Non-determinism and Uncertainty in the Situation Calculus

Javier Pinto  
Departamento de Ciencia de la Computación, Pontificia Universidad Católica de Chile.  
Email: jpinto@acm.org

Amílcar Sernadas  
Cristina Sernadas  
Paulo Mateus  
CMA, Departamento de Matemática, Instituto Superior Técnico, Portugal.  
Email: {acs,css,pmat}@math.ist.utl.pt

Abstract

A novel approach is presented to modeling action and change within the Situation Calculus in the presence of both non-determinism and probabilistic behavior. Two examples are given to illustrate the approach: the Russian roulette and the alternating bit protocol.

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 dartboard 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 he 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 (Pinto 1998), 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
sender to a receiver. The non-determinism that arises is due to losses in the communication channels. Finally we present our conclusions and discuss our future work on this subject.

**Standard Situation Calculus**

A many sorted second order language. The sorts are \( \mathcal{A} \) for primitive actions, \( \mathcal{S} \) for situations, and \( \mathcal{F} \) for fluents (i.e., we chose to reify fluents). We also include a sort \( \mathcal{D} \) for domain objects. We also 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 \( Poss \subseteq \mathcal{A} \times \mathcal{S} \). Also, we make use of \( \mathcal{R}_0^+ \), the positive reals, as an interpreted sort.

The following are foundational axioms for the situation calculus:

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

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

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

\[
(\forall a,s,s') s < do(a,s') \equiv Poss(a,s') \land s \leq s'. \tag{4}
\]

Axiom (1) is an induction axiom, inspired by Peano’s induction axiom for the natural numbers. In a second order language, \( \varphi \) is taken to be a predicate variable. In a first order axiomatization, we take the axiom to be an induction schema and work with the standard models of the theory.

**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.

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 domain of outcomes, whenever there are free generators for \( \mathcal{E} \).

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:

\[
(\forall a) (\exists b, e) a = \langle b, e \rangle. \tag{5}
\]

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

\[
\langle b, e \rangle = \langle b', e' \rangle \supset b = b' \land e = e'. \tag{6}
\]

We also need the structural axioms:

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

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

From where it follows:

**Remark 1** From the foundational axioms (1)-(4), and axioms (5)-(7), it follows that:

\[
\langle b, e, s \rangle = do_{\langle b, e \rangle}(b', s') \supset b = b' \land s = s',
\]

and

\[
\langle b, e, t \rangle = do_{\langle e, t \rangle}(e', t') \supset e = e' \land t = t'.
\]

For \( Poss \) we have:

\[
Poss(\langle b, e \rangle, s) = Poss(b, s) \land Poss(e, do(b, s)). \tag{8}
\]

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.

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

\[
Poss(b, s) \supset (\exists e) Poss(\langle b, e \rangle, s).
\]

**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 \( \text{Toss} \) of sort \( \mathcal{B} \). The outcomes can be heads or tails, modeled with the constants \( \text{Heads} \) and \( \text{Tails} \) of sort \( \mathcal{E} \).

**Precondition axioms**

\[
Poss(\langle \text{Toss}, \text{Heads} \rangle, s),
\]

\[
Poss(\langle \text{Toss}, \text{Tails} \rangle, s).
\]

\[\text{In the rest of the article, all free variables that appear in formulas are assumed to be universally quantified with maximum scope.}\]
Effect axioms

\[ \text{Poss}((\text{Toss}, \text{Heads}), s) \supset \text{holds}(\text{Head-Up}, \text{do}((\text{Toss}, \text{Heads}), s)) \]
\[ \text{Poss}((\text{Toss}, \text{Tails}), s) \supset \text{holds}(\text{Tail-Up}, \text{do}((\text{Toss}, \text{Tails}), s)). \]

State constraints

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

Possible outcomes axioms

\[ \text{Poss}(e, \text{do}((\text{Toss}, s))) \supset e = \text{Heads} \lor e = \text{Tails}. \]

