Observations and the Probabilistic Situation Calculus

Paulo Mateus
CLC - Dep. Mathematics
Instituto Superior Técnico
Av. Rovisco Pais
1049-001 Lisboa

António Pacheco
CMA - Dep. Mathematics
Instituto Superior Técnico
Av. Rovisco Pais
1049-001 Lisboa

Javier Pinto *

Abstract

In this article we propose a Probabilistic Situation Calculus logical language to represent and reason with knowledge about dynamical worlds in which actions have uncertain effects. Two essential tasks are addressed when reasoning about change in worlds: Probabilistic Temporal Projection and Probabilistic Belief Update. Uncertain effects are modeled by dividing an action into two subparts: a deterministic input (agent produced) and a probabilistic reaction (nature produced). The probability distributions of the reactions are assumed to be known. Our logical language is an extension to Situation Calculae in the style proposed by Raymond Reiter. There are three aspects to this work. First, we extend the language to accommodate terms dealing with belief and probability. Second, we provide a operational semantics based on Randomly Timed Automata. Finally, we develop Monte-Carlo algorithms to efficiently interpret the probability and belief terms. With the framework proposed we discuss how to develop a reasoning system in Mathematica capable of performing temporal projection and belief update in the Probabilistic Situation Calculus. Finally, we present a sound basis to set rewards and observation planning.

BACKGROUND AND OBJECTIVES

Lately, there has been increasing interest in improving the expressive power of logical languages for representing knowledge about dynamic worlds with non-determinism and uncertainty (e.g., [Pinto et al 2000, Bacchus et al 1999, Poole 1998, Grosskreutz and Lakemeyer 2000, Baral and Tuan 2001]). In this article, we extend previous work on the *Probabilistic Situation Calculus* [Pinto et al 2000, Mateus et al 2001], a logical language for knowledge representation and reasoning about dynamic worlds in which actions have uncertain effects. In this article, we:

- Show how to represent knowledge about actions that have uncertain effects by exploiting the separation between *agent initiated actions* and nature's *random reaction*, first introduced in [Pinto et al 1999].
- Introduce observations as agent actions with uncertain effects. Thus, as in [Bacchus et al 1999] observations are noisy; e.g., we may observe that the outside temperature is 25 degrees, but the reading may have an error with a Gaussian distribution.

As discussed in [Grosskreutz and Lakemeyer 2001], there at least two essential tasks to be addressed when reasoning about change in worlds in which there are uncertain actions and noisy observations: [a] Probabilistic Temporal Projection: Given an initial state, a sequence of uncertain actions (an uncertain plan) and some fluent, determine the probability that the fluent will hold after the actions are executed in the initial state. [b] Probabilistic Belief Update: Given an initial state, a sequence of uncertain actions (an uncertain plan), an observation, and some fluent, determine the probability that the fluent will hold after the uncer-

* Unfortunately Javier Pinto passed away in an accident while this paper was being prepared. Formerly, he was at Bell Labs, Database Systems Research Dept., 600 Mountain Ave., New Jersey 07974, U.S.A.

tain actions are executed in the initial state, and an uncertain (or *noisy*) observation is performed.

An important characteristic of the language of the probabilistic situation calculus is the hybrid nature of its semantics. In particular the logic includes the reals, and its usual operators, without axiomatizing them. Thus, we assume, at the semantic level, a fixed interpretation for them, and the satisfaction relation has to take this into account. When reasoning with such a theory, logical reasoning has to be combined with reasoning about the reals. The latter is done by appealing to an oracle (in our case, we implement this oracle using MATHEMATICA). Furthermore, probability theory is assumed at the semantic level, and is not axiomatized. Our approach is radically different from approaches based on possible world semantics (e.g., [Bacchus et al 1999]), as discussed in the full article.

STATE TRANSITIONS

Our work extends the Situation Calculus in the style presented in [Reiter 1991, Reiter 2000], with the notion of uncertain actions. Recall that the situation calculus is a sorted logic with sorts $\mathcal{S}$ (situations), $\mathcal{A}$ (actions), $\mathcal{F}$ (fluents), etc. An essential component of any situation calculus theory is a description of how the world is affected by actions. This is done by means of effect axioms, which given a situation s, specify how the world would be in a situation $do(a, s)$. Here, a is an action term, while s and $do(a, s)$ are situation terms. The latter term denotes the situation that results from performing action a in situation s. For instance, if the action is $jump(x)$, meaning jump forward x meters, then one effect axiom could be:

$$holds(at(y-x), s) \supset holds(at(y), do(jump(x), s)). \tag{1}$$

At the heart of our proposal to integrating actions with uncertain outcomes, is the introduction of the sorts $\mathcal{I}$ (inputs) and $\mathbb{R}$ (reals), along with the requirement that any action corresponds to a pair $\langle i, r\rangle$ (an input and a real). The second component is referred to as a random *reaction*. For instance, we can introduce uncertainty in the previous example by having $jump_i(x)$ as an input, and the reaction being the actual distance jumped. Thus, the pair $\langle jump_i(3), 3.4\rangle$ would denote the action of jumping forward 3 meters but actually jumping 3.4 meters. The effect axiom (1) would now look like:

$$holds(at(y-r), s) \supset holds(at(y), do(\langle jump_i(x), r\rangle, s)). \tag{2}$$

As a second example, we model a dice throw as six different actions: one input (*throw*) and six possible reactions (the reals $1, 2, 3, 4, 5$ and 6). Pictorially: Thus,

