced to make a decision without recourse to knowledge finding actions; in this case, we might simply appeal to a subjective likelihood estimate.

Another perceived limitation is that we have made *independence assumptions* of the outcomes (witness formula (16)), in which the probabilities at each step in the development of the situations are simply multiplied. This assumption has indeed been made in the examples provided in this article. However, this is not an assumption made in the overall framework. Indeed, the Pr function, used in order to specify probability assignments, depends explicitly on the transition. Thus, one can define a different probability distribution depending upon what things hold or do not hold in the given transition. Thus, the dependence can be explicitly stated

[§]In progress[13].

in these terms.

In future research, we would like to study the possibility of having agents *learn* probability distributions as a result of their interaction with the world. This *statistical* form of learning would allow us to extend the applicability of our approach to domains in which the parameters of certain distributions might be unknown. For instance, imagine that one wheel of a robot is out of tune, knowing that the actual output of the wheel distributes normally, one might want to adjust its mean as it moves around the world and discovers where it goes after performing some motion action.

8. Acknowledgments

This work was partially supported by FCT, the PRAXIS XXI Projects PRAXIS/P/MAT/10002/1998 ProbLog, PCEX/P/MAT/46/96 ACL and 2/2.1/TIT/1658/95 LogComp, as well as by the ESPRIT IV Working Groups 22704 ASPIRE and 23531 FIREworks. Also, Javier Pinto was supported by the FONDECYT grants #198-0945 and #199-0089.

1. J. Pinto, "Causality in theories of action", in *Common-Sense '98*, eds. R. Miller and M. Shanahan, Queen Mary and Westfield College, University of London, Jan. 1998, pp. 349–364. Also found at `URL = http://www.ida.liu.se/ext/etai/-nj/fcs-98/349/paper.ps`.
2. J. McCarthy and P. J. Hayes, "Some philosophical problems from the standpoint of artificial intelligence", in *Machine Intelligence* **4**, eds. B. Meltzer and D. Michie (Edinburgh University Press, Edinburgh, 1969) pp. 463–502.
3. R. Reiter, "The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression", in *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy* (Academic Press, San Diego, 1991) pp. 359–380.
4. C. Pereira, F. Garcia, J. Lang and R. Martin-Clouaire, "Planning with graded non-deterministic actions: A possibilistic approach", *International Journal of Intelligent Systems* ***12*** (1997).
5. A. Saffiotti, K. Konolige and E. Ruspini, "A multivalued logic approach to integrating planning and control", *International Journal of Intelligent Systems* **76** (1995) 481–526.
6. D. Poole, "Decision theory, the situation calculus, and conditional plans", *Linköping Electronic Articles in Computer and Information Science* **3**, 8 (1998). `http://www.ep.liu.se/ea/cis/1998/008/`.
7. F. Bacchus, J. Halpern and H. Levesque, "Reasoning about noisy sensors and effectors in the situation calculus", *Artificial Intelligence* (1999). To appear.
8. J. McCarthy, "Situation, actions, and causal laws", Tech. Rep. Memo 2, Stanford Artificial Intelligence Project, 1963.
9. F. Lin and R. Reiter "State constraints revisited", *Journal of Logic and Computation* **4**, 5 (1994) 655–678. Special Issue on Actions and Processes.
10. J. Pinto, "Occurrences and narratives as constraints in the branching structure of the situation calculus", *Journal of Logic and Computation* **8** (1998) 777–808.
11. C. Baier and M. Kwiatkowska, "Automatic verification of liveness properties of randomized systems (extended abstract)", in *Proceedings of PODC'97* (June 1997).
12. G. Shafer, *The art of causal conjecture*, (The MIT Press, 1996).
13. J. Pinto, A. Sernadas, C. Sernadas, P. Mateus and A. Pacheco, "Probabilistic reasoning

in the situation calculus", in progress.

14. J. Pinto, "Temporal reasoning in the situation calculus", Ph. D. Thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, Feb. 1994. URL = ftp://ftp.cs.toronto.edu/~cogrob/jpThesis.ps.Z.
15. R. Reiter, *KNOWLEDGE IN ACTION: Logical foundations for describing and implementing dynamical systems*, Book Draft, available from `http://www.cs.utoronto.ca/~cogrobo`, 1999.
16. K. Apt and M. Bezem, "Acyclic programs", in *Logic Programming: Proceedings of the Seventh International Conference.* eds. D. Warren and P. Szeredi (1990) pp. 617–633.
17. R. Fagin, J. Halpern, Y. Moses and M. Vardi, *Reasoning about knowledge*, (The MIT Press, 1995).
18. R. Scherl and H. Levesque, "The frame problem and knowledge producing actions", in *Proceedings AAAI-93* (AAAI press, Washington, D.C., July 1993), pp. 689–695.
19. R. C. Moore, "A formal theory of knowledge and action", in *Formal Theories of the Commonsense World*, eds. J. B. Hobbs and R. C. Moore (Ablex Publishing Corp., Norwood, New Jersey, 1985) ch. 9, pp. 319–358.