Probability assignments

\[ \text{Pr}(\text{Heads}, \text{do}((\text{Toss}, s))) = \text{Pr}(\text{Tails}, \text{do}((\text{Toss}, s))). \]

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:

\[ \text{do}((\text{Toss}, \text{Heads}), \text{do}((\text{Toss}, \text{Tails}), S_0)) \]

corresponds to one possible outcome of the behavioral sequence \(\text{Toss}, \text{Toss}\). 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:

\[ \neg \text{Poss}([\,], s). \] (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 countable 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 \([s, \vec{b}, s']\) 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:

\[ s^0 \xrightarrow{b_1, \ldots, b_n}s^n \equiv (\exists e_1, \ldots, e_n)
\]
\[ s^n = \text{do}([b_1, e_1], \ldots, (b_n, e_n)], s^0)) \land s^0 \leq s^n. \]

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.

\[ s \xrightarrow{\vec{b}} s \equiv \vec{b} = []. \] (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:

\[ \vec{b} \neq [] \supset [s \xrightarrow{b} s' \equiv s < s' \land (\exists s'', e)\]
\[ s'' = \text{do}((\text{first}(\vec{b}), e), s) \land s' \overset{\text{tail}(\vec{b})}{\rightarrow} s']. \] (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:

\[ \text{successful}(\vec{b}, s) =_{df} (\exists s') s \xrightarrow{\vec{b}} s'. \]

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

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

\[ \neg s \xrightarrow{\vec{b}} s' \supset \text{Pr}(s, \vec{b}, s') = 0.0, \] (14)
\[ \text{Pr}(s, [\,], s) = 1.0, \] (15)
\[ \text{do}(b, e, s) \leq s' \land s \xrightarrow{[b][\vec{b}]} s' \supset \]
\[ \text{Pr}(s, [b][\vec{b}], s') = \]
\[ \text{Pr}(e, \text{do}(b, s)) \times \text{Pr}(\text{do}((b, e), s), \vec{b}, s'). \] (16)
• \( Prb : B^* \times S \rightarrow R_0^+ \). We write \( Prb(\vec{b},s) \) to denote the probability that it is possible that the non-deterministic action \( \vec{b} \) is performed successfully in \( s \).

\[
Prb(\vec{b},s) = \sum_{s' \in S} Prx(s,\vec{b},s').
\]

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 \( B \) terms. This formula can only be expressed when \( \vec{b} \) is a fixed and finite sequence of \( B \) terms. The same restrictions apply to (18) and (21) below. Alternatively, this summation can be expressed by using a recursive specification.

• \( Prf : F \times B^* \times S \rightarrow R_0^+ \). 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 \( s \).

\[
Prf(f,\vec{b},s) = \sum_{s' \in S} Prx(s,\vec{b},s') \times \overline{\text{holds}(f,s')},
\]

where we have used the function symbol \( \overline{\text{holds}} : F \times S \rightarrow R_0^+ \), which is simply defined to be equal to 1.0 for a fluent \( f \) and situation \( s \), if the fluent \( f \) holds in \( s \), and is defined to be 0.0 otherwise. Thus, we write:

\[
\overline{\text{holds}}(f,s) = 1 \equiv \text{holds}(f,s),
\]

\[
\overline{\text{holds}}(f,s) = 0 \equiv \neg\text{holds}(f,s).
\]

From (18), it follows that probability \( Prf(f,\vec{b},s) \) is 0 if \( \vec{b} \) may not be performed in \( s \).