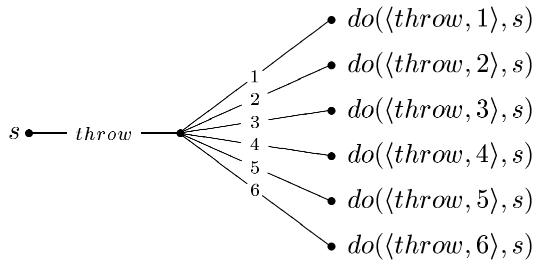

Figure 1: Throwing a dice, and possible reactions

an agent's input (e.g., *throw*), has reactions that are indeterminate. Probability distributions are placed over the reactions. In the dice example, the distribution would be discrete, and in the *jump* example the distribution would be continuous. However, in order to have a uniform theory, without having to explicitly distinguish between continuous and discrete cases, we approximate discrete distributions by continuous ones that are described using the *Dirac Delta* function. Recall that the Dirac Delta of a real x is a function that is zero over $\mathbb{R}$ except in an infinitesimal neighborhood around x, whose indefinite integral is 1. Thus, the probability density function in the case of the dice example, would be

$$dice(x) = \sum_{i=1}^{6} 1/6 \ \mathrm{DiracDelta}(i)(x).$$

This approach is completely general and allows for mixed distributions; i.e., inputs with reactions that have distributions that are continuous in some areas and discrete in others. Given that we have divided actions into its *input* and *reaction* components, we have decomposed the *Poss* relation[1] into *iposs* and *rposs*. Thus, we write:

$$Poss(\langle i, r\rangle, s) \equiv iposs(i, s) \wedge rposs(r, i, s). \tag{3}$$

iposs is similar to the standard *Poss* predicate. $rposs(r, i, s)$ is true when reaction r is possible after input i is performed in situation s. In the dice example,

$$rposs(r, throw, s) \equiv dice(r) > 0.$$

As discussed in the sequel, we actually use the *cummulative distribution function (cdf)* instead of the density

[1] In the situation calculus, $Poss(a, s)$ is a relation that holds when action a can be executed in situation s.

function. In general, we write $cdf(i, s)$ to denote the cdf of the reaction to input i in situation s. Thus $cdf(throw, s)(x)$ has value zero for values of x smaller than 1, value 1 for x greater or equal than 6, value 1/6 when $1 \leq x < 2$, value 2/6 when $2 \leq x < 3$, etc.

Notice that we are assuming the structure of the reals ($\mathbb{R}$) along with a very rich arsenal of real operators. We do not attempt to axiomatize $\mathbb{R}$, instead we assume standard interpretations for $\mathbb{R}$ and for the real operators. Furthermore, for reasoning with the resulting theories, we assume the existence of an *oracle* for computations involving the reals. In particular, as shown in examples later, we take MATHEMATICA to be an implementation of this oracle. In essence, our approach mixes *logical reasoning* with *analytical calculus.*

Getting back to the dice example, if one wanted to compute the probability that some fluent hold after input i is performed in a situation s, we would have to compute:

$$prob(f, i, s) = \int_{-\infty}^{+\infty} cf(holds(f, do(\langle i, x\rangle, s)))cdf(i, s)(dx), \tag{4}$$

where the integral is of Lebesgue-Stieltjes type [Resnick 1999] and cf stands for characteristic function (1 if its argument is true, and zero otherwise).

TREE STRUCTURE

In the previous section, we concentrated on describing the approach to model uncertain actions, which derives from [Pinto et al 2000] (where only discrete distributions were considered), and extensions, presented in [Mateus et al 2001], to deal with continuous distributions. We have basically addressed the problem of describing the uncertainty that arises after inputs are performed in a known situation. Notice that a single input, in a fixed situation, gives rise to a potentially infinite set of possible successor situations. For instance, in formula (2), the distribution for the reaction r to a $jump(x)$ input can be a normal[2] $\mathtt{N}(0, 1)$; in which case the possible situations after a $jump(x)$ input would be infinite.

Let us consider now what happens when several inputs are performed in sequence[3]. For instance, when a dice is thrown several times in succession. Naturally we would be interested in computing the probability that some fluent holds after such a sequence is performed. In order to be able to pose such a query, we introduce input sequences as elements of discourse via a new sort $\mathcal{I}^*$ (a generic variable of type $\mathcal{I}^*$ will be $\vec{i}$). For instance, suppose that we define the fluent *Close* as being at a distance of less than 1 from the origin:

$$holds(Close, s) \equiv holds(at(y), s) \wedge -1 < y < 1,$$

Then, we can ask what is the probability that *Close* is true after some sequence of inputs (e.g., $\{jump_i(0.3), jump_i(0.3), jump_i(0.3)\}$) is performed.

A sequence of inputs and a situation define a *Markov Process.* Thus, we assume that the distribution of the outcomes resulting from an input depends only on the situation on which the input is performed. A situation s and an input sequence $\vec{i}$ define a subtree of the original situation structure. We can say that a situation s' is *mp-accessible* from a situation s and an input sequence $\vec{i}$, when s' is reached at the end of some evolution of the Markov process induced by performing the sequence of inputs $\vec{i}$ in the situation s. For instance, the situation $do(\langle jump_i(0.3), 0.25\rangle, do(\langle jump_i(0.3), 0.4\rangle, S_0))$ is *mp-accessible* from situation S_0, with the input sequence $\{jump_i(0.3),\ jump_i(0.3)\}$. The Markovian assumption lets us generalize (4). Thus we introduce an inductive definition of *prob* on the input sequence $\vec{i}$.:

$$\begin{aligned}
&\vec{i} = \epsilon \supset prob(f, \vec{i}, s) = cf(holds(f, s)),\\
&(\exists\, i_0, \vec{i}_+).\vec{i} = (i_0; \vec{i}_+) \supset\\
&\quad prob(f, \vec{i}, s) =\\
&\qquad \int_{-\infty}^{+\infty} prob(f, \vec{i}_+, do(\langle i_0, x\rangle, s))cdf(i_0, s)(dx).
\end{aligned} \tag{5}$$

Here x represents all the outcomes that can arise from performing i_0 in situation s. For each such outcome, we then consider all the outcomes that result from performing the remaining part of the input sequence.

THE INITIAL SITUATION(S)

In the same spirit as the situation calculus, the probabilistic situation calculus allows one to write theories to specify hypothetical developments from the initial situation S_0. Temporal projection provides answers to questions of the form: *Does fluent f hold after a specific sequence of actions is performed in S_0.* On the other hand, *probabilistic* temporal projection, in the framework of the probabilistic situation calculus, seeks to answer questions of the following forms:

[2] We use $\mathtt{N}(\mu, \sigma)$ to denote the *cdf* of the Normal distribution with the usual parameters. Also, later, we use $U(\cdot, \cdot)$ to denote the cdf of the uniform distribution.

