Non-determinism and Uncertainty in the Situation Calculus

Javier Pinto*, Amilcar Sernadas†,
Cristina Sernadas† and Paulo Mateus†

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

†Departamento de Matemática, Instituto Superior Técnico, Universidad Técnica de Lisboa, Portugal. Email {acs,css,pmat}@math.ist.utl.pt.

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. On 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 [8], 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, as in [8], the solutions to the *frame* and related problems still apply.

Our view of indeterminacy is independent of the formalism that 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 [6], and relying on Reiter's approach to deal with the frame problem [9].

The article is organized as follows. In section 2 we present the logical framework that constitutes the basis for our technical proposal. In section 3 we extend the language presented in section 2 in order to accommodate indeterminate actions (as pairs), and to give probabilities to the outcomes. In section 4, we deal with ordered sequences of indeterminate actions. Later, in section 5, we model the *alternatig 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. Finally, in section 6, we present our conclusions and discuss our future work on this subject.

2 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, φ 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.

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.

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 $do_{[} : \mathcal{B}\times\mathcal{S} \to \mathcal{T}$, and $do_{]} : \mathcal{E}\times\mathcal{T} \to \mathcal{S}$ 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 $\langle\rangle : \mathcal{B} \times \mathcal{E} \to \mathcal{A}$. 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[1]:

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

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

We also need the structural axioms:

$$do(\langle b, e\rangle, s) = do_{]}(e, do_{[}(b, s)), \tag{6}$$
$$(\forall\, t)\ (\exists\, b, s)\ t = do(b, s). \tag{7}$$

From where it follows:

Remark 3.1 *From the foundational axioms (1)-(4), 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.

Remark 3.2 *From (9) and (10), and the probability axioms, it follows that:*

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

□

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

$$Poss(\langle Toss, Heads\rangle, s),$$
$$Poss(\langle Toss, Tails\rangle, s).$$

Effect axioms

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

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 $| : \mathcal{B} \times \mathcal{B}^* \to \mathcal{B}^*$ 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 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 $\rightarrow (s, \vec{b}, s')$ 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:

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

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:

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

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

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

Starting from the transition probabilities introduced in section 3, 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:

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

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

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

- $Prb : \mathcal{B}^* \times \mathcal{S} \rightarrow \mathcal{R}_0^+$. 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 sumation 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.

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 4.1

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

$$((\exists\, s')\ s \leq s' \wedge s \xrightarrow{\vec{b}} s') \supset Prb(\vec{b}, s) = 1.0.$$

□

- 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 that gets 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:

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

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:

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

Also,

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

and

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

From where we obtain:

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

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:

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

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 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 in order to specify the status of the communication. The receiver starts with a bit, say B, and the transmitter starts with its negation, $\overline{B}$. The receiver always acknowledges to the receiver with its own bit. The sender, sends the current message along with its own bit. When the receiver gets a message, it changes its bit by the one contained in the message. When the sender receives a bit from the receiver it checks whether it is the same it has. If so, it interprets that as a confirmation that the message it is sending has been delivered. In that case, it changes its 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:

- $\mathit{sbit}, \mathit{rbit} : \{0,1\} \to \mathcal{F}$: 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.
- $\mathit{sent} : \mathcal{D} \to \mathcal{F}$: This fluent holds when a given message has been successfully delivered.

We have the following non-deterministic actions:

- $\mathit{send} : \mathcal{D} \times \{0,1\} \to \mathcal{B}$: It transmits the current message and bit to the receiver.
- $\mathit{ack} : \{0,1\} \to \mathcal{B}$: 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 acknowledgements followed by either a loss or a succesful delivery.

State Constraints

$$holds(sbit(0), s) \equiv \neg holds(sbit(1), s), \tag{25}$$
$$holds(rbit(0), s) \equiv \neg holds(rbit(1), s). \tag{26}$$

Initial Situation

$$holds(sbit(0), S_0), \tag{27}$$
$$\neg holds(sent(m, b), S_0), \tag{28}$$
$$holds(rbit(1), S_0). \tag{29}$$

Effect Axioms

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

Action Precondition Axioms

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

5.2 The Theory of Action

The theory of action results from applying Reiter's methodology to solve the frame problem [9], along with Pinto's extension to handle binary state constraints [7]. Let Σ denote the resulting theory, which contains the foundational axioms of section (2), 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]:

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

$$Poss(a,s) \supset holds(sbit(b), do(a,s)) \equiv [\neg holds(rbit(b),s) \wedge a = \langle ack(\underline{b}), Dlvrd\rangle \vee holds(sbit(b),s) \wedge \neg(\exists\, b')\ a = \langle ack(b'), Dlvrd\rangle],$$
$$Poss(a,s) \supset holds(rbit(b), do(a,s)) \equiv [holds(sbit(b),s) \wedge ((\exists\, m)\ a = \langle send(m,b), Dlvrd\rangle) \vee holds(rbit(b),s) \wedge \neg((\exists\, m,b)\ a = \langle send(m,b), Dlvrd\rangle)],$$
$$Poss(a,s) \supset holds(sent(m,b), do(a,s)) \equiv a = \langle send(m,b), Dlvrd\rangle \vee holds(sent(m,b),s).$$

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:

$$Poss(\langle send(m,b), Dlvrd\rangle, s) \equiv holds(sbit(b),s),$$
$$Poss(\langle send(m,b), Loss\rangle, s) \equiv holds(sbit(b),s),$$
$$Poss(\langle ack(b), Dlvrd\rangle, s) \equiv holds(rbit(b),s),$$
$$Poss(\langle ack(b), Loss\rangle, s) \equiv holds(rbit(b),s).$$

Possible Outcomes Axioms

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

Probability Assignments

$$Pr(Loss, do_{[}(send(m,b),s)) = K_1,$$
$$Pr(Loss, do_{[}(ack(b),s)) = K_2.$$

Thus, we consider that the probabilities of losing messages or acknowledgements is 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:

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

Remark 5.1 *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 where the bits for the sender and receiver are different, then, after making n attempts to deliver m, the delivery will have been successfull in a later state s if and only if s is such that the bits of the sender and the receiver are the same.*

$$\Sigma \models (\forall\, s, s', m, b, n).$$
$$[s \geq S_0 \wedge \neg sent(m, s) \wedge holds(Ready, s) \wedge s \overset{send^n(m,b)}{\longrightarrow} s'] \supset$$
$$(sent(m, s') \equiv \neg holds(Ready, s')).$$

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 acknowledgement received by the sender from the receiver. After a successful acknowledgement, 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:*

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

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

$$\Sigma \models (\forall\, s).\ s \geq S_0 \wedge holds(Ready, s) \wedge Poss(\langle ack(b), Dlvrd\rangle, s) \supset$$
$$[holds(f, s) \equiv holds(f, do(\langle ack(b), Dlvrd\rangle, s))],$$
$$\Sigma \models (\forall\, s).\ s \geq S_0 \wedge \neg holds(Ready, s) \wedge Poss(\langle ack(b), Dlvrd\rangle, s) \supset$$
$$holds(Ready, do(\langle ack(b), Dlvrd\rangle, s)).$$

Obviously, the the acknowledgement 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 acknowledgement is finally delivered (resulting in a state where Ready holds) after n attempts is given by:*

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

- *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 $K_1^n \times K_2^{n'}$.*

$\square$

6 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 work on the modeling on probabilistic processes (e.g., [2]). 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. Particularly, following the work of Bacchus [1]. 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., [5]), and on the other to approaches that integrate probabilities within other logical frameworks (e.g., [4, 3]).

References

[1] Fahiem Bacchus. **Lp**, A Logic for Representing and Reasoning with Statistical Knowledge. *Computational Intelligence*, 6:209–231, 1990.

[2] Christel Baier and Marta Kwiatkowska. Automatic Verification of Liveness Properties of Randomized Systems (Extended Abstract). Submitted to Information Processing Letters, June 1998.

[3] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proceedings of the Foundations of Software Technology and Theoretical Computer Science*, volume 1026 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[4] H. Hansson and B. Jonsson. A Logic for Reasoning about Time and Probability. *Formal Aspects of Computing*, 6:512–535, 1994.

[5] Fangzhen Lin. Embracing Causality in Specifying the Indeterminate Effects of Actions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-96)*, 1996.

[6] John McCarthy and Patrick J. Hayes. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence* **4**, pages 463–502. Edinburgh University Press, Edinburgh, Scotland, 1969.

[7] Javier Pinto. *Temporal Reasoning in the Situation Calculus.* PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, February 1994. URL = ftp://ftp.cs.toronto.edu/˜cogrob/-jpThesis.ps.Z.

[8] Javier A. Pinto. Causality in Theories of Action. In Rob Miller and Murray Shanahan, editors, *Common-Sense '98*, pages 349–364. Queen Mary and Westfield College, University of London, January 1998. Also found at `URL = http://www.ida.liu.se/ext/etai/nj/fcs-98/349/-paper.ps`.

[9] Raymond Reiter. *The Frame Problem in the Situation Calculus: A Simple Solution (sometimes) and a completeness result for goal regression*, pages

359–380. Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy. Academic Press, San Diego, CA, 1991.