Also, we define \( Prnf : F \times B^* \times S \rightarrow R_0^+ \). 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 \( s \).

\[
Prnf(f,\vec{b},s) = \sum_{s' \in S} Prx(s,\vec{b},s') \times (1 - \overline{\text{holds}}(f,s')).
\]

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:

\[
CPrf(f,\vec{b},s) = \frac{Prf(f,\vec{b},s)}{Prb(\vec{b},s)}.
\]

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 \( \text{Rshoot} \) for rotate and shoot (of sort \( B \)), with two possible outcomes \( \text{Die} \) and \( \text{Survive} \) of sort \( E \). 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 \( \text{Rshoot}, \text{Die} \) and \( \text{Survive} \) were specified above. We also have the single fluent constant \( \text{Alive} \). Again, for simplicity, we assume that the revolver is always properly loaded, and don’t characterize its behavior.

Action precondition axioms

\[
\text{Poss}(\langle \text{Rshoot}, \text{Die} \rangle, s) \equiv \text{holds}(\text{Alive}, s),
\]

\[
\text{Poss}(\langle \text{Rshoot}, \text{Survive} \rangle, s) \equiv \text{holds}(\text{Alive}, s).
\]

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:

\[
\text{Poss}(\langle \text{Rshoot}, \text{Die} \rangle, s) \supset \neg\text{holds}(\text{Alive}, \text{do}(\langle \text{Rshoot}, \text{Die} \rangle, s)).
\]

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

\[
\text{Poss}(a,s) \supset [\text{holds}(\text{Alive}, \text{do}(a,s)) \equiv \neg\text{holds}(\text{Alive}, s) \land a \neq \langle \text{Rshoot}, \text{Die} \rangle].
\]

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

\[
\text{holds}(\text{Alive}, S_0).
\]

Possible outcomes axiom

\[
\text{Poss}(e, \text{do}(\text{Rshoot}, s)) \supset e = \text{Die} \lor e = \text{Survive}.
\]

Probability assignments

\[
\text{Pr}(\text{Die}, \text{do}(\text{Rshoot}, s)) = 1/6,
\]

\[
\text{Pr}(\text{Survive}, \text{do}(\text{Rshoot}, s)) = 5/6.
\]

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

\[
\text{Prf} (\text{Alive}, [\text{Rshoot}, \text{Rshoot}], S_0).
\]

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:

\[ S_0^{[R\text{shoot}]} \equiv s \equiv \]

\[ s = \text{do}((R\text{shoot, Die}), S_0) \lor \]

\[ s = \text{do}((R\text{shoot, Survive}), S_0). \]

Also,

\[ \neg(\text{do}((R\text{shoot, Die}), S_0)^{[R\text{shoot}]} s), \]

and

\[ \text{do}((R\text{shoot, Survive}), S_0)^{[R\text{shoot}]} s \equiv \]

\[ s = \text{do}((R\text{shoot, Die}), \text{do}((R\text{shoot, Survive}), S_0)) \lor \]

\[ s = \text{do}((R\text{shoot, Survive}), \text{do}((R\text{shoot, Survive}, S_0)). \]

From where we obtain:

\[ S_0^{[R\text{shoot, R\text{shoot}}]} s \equiv \]

\[ s = \text{do}((R\text{shoot, Die}), \text{do}((R\text{shoot, Survive}), S_0)) \lor \]

\[ s = \text{do}((R\text{shoot, Survive}, \text{do}((R\text{shoot, Survive}, S_0)). \]

Thus, the previous sentence characterizes all situations that one can get to by following the behavioral sequence \([R\text{shoot, R\text{shoot}}].\]

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

\[ \text{Prf}(\text{Alive}, [R\text{shoot, R\text{shoot}}], S_0) = \]

\[ \text{Prf}(S_0, [R\text{shoot, R\text{shoot}}], \]

\[ \text{do}((R\text{shoot, Survive}, \text{do}((R\text{shoot, Survive}, S_0))). \]

(24)

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

The Alternating Bit Protocol

This protocol has been designed to communicate a sender with a receiver. The sender has a sequence of messages that need to be transmitted in order to a 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, \(\bar{b}\). The sender, sends the current message along with his own bit. When the receiver gets a message, he changes his bit by the one contained in the message. 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.