[3] In previous work, this problem was only addressed in the semantical and computational aspects

- *What is the probability that some fluent f holds after a sequence of* inputs *is performed starting in S_0?*. This question is only meaningful if this probability is discrete.

- *What is the probability that a fluent $f(x)$, holds after a sequence of* inputs *is performed starting in S_0, and for some value of x in the range x_0 to x_1?*. We have implicitly assumed that the argument x is of type $\mathbb{R}$.

In the context of probabilistic temporal projection, we should consider cases in which an axiomatizer might want to specify a set of situations as probable starting situations. In contrast, a feature of the situation calculus is that the world is assumed to have a single initial situation, called S_0. This means that the agent starts in a completely defined world with no uncertainty about what is true or not. In general, however, agents may not have complete knowledge about the starting state. Instead, an agent may have a family of situations as possible starting situations. There are two natural ways in which one can avoid this initial determinism of the situation calculus. First, one can consider an explicit family of initial situations (as in [Bacchus et al 1999]) and, if one were so inclined, a probability distribution could be defined over these situations. Second, one could define a *phantom* starting situation, and a starting input that generates all the situations that are possible. Hence, the initial situation S_0 would not represent any actual situation. Rather, it represents an initial time in which a phantom *big bang* generates the situations that can be regarded as representing the initial state of the world (as modeled by the theory).

Recall that in the situation calculus a special relation $Poss \subseteq Actions \times Situations$ is used to estabish conditions under which actions can or cannot be executed. Since now we have split actions into subparts, we also separated $Poss$. We use $iPoss \subseteq Input \times Situations$ to express the conditions necessary for inputs to be possible, and $rposs \subseteq Input \times Reals \times Situations$ to express the conditions under which specific reactions are possible after an input is performed in a situation. Thus, to model the big bang, we will need:

$$iPoss(i, s) \supset [i = BigBang \equiv s = S0]$$

Now, as an example, imagine defining some object's position with respect to the X axis. We can have the following:

$$Poss(\langle BigBang, x\rangle, s) \supset at(x, do(\langle BigBang, x\rangle, s)),$$
$$cdf(BigBang, S_0) = \mathtt{N}(\mu_x, \sigma_x).$$

If we wanted to introduce a more complex initial scenario, with more properties and different distributions, the *BigBang* would have to be an input with multivariate reactions (as in a later example).

OBSERVATIONS AS CONDITIONS

In this section we define formally the *probabilistic temporal projection with observations*, which one might also call *Belief Updates*. The problem can be seen as solving probabilistic temporal projection (as informally defined in the previous section) but conditioned on one or more observations along the input sequence. To ease the presentation and for lack of space, we only consider a simplified case in which there is a single observation performed as a last input in the input sequence.

Observations are introduced into the logical language as a subsort $\mathcal{O}$ of $\mathcal{I}$. Thus, $\mathcal{O} \subseteq \mathcal{I}$, for convenience, we also introduce a sort predicate $observ = \mathcal{O}$. Notice that observations, together with their outcomes, are regular actions (inputs followed by reactions). For technical reasons (that we explain later), we introduce a fluent $Obs : \mathbb{R} \to \mathcal{F}$, such that if o is an observation (i.e., $o \in \mathcal{O}$):

$$holds(Obs(r), do(\langle o, r\rangle, s))$$

That is, Obs registers the outcome of the last observation performed. Furthermore, if $i \in \mathcal{I} \backslash \mathcal{O}$, then the only effect of observations is described with:

$$\neg observ(i) \supset holds(Obs(0), do(\langle i, r\rangle, s)).$$

Thus, after an input that is not an observation, the observed value is an arbitrary constant (e.g., 0). Notice that these axioms are domain independent.

Assuming that changes in the world are described using effect axioms, and the frame problem solved by using explanation closure axioms, then the resulting theory will lead to the conclusion that observations don't affect any fluents other than Obs. Indeed, the result of the completion should be that:

$$f \neq Obs(x) \supset holds(f, do(\langle o, x\rangle, s)) \equiv holds(f, s)$$

The axiom states that the state of the world does not change as result of performing observations. Note that all variables are implicitly universally quantified.

Example 1 Let us consider the simplest possible example with observations. Assume that there is a coin inside a box, and an unreliable witness tells us whether it is heads or tails. We believe this witness with probability 0.7. We might want to know what is the probability that the coin is actually tails when the witness says so. To set things up, we have a theory with a single fluent *Tails* which holds only if *Tails* is up[4].

$$
\begin{aligned}
&Poss(\langle BigBang, r\rangle, s) \supset [holds(Tails, s) \equiv r = 1] \\
&cdf(BigBang, s)(x) = Bernoulli(0.5)(x), \\
&cdf(ObsTails, s)(x) = \\
&\qquad Bernoulli(0.7cf(holds(Tails, s)) + \\
&\qquad\qquad 0.3(1\text{-}cf(holds(Tails, s))))(x).
\end{aligned}
$$

By using (5) one can verify that the probability that *Tails* holds after the unitary input sequence $\{BigBang\}$ is the same as the probability that *Tails* holds after the input sequence $\{BigBang, ObsTails\}$. However, as we see later, conditioning on the observation (i.e., computing the probabiity that *Tails* holds after $\{BigBang, ObsTails\}$ and seeing the outcome of $ObsTails$), allows us to do *Bayesian belief updates.*

Notice that observations are handled in a radically different manner than in proposals based on epistemic logics (e.g., [Bacchus et al 1999]). Basically, an observation action simply changes the nature of the distribution over the space of situations reached after performing actions. Thus, the distribution over the space of situations possible after input sequence $\vec{i}$ is changed after an observation o is made.

As in the previous example, the rest of the article considers only univariate outcomes for all the inputs. However, it is not difficult to extend this approach to multivariate cases (extending the presentation in [Mateus et al 2001] with observations).

Now, we extend the vocabulary of the probabilistic situation calculus to deal with updates based on observations (we deal with semantic issues in a later section). We introduce the operator $bel : \mathcal{F} \times \mathcal{I}^* \times \mathcal{S} \times \mathcal{O} \times \mathbb{R} \rightarrow \mathcal{P}$, with the following syntactic sugar: The term $bel(f, \vec{i}, s|o = r)$, denotes the probability that f holds, after input $\vec{i}$ is performed starting in s, and observing (at the end) reaction r to random observation o. We need to add an axiom to define[5]:

$$bel(f, \vec{i}, s|o = r) = \frac{prob(f \wedge Obs(r), (\vec{i}; o), s)}{prob(Obs(r), (\vec{i}; o), s)} \qquad (6)$$

[4]Here, Bernoulli represents the following cumulative distribution function:

$$Bernoulli(p)(y) = \begin{cases} 0 & \text{whenever } y < -1 \\ 1 - p & \text{whenever } -1 \leq y < 1 \\ 1 & \text{otherwise} \end{cases} .$$

[5]This axiom is based on the definition of conditional probability $P(A|B) = P(A \wedge B)/P(B)$.

In this expression we have used $f \wedge Obs(r)$ as syntactic sugar for $and(f, Obs(r))$; where $and : \mathcal{F} \times \mathcal{F} \rightarrow \mathcal{F}$ is a fluent operator. Here, if f_1, f_2 are fluent, $and(f_1, f_2)$ is a *defined* fluent (see [Pinto 1994]).

Note that to evaluate *bel* in (6), we need to evaluate two *prob* terms using (5). The expression (5) is a recursive definition for *prob*. By simple inspection, it turns out that to evaluate an expression like $prob(f, \vec{i}, s)$, where the input sequence $\vec{i}$ has dimension n requires solving an n-dimensional indefinite integral. Except for trivial cases, this integration is impossible to solve in exact terms. Before we discuss how we address this difficulty, we will discuss the semantics underlying the probabilistic situation calculus with observations.

EXTENDED PROBABILISTIC SITUATION CALCULUS SEMANTICS

In this section, we introduce the semantics for the probabilistic extension to the situation calculus. We specify how the situation structure is generated and how the probability distributions are laid on top of this structure. The presentation of the semantics for the probabilistic situation calculus provided here is an extension to the Randomly Reactive Automata presented in [Mateus et al 2001]. In this extended abstract, we do not present the full semantics and leave standard aspects of Tarskian semantics out.

We provide a probabilistically sound specification for the computation of beliefs and probabilities based on the Randomly Reactive Automata. This should be understood as an *operational semantics*, in the sense that it specifies a mechanism to evaluate the prior and posterior probabilities (of fluents being true before or after certain observations). Notice that any system that reasons about these probabilities (like our oracle MATHEMATICA) has to obey this semantic specification.

As described formally below, in a Randomly Reactive Automaton with observations, actions are represented as pairs from $\mathcal{I}\mathbb{R}$, where $\mathcal{I}$ is the set of inputs (part of the specification of the Randomly Reactive Automaton), and $\mathbb{R}$ are the real numbers. These pairs correspond exactly to the input-reaction pairs of the last section. The situations are not explicitly introduced because they are simply sequences of $\mathcal{I}\mathbb{R}$ elements. Thus, we can correctly define the universe of situations as $S = (\mathcal{I}\mathbb{R})^*$. The empty string corresponds to the initial situation. We formally define *Randomly*

Reactive Automata with observations as a tuple

$$\mathcal{A} = \langle I, O, F, T, M, g, h \rangle$$

where:

- I is a set of *inputs*. $O \subseteq I$ is a set of *observations*. F is a set of (proper) *fluents* (as specified in the full semantics, this set excludes the special fluent *Obs*).
- $T = \{T^f\}_{f \in F}$ where $T^f \subseteq S$ is such that

  $$T^f_{\vec{i}\,s} = \{\vec{x} \in \mathbb{R}^n : s i_1 x_1 \ldots i_n x_n \in T^f\}$$

  is Borel-measurable for all $\vec{i} = \langle i_1 \ldots i_n \rangle \in I^*$ and $s \in S$. For each fluent $f \in F$, T^f represents the set of situations in which f holds.
- $M = \{M^{[s]}\}_{[s] \in [S]}$ where $M^{[s]} \subseteq I$ is the set of *possible inputs* at $[s]$, and $[S] = S/\approx$, $[s] = \{s' \in S : s \approx s'\}$ and $\approx$ is an equivalence relation of *indistinguishability* over S such that $s_1 \approx s_2$ iff $s_1 \in T^f \equiv s_2 \in T^f$ for all $f \in F$.
- $g = \{g_i^{[s]}\}_{i \in I \setminus O, [s] \in [S]}$, where $g_i^{[s]}$ is a generalized density function [Mateus et al 2001] of the *(hidden) reaction after performing input i at* $[s]$;
- $h = \{h_o^{[s]}\}_{o \in O, [s] \in [S]}$, where $h_o^{[s]}$ is a generalized density function of the *(observed) reaction after performing observation o at* $[s]$;

In order to completely specify the structures for probabilistic situation calculus interpretations, we need to add sets of objects for all domain sorts, along with sets of symbols for domain predicates and functions. Also, we need to specify the interpretation for the *prob* and *bel* predicates, which are not fully axiomatized and play a special role in the theory. This latter part is completed in the next section. Armed with these interpretation structures it is relatively straightforward to provide a Tarskian style specification for the satisfaction relation between a Probabilistic Situation Calculus interpretation structure and a Probabilistic Situation Calculus formula. Moreover, as expected, first order reasoning is sound with respect to the proposed semantics. Both the Tarskian specification for the satisfaction and the soundness results are deferred to the full paper.

BELIEF

In this section we introduce the interpretation for the *prob* and *bel* predicates in the light of randomly timed automata. In general, such interpretation requires solving an arbitrary n-dimensional indefinite integral, which is unfeasible via classical analytic calculus. To overcome this hurdle, we present a Monte-Carlo based method for approximating such integrals and discuss its efficiency for the applications in mind.