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 \(D\), and variables \(m, m', \ldots\) are used to denote objects of this sort. The fluents are as follows:

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

We have the following non-deterministic actions:

- \(\text{send} : D \times \{0, 1\} \rightarrow \mathcal{B}\): It transmits the current message and bit to the receiver.
- \(\text{ack} : \{0, 1\} \rightarrow \mathcal{B}\): It transmits the current acknowledgement bit to the sender.

Also, we have the following outcome constants:

- \(\text{Loss}\): The transmitted message is lost.
- \(D\text{lvrd}\): 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

\[ \text{holds}(s\text{bit}(0), s) \equiv \neg\text{holds}(s\text{bit}(1), s), \] (25)

\[ \text{holds}(r\text{bit}(0), s) \equiv \neg\text{holds}(r\text{bit}(1), s). \] (26)

Initial Situation

\[ \text{holds}(s\text{bit}(0), S_0), \] (27)

\[ \neg\text{holds}(\text{sent}(m, b), S_0), \] (28)

\[ \text{holds}(r\text{bit}(1), S_0). \] (29)

Effect Axioms

\[ \text{Poss}(\text{send}(m, b), D\text{lvrd}), s) \land \text{holds}(s\text{bit}(b), s) \supset \]

\[ \text{holds}(r\text{bit}(b), do((\text{send}(m, b), D\text{lvrd}), s)), \]

\[ \text{Poss}(\text{send}(m, b), D\text{lvrd}), s) \supset \]

\[ \text{holds}(\text{sent}(m, b), do((\text{send}(m, b), D\text{lvrd}), s)), \]

\[ \text{Poss}(\text{ack}(b), D\text{lvrd}), s) \land \text{holds}(r\text{bit}(b), s) \supset \]

\[ \neg\text{holds}(s\text{bit}(b), do((\text{ack}(b), D\text{lvrd}), s)). \]

Action Precondition Axioms

\[ \text{Poss}(\text{ack}(b), s) \supset \text{holds}(r\text{bit}(b), s), \]

\[ \text{Poss}(\text{send}(m, b), s) \supset \text{holds}(s\text{bit}(b), s). \]
The Theory of Action

The theory of action results from applying Reiter’s methodology to solve the frame problem (Reiter 1991), along with Pinto’s extension to handle binary state constraints (Pinto 1994). Let $\Sigma$ denote the resulting theory, which contains the foundational axioms (1)-(4) 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\(^2\):

\[
\begin{align*}
Poss(a,s) & \supset holds(sbit(b), do(a,s)) \equiv \\
& [-holds(rbit(b), s) \land a = \langle{\text{ack}(b), Dlvr}\rangle \lor \\
& holds(sbit(b), s) \land -((\exists b') a = \langle{\text{ack}(b'), Dlvr}\rangle),
\end{align*}
\]

\[
\begin{align*}
Poss(a,s) & \supset holds(rbit(b), do(a,s)) \equiv \\
& [holds(sbit(b), s) \land \\
& ((\exists m) a = \langle{\text{send}(m,b), Dlvr}\rangle) \lor \\
& holds(rbit(b), s) \land \\
& -((\exists m) a = \langle{\text{send}(m,b), Dlvr}\rangle),
\end{align*}
\]

\[
\begin{align*}
Poss(a,s) & \supset holds(sent(m,b), do(a,s)) \equiv \\
& a = \langle{\text{send}(m,b), Dlvr}\rangle \lor holds(sent(m,b), s).
\end{align*}
\]

**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:

\[
\begin{align*}
Poss((send(m,b), Dlvr), s) & \equiv holds(sbit(b), s), \\
Poss((send(m,b), Loss), s) & \equiv holds(sbit(b), s), \\
Poss((ack(b), Dlvr), s) & \equiv holds(rbit(b), s), \\
Poss((ack(b), Loss), s) & \equiv holds(rbit(b), s).
\end{align*}
\]

**Possible Outcomes Axioms**

\[
Poss(e,t) \supset e = Loss \lor e = Dlvr.
\]

\(^2\)If $b$ is a variable with domain $\{0,1\}$, we write $\bar{b}$ to denote the complement of $b$.