Note that, if no observation is made, the *belief* that a fluent f holds after executing a sequence of actions $\vec{i}$ at a situation s is just given by the probability $prob(f, \vec{i}, s)$ as defined in (5). Given a Randomly Reactive Automata $\mathbf{A}$ we fix the value of $prob(f, \vec{i}, s)$ by calculating the probability of reaching a situation in T^f by a sequence of inputs denoted by $\vec{i} = i_1 \ldots i_n$ from the situation denoted by s, more precisely:

$$[\![prob(f, \vec{i}, s)]\!]_{\mathbf{A}} = \int_{\mathbb{R}^n} 1_{T^f_{\vec{i}\,s}}(\vec{x}) \prod_{k=0}^{n-1} g_{i_{k+1}}^{[s\vec{i}_{k]}\vec{x}_{k]}]}(x_{k+1}) dx_n \ldots dx_1 \qquad (7)$$

where $1_{T^f_{\vec{i}\,s}}$ is the characteristic function of $T^f_{\vec{i}\,s}$ and $s\vec{i}_{k]}\vec{x}_{k]} = s_1 i_1 x_1 \ldots i_k x_k$. This interpretation is sound with respect to the axiomatization presented in (5).

Unfortunately, solving this kind of integrals with all generality is an herculean task, and therefore methods of approximation are required. Literature, such as [Evans and Swartz 2000], clearly indicates that Monte-Carlo based methods are the most suitable for dealing with such probabilities. The overall idea is to sample over m independent random situations reached by $\vec{i}$ from s and take the rate of those that reach T^f to approximate the probability. In practice, this sampling can be easily obtained via a pseudo-random generator, which makes easy to encode this function in any language with this utility, like for instance, MATHEMATICA.

Proposition 1 The value $[\![\mathrm{prob}(f, \vec{i}, s)]\!]_{\mathbf{A}}$ can be estimated via a Monte-Carlo approach by

$$\widehat{\mathrm{prob}}_m(f, \vec{i}, s) = \frac{\sum_{k=1}^{m} 1_{T^f_{\vec{i}\,s}}(\vec{x}_k)}{m}$$

where $\{\vec{x}_k\}_{k=1,\ldots,m}$ are m vectores of reactions obtained by a Monte-Carlo independent sampling of reactions obtained after executing input $\vec{i}$ from s. Moreover $\widehat{\mathrm{prob}}_m(f, \vec{i}, s)$ is a consistent estimator of $\mathrm{prob}(f, \vec{i}, s)$; that is, it converges with probability 1 to $[\![\mathrm{prob}(f, \vec{i}, s)]\!]_{\mathbf{A}}$.

An advantage of the proposed Monte-Carlo algorithm is that, fixed s and $\vec{i}$, it is polynomial in the size of the sampling m. Therefore, if we want a more precise approximation, it is feasible just to consider more samples. An important addon result is that the (random) estimation error of this approximation has an asymptotic Gaussian distribution with mean 0 and standard deviation given by:

$$\sqrt{\frac{p(1-p)}{m}}$$

where $p = [\![\mathrm{prob}(f, \vec{i}, s)]\!]_{\mathbf{A}}$.

A more precise notion of whether a fluent f holds or not might be achieved if an observation o is made after executing $\vec{i}$ from s. Naturally, we only gain any information with the observation if the latter is not *independent* from the fluent holding. If the outcome of observing o is independent from fluent f holding the following bad thing happens

$$bel(f, \vec{i}, s|\ o = r_1) = prob(f, \vec{i}, s),$$

meaning that observing o is totally useless to determine f. With our framework it is possible to measure the information gained by performing an observation o in order to know if a fluent f holds or not. This problem is of utmost relevance when observations are expensive and a plan of observations has to be set up. The full treatment of this problem is deferred to an extended version of this paper.

Independently of the value that an observation might provide to determine whether a fluent is holding or not, it is always possible to define in rigorous terms the *belief* of a fluent holding given an observation. Indeed, as presented for the discrete case in (6), for interpreting $bel(f, \vec{i}, s|o = r_1)$ we have to compute the conditional probability of reaching T^f by $\vec{i}$ from s given that observing o resulted in the value r_1. Hence, for a Randomly Reactive Automata $\mathbf{A}$ we have $[\![bel(f, \vec{i}, s|\ o = r_1)]\!]_{\mathbf{A}} = \lim_{y \to +\infty}$

$$\frac{\int_{(-y,y)^k} 1_{T^f_{\vec{i}\,s}}(\vec{x}) h_o^{[s\vec{i}_n]\vec{x}_n]}(r_1) \prod_{k=0}^{n-1} g_{i_{k+1}}^{[s\vec{i}_k]\vec{x}_k]}(x_{k+1}) dx_n..dx_1}{\int_{(-y,y)^k} h_o^{[s\vec{i}_n]\vec{x}_n]}(r_1) \prod_{k=0}^{n-1} g_{i_{k+1}}^{[s\vec{i}_k]\vec{x}_k]}(x_{k+1}) dx_n..dx_1}$$

Once again, solving the integrals presented above is an unfeasible task. However, contrasting with *prob*, approximation of conditional probabilities through Monte-Carlo methods is not straightforward, as we shall see.

Proposition 2 The value $[\![bel(f, \vec{i}, s|\ o = r_1)]\!]_{\mathbf{A}}$ can be estimated via a Monte-Carlo method by

$$\widehat{\mathrm{bel}}_m(f, \vec{i}, s|\ o = r_1) = \frac{\sum_{k=1}^m 1_{T^f_{\vec{i}\,s}}(\vec{x}_k) h_o^{[s\vec{i}\vec{x}_k]}(r_1)}{\sum_{k=1}^m h_o^{[s\vec{i}\vec{x}_k]}(r_1)}$$

where $\{\vec{x}_k\}_{k=1,\ldots,m}$ are the first m vectors of reactions obtained by a Monte-Carlo independent sampling verifying $h_o^{[s\vec{i}\vec{x}_k]}(r_1) > 0$. Moreover $\widehat{\mathrm{bel}}_m(f, \vec{i}, s|\ o = r_1)$ is a consistent estimator of $bel(f, \vec{i}, s|\ o = r_1)$.

The need to guarantee m samples where there is positive probability density of obtaining r_1 from observing o is a shortcoming of Proposition 2. Observe that, if the reaction value r_1 has no probability (density) of being observed and it is, something very bad has happened. In particular, it is a clear evidence that one has misspecified the stochastic behavior of the agent. In this latter case it is impossible to obtain the required m samples, since observing the value r_1 is *inconsistent* with the specification, and therefore no *belief update* can be achieved. Putting aside such pathological behaviours, it still might be very hard to find situations where there is positive probability density of obtaining r_1 from observing o. In detail, for a starting situation s and sequence of inputs $\vec{i}$, the probability of reaching such situations is

$$p(\vec{i}, s, o, r_1) = \int_{\mathbb{R}^n} 1_{\{\vec{x}: h_o^{[s\vec{i}\vec{x}]}(r_1) > 0\}}(\vec{x}) \prod_{k=0}^{n-1} g_{i_{k+1}}^{[s\vec{i}_k]\vec{x}_k]}(x_{k+1}) dx_n \ldots dx_1.$$

Note, comparing with (7), that this value can be approximated via Monte-Carlo in the same way as *prob*. So, given s, $\vec{i}$, o and r_1, the next question arises:

How big must a sample be in order to obtain m vectors in $\{\vec{x} : h_o^{[s\vec{i}\vec{x}]}(r_1) > 0\}$?

It is intuitive that the expected value for the number of samples is $m/p(\vec{i}, s, o, r_1)$, further introspection leads to the following result.

Proposition 3 Let $M_m(\vec{i}, s, o, r_1)$ be the random variable defined as

$$M_m(\vec{i}, s, o, r_1) = \inf\left\{k\ :\ \sum_{j=1}^{k} 1_{\{\vec{x}: h_o^{[s\vec{i}\vec{x}]}(r_1) > 0\}}(\vec{X}_j) \geq m\right\}$$

where $\vec{X}_j = (X_{1j}, X_{2j}, \ldots, X_{|\vec{i}|j})$ with $\vec{X}_1, \vec{X}_2, \ldots$ being independent sequences of reactions obtained after executing input $\vec{i}$ from s, and $|\vec{i}|$ denoting the cardinality of $\vec{i}$. Then $M_m \sim$ Negative Binomial $(m, p(\vec{i}, s, o, r_1))$; that is:

$$P(M_m(\vec{i}, s, o, r_1) = k) = \binom{k-1}{m-1} (1 - p(\vec{i}, s, o, r_1))^{k-m} p(\vec{i}, s, o, r_1)^m$$

for $k = m, m+1, \ldots$.

Thus, the random variable $M_m(\vec{i}, s, o, r_1)$ has expected value $m/p(\vec{i}, s, o, r_1)$ and variance $m(1 - p(\vec{i}, s, o, r_1))/p(\vec{i}, s, o, r_1)^2$. For large m, $M_m(\vec{i}, s, o, r_1)$ can be approximated by a normal distribution via

$$\frac{M_m(\vec{i}, s, o, r_1) - \frac{m}{p(\vec{i}, s, o, r_1)}}{\sqrt{\frac{m(1 - p(\vec{i}, s, o, r_1))}{p(\vec{i}, s, o, r_1)^2}}} \simeq \mathrm{N}\,(0, 1)$$

where $\simeq$ stands for *approximate in distribution.*

Rewards can be introduced by considering a real valued (measurable) function from the set of indistinguishable situations. The reward of executing a plan of actions is obtained by approximating (via Monte-Carlo) the integral of such function over the probability measure associated to the plan. The information given by an observation for checking whether a fluent holds or not (value relevant for observation planning) can be approximated by this method, since the entropy of a sequence of actions given an observation can be obtained as a reward.

THE FROGBOT EXAMPLE

This is a simple example in which a frogbot can hop to the left or to the right, following a random exponentional law (along a single dimension, starting at position 0). The frogbot is also equipped with a (faulty) observation device that indicates the position of the robot with an uniform random error.

SPECIFICATION

To model this example we use the sort $\mathcal{R}$ for real numbers, and introduce two constants *MoveLeft* and *MoveRight* of type $\mathcal{I}$, the constant *Find* of type $\mathcal{O}$, the function $pos : \mathcal{R} \rightarrow \mathcal{F}$, and the constants *Center* and *Hawk* of type $\mathcal{F}$. The axiomatization follows:

Axioms about the initial situation

$$holds(pos(0), S_0). \tag{8}$$

$$holds(obs(0), S_0). \tag{9}$$

Hence, we assume that the frogbot starts at position 0 and starts by observing 0.

CDF axioms:

$$iposs(MoveLeft, s) \wedge s \geq S_0 \supset cdf(MoveLeft, s) = Exponential(3); \tag{10}$$

$$iposs(MoveRight, s) \wedge s \geq S_0 \supset cdf(MoveRight, s) = Exponential(2); \tag{11}$$

$$iposs(Find, s) \wedge s \geq S_0 \supset cdf(Find, s) = Uniform(pos(s)). \tag{12}$$

Here, we refer to the uniform distribution over the interval $[pos(s) - 0.5, pos(s) + 0.5]$.

Action precondition axioms

Action precondition axioms, which, in Reiter's style of axiomatization, are expressed in terms of conditions for *Poss*, are now expressed in terms of *iposs*. I.e., we specify constraints for the possibility of giving some input. Thus, for this example, we write:

$$iposs(MoveLeft, s) \equiv holds(pos(x), s) \wedge x \geq 0; \tag{13}$$

$$iposs(MoveRight, s) \equiv holds(pos(x), s) \wedge x \geq 0. \tag{14}$$

$$iposs(Find, s) \equiv holds(pos(x), s) \wedge x \geq 0. \tag{15}$$

Effect axioms

$$[legal(do(\langle MoveLeft, d\rangle, s)) \wedge holds(pos(x), s) \wedge n > 0] \supset holds(pos(x - d), do(\langle MoveLeft, d\rangle, s)) \tag{16}$$

$$[legal(do(\langle MoveRight, d\rangle, s)) \wedge holds(pos(x), s) \supset holds(pos(x + d), do(\langle MoveRight, d\rangle, s). \tag{17}$$