**Probability Assignments**

\[
\begin{align*}
Pr(Loss, do(send(m,b), s)) & = K_1, \\
Pr(Loss, do(ack(b), s)) & = K_2.
\end{align*}
\]

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

**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:

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

**Remark 4** Let $\Sigma$ 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 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.

\[
\begin{align*}
\Sigma & \models (\forall s, s', m, b, n) \\
& [s \geq S_0 \land \neg sent(m, s) \land \\
& holds(Ready, s) \land s \overset{send^n(m,b)}{\rightarrow} s'] \supset \\
& (sent(m, s') \equiv \neg holds(Ready, s')).
\end{align*}
\]

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:

\[
\begin{align*}
\Sigma & \models (\forall m, n, b, s) \\
& s \geq S_0 \land holds(Ready, s) \land \neg sent(m, s) \supset \\
& Prf(sent(m), send^n(m,b), s) \\
& = 1 - Prf(sent(m), send^n(m,b), s) \\
& = 1 - Prx(s, send^n(m,b), \\
& do((send, Loss)^n(m,b), s)) \\
& = 1 - [Poss(Loss, do(send(m,b), s))]^n.
\end{align*}
\]

- If the receiver sends an acknowledgment, this acknowledgment will have no effect, unless it is deliv-
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 (e.g., (Baier & Kwiatkowska 1997)). Our main contribution is to integrate non-determinism to an existing language (Baier & Kwiatkowska 1997)). Our main contribution is to integrate non-determinism to an existing language (Baier & Kwiatkowska 1997)). Our main contribution is to integrate non-determinism to an existing language (Baier & Kwiatkowska 1997)).

\[ \Sigma \models (\forall s) \, s \geq S_0 \land \neg \text{holds}(\text{Ready}, s) \land \\
\text{Poss}(\langle \text{ack}(b), \text{Dlvrd} \rangle, s) \supset \\
[\text{holds}(f, s) \equiv \text{holds}(f, \text{do}(\langle \text{ack}(b), \text{Dlvrd} \rangle, s))], \]

\[ \Sigma \models (\forall m, n, b, s) \, s \geq S_0 \land \neg \text{holds}(\text{Ready}, s) \land \\
\text{Poss}(\langle \text{ack}(b), \text{Dlvrd} \rangle, s) \supset \\
\text{Prf}(\text{Ready}, \text{ack}^n(b), s) = 1 - \text{Prnf}(\text{Ready}, \text{ack}^n(b), S_0) = 1 - \text{Prx}(s, \text{ack}^n(b), \\
\text{do}(\langle \text{ack}, \text{Loss} \rangle^n(b), s)) = 1 - [\text{Poss}(\text{Loss}, \text{do}(\langle \text{ack}(b), s \rangle)]^n. \]

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 (e.g., (Baier & Kwiatkowska 1997)). 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.

In our immediate future work, we would like to generalize this approach by integrating explicit axioms of probabilities into the logical language taking into account the work of Bacchus (Bacchus 1990). It is also necessary to relate our approach, on the one hand to current approaches to model indeterminism within the Theories of Action community (e.g., (Lin 1996)), and on the other to approaches that integrate probabilities within other logical frameworks (e.g., (Hansson & Jonsson 1994; Bianco & de Alfaro 1995)).

Acknowledgments

This work was partially supported by FCT, the PRAXIS XXI Projects PCEX/P/MAT/46/96 ACL and 2/21/TIT/1658/95 LogComp, the ProbLog initiative of CMA, as well as by the ESPRIT IV Working Groups 22704 ASPIRE and 23531 FIREworks.

References