Ramification State Constraints

The definitions of *central situations* and *hawk situations* are given as an equivalence. Also, constraint (20) states that pos is a functional fluent.

$$holds(Center, s) \equiv holds(pos(x), s) \wedge -1 < x < 1 \tag{18}$$

$$holds(Hawk, s) \equiv holds(pos(x), s) \wedge 0 > x \tag{19}$$

$$holds(pos(x), s) \wedge holds(pos(y), s) \supset x = y. \tag{20}$$

Based on the approach presented in [Pinto 1999], the effect axioms and the ramification state constraints for the fluent *pos* are replaced by *successor state axioms*. These axioms are derived, by syntactic transformations, from the effect axioms and ramification constraints. The following axioms are obtained after some simplifications:

Successor State Axioms:

$$\begin{array}{l} legal(do(a,s)) \supset [holds(pos(x), do(a,s)) \equiv \\ \quad holds(pos(x+d), s) \wedge a = \langle MoveLeft, d\rangle \vee \\ \quad holds(pos(x-d), s) \wedge a = \langle MoveRight, d\rangle \vee \\ \quad holds(pos(x), s) \wedge a = Find]. \end{array} \tag{21}$$

Therefore, the frogbot theory is composed by the PSC axioms, along with axioms (8)-(15), (18)-(19) and (21).

Next, we show how to encode the PSC axioms in MATHEMATICA.

REASONING

In this subsection we present an approach for the computational realization of a system that reasons about actions based on theories written in the Probabilistic Situation Calculus. The approach is based on the encoding of the logical theories into rewrite rules of Mathematica [Wolfram 1996]. We consider the frogbot example to illustrate the approach.

Mathematica Encoding of Domain Independent Definitions

In the example, we will use the *uniform* and the *exponential* probability distributions. The uniform distribution has density over all points that distance one or less from a given point s. To specify the distributions, we explicitly encode their *cummulative distribution functions* (`cdf`) and their inverses (`idf`). In the examples, we'll use the following:

```
cdfuniform[s_] :=
Function[x,If[x≤s-1,0,
     If[x≥s+1,1,1/2(x-s+1)]]];
idfuniform[s_] := Function[x,s-1+2x];
cdfexp[m_] := Function[x, UnitStep[x]
(1 -  E^(-x/m))];
idfexp[m_] := Function[z, m
Log[1/(1-z)]];
```

Note that, in order to uniformly handle discrete and continuous distributions, one can model discrete distributions with real functions by making use of the *Dirac delta*. Recall that the Dirac delta is a function that is zero over $\mathbb{R}$ except in an infinitesimal neighborhood around 0, whose indefinite integral is 1.

The Monte–Carlo simulation for the univariate case is fairly straightforward. Given a fluent, and an input sequence, we want to determine the probability that the fluent will hold after executing the input sequence with the corresponding reactions. Since the reactions are random, the simulation generates the random reaction from the corresponding inverse cumulative probability distribution, this is simply done as:

```
gen[i_,s_] := idf[i,s][Random[]];
```

Thus, given an input and a situation, the `gen` function gives a random reaction. Notice that the probability distribution of the reaction might depend on the situation where the input is performed. The `idf` function is domain dependent, hence is specified in the domain axiomatization below.

A Monte–Carlo run is defined as a Bernoulli experiment consisting of taking an input sequence and simulating the execution of the ordered sequence of inputs, each followed by its random reaction. We obtain the situation reached in an Monte–Carlo sampling recursively as follows:

```
gensit[{}, s_] := s;
gensit[w_, s_] := gensit[Rest[w],
     do[{First[w],gen[First[w], s]}, s]];
```

The experiment is successful if the fluent of interest (parameter of the experiment) holds over the reached situation.

```
mcrun[f_, w_, s_] := holds[f, gensit[w, s]];
```

As stated in Proposition 1, in order to estimate the probability that fluent `f` will hold after an input sequence `w`, we use a frequency count. Thus, we run the Monte–Carlo experiment `n` times, and the estimate of the probability that the outcome will be such that `f` holds is the proportion of times that the experiment succeeds. Hence, we define the `prob` function as follows:

```
prob[f_,w_,s_,n_] := N[Count[
    Table[mcrun[f,w,s],{jrun,1,n}],True]/n];
```

Belief is modeled similarly, and hence, the ratio obtained in Proposition 2 can be calculated as follows:

```
ratio[l_,f_,o_,r_]:=
 Apply[Plus,
```

```
  Map[
   Function[s,If[holds[f,s],cdf[o,s]'[r],0]],
  l]]/
 Apply[Plus,
  Map[
   Function[s, cdf[o, s]'[r]],
  l]];

bel[f_,w_,s_,o_,r_,n_] := N[ratio[
     Table[gensit[w,s],{jrun,1,n}],f,o,r]];
```

The domain independent situation calculus axioms required are direct translations of logical axioms introduced before. These are:

```
poss[{i_,r_},s_] := iposs[i,s]∧
cdf[i,s]'[r]>0;
legal[s0] := True;
legal[do[{i_,r_},s_]] :=
legal[s]∧poss[{i,r},s];
holds[f_,do[a_,s_]] :=
holds[f,s]/;¬legal[do[a,s]];
```

Here the string `/;` should be read as `if`.

Frogbot

Now we translate the frogbot example to illustrate the reasoning capabilities of the oracle in a simple domain in which the probabilities have a continuous distribution. Recall that the frogbot can produce two possible proper inputs, `moveleft` and `moveright`, and the observation `find`. When an input is performed, the random reaction obtained is the distance by which it moves. The random reactions are modeled in our oracle as follows:

```
cdf[moveleft,s_] := cdfexp[3] ;
idf[moveleft,s_] := idfexp[3] ;
cdf[moveright,s_] := cdfexp[2] ;
idf[moveright,s_] := idfexp[2] ;
cdf[find, s_] := cdfuniform[pos[s]];
idf[find, s_] := idfuniform[pos[s]];
```

Notice that, we specify the cumulative distribution function and its inverse. The latter is necessary for performing the simulation in the Monte–Carlo approach to estimate probabilities of fluents holding.

We will make use of the following auxiliary definition:

```
pos[s_]:=iota[x,holds[position[x],s]];
obs[s_]:=iota[x,holds[observation[x],s]];
```

this means that `pos` is a functional fluent, whose value corresponds to the position of the frogbot in a given situation. It is assumed that all observations are functional, and so a similar auxiliar definition can be applied to `obs`. As explained before, the function `iota` gives, in this case, the value of `x` such that `x` is the position of the frogbot in the specified situation. For these definitions, we appeal to the `iota` function. In the `pos` case, `iota` gives the value of `x` such that `holds[position[x],s]` is true. This value has to be unique, in the implementation it gives the first solution it encounters, if any. The Mathematica definition is:

```
iota := Function[z,c,Solve[c,z][[1,1,2]]];
```

Next, we specify the preconditions for the inputs and the initial situation

```
iposs[i_,s_]:=(i===moveright ∨
   i===moveleft ∨ i===find) ∧ pos[s]≥0;
holds[position[x_],s0]:=(x==0);
holds[observations[x_],s0]:=(x==0);
```

The successor state axioms for the position fluent in this domain is as follows:

```
holds[position[x_],do[{i_,y_},s_]] :=
  ((i===moveleft ∧ holds[position[x+y],s]) ∨
  (i===moveright ∧ holds[position[x-y],s]) ∨
  (i=!=moveleft ∧ i=!=moveright ∧
   holds[position[x],s]))/;legal[do[{i,y},s]];

holds[observation[x_],do[{i_, y_},s_]] :=
       ((i === find ∧ x == y)
         ∨ (i =!= find ∧
              x == 0)) /; legal[do[i, y, s]];
```

In this example, we define two fluents in terms of the position of the walking man. First, `hawk` is a fluent that tells us whether or not the walking man is to the right of the initial position. Also, the fluent `center` holds whenever the position of the walking man is within one unit from the center. Notice that these two fluents are acting as abbreviations for their corresponding definitions.

```
holds[hawk,s_]:=pos[s]≥0;
holds[center,s_]:=pos[s]≥-1∧pos[s]≤+1;
```

Given the above domain specification, along with the general domain independent specification of section , we can query our oracle. First, we ask general situation calculus questions that take a specific situation (built with specific action executions, always starting at `s0`) and asking various properties of such a situation:

```
legal[do[{moveright,7},s0]]
```

```
True
holds[position[x],do[{moveright,7},s0]]
-7+x==0
holds[hawk,do[{moveright,7},
                do[{moveleft,2},s0]]]
False
holds[hawk,do[{moveright,7},
                do[{moveright,2},s0]]]
True
holds[center,do[{moveleft,1},s0]]
True
```

Now, we estimate the real probabilities of fluents `hawk` and `center` holding at the specified situations, by using a Monte–Carlo approach with `1000` points. Belief updates for these fluents are also computed. It is easy to verify that the results are very close to the theoretical results expected.

```
prob[hawk,{moveright},s0,1000]
1.
prob[hawk,{moveleft},s0,1000]
0.
prob[hawk,{moveright,moveleft},s0,1000]
0.41
prob[center,{moveleft},s0,1000]
0.3
prob[center,{moveleft,moveright},s0,1000]
0.299
bel[center,{moveright},s0,find,1.8,1000]
0.161663
bel[center,{moveright},s0,find,0.5,1000]
0.704174
bel[hawk,{moveright,moveleft},s0,
                        find,0,1000]
0.482428
```

CONCLUSIONS

With this paper we continued our work on the development of a logical language to support Knowledge Representation and Reasoning for dynamic domains that have uncertain actions. The main contribution of the research reported here (which should be considered upon those already established in [Mateus et al 2001]) is the effective treatment of *probabilistic temporal projection* and *probabilistic update belief* on agents having discrete, continuous or mixed probability distributions modeling their uncertain actions. In detail, we consider an extension of the Situation Calculus, which we call the Probabilistic Situation Calculus (PSC), and endow this logic with terms to deal with the probabilities related with beliefs. This language will be the ground to specifying an agent that executes uncertain actions. Since PSC is a first order logic, first-order reasoning is inherited and it is shown to be sound to a given semantics.

To deal with probabilistic reasoning we provide a Monte-Carlo algorithm as an efective interpretation of the terms dealing with beliefs, all other reasoning on real numbers relies on an oracle, that we implement using MATHEMATICA. Probabilistic Situation Calculus specifications can be translated to a conditional rewriting system, where it is also possible to encode the given Monte-Carlo algorithms. With the frogbot example, we were able to give an idea of the reasoning capabilities of the rewriting system of MATHEMATICA endowed with our algorithms. A detailed comparison with related works, as well as the treatment of observation planning and rewards is also to be included in an extended version of this paper.

Acknowledgements

Paulo Mateus was partially supported by FCT grant SFRH/BPD/5625/2001 and FCT project POCTI/-2001/MAT/37239. António Pacheco was partially supported by FCT grant SFRH/BSAB/251/01 and FCT projects POSI/34826/CPS/2000 and POSI/-42069/CPS/2001.

References

F. Bacchus, J. Halpern, and H. Levesque. Reasoning about noisy sensors and effectors in the Situation Calculus. *Artificial Intelligence*, 111(1-2):171–208, 1999.

C. Baral and L. Tuan. Reasoning about Actions in a Probabilistic Setting. In *Commonsense 2001, Fifth Symposium on Logical Formalizations of Commonsense Reasoning*, 2001. URL = `http://www.-cs.nyu.edu/faculty/davise/commonsense01/`.

M. Evans and T. Swartz. *Approximating integrals via Monte Carlo and deterministic methods.* Oxford University Press, Oxford, 2000.

H. Grosskreutz and G. Lakemeyer. Turning High-Level Plans into Robot Programs in Uncertain Domains. In *ECAI*, 2000.

H. Grosskreutz and G. Lakemeyer. Belief Update in the pGOLOG Framework. In *Commonsense 2001, Fifth Symposium on Logical Formalizations of Commonsense Reasoning*, 2001. URL = `http://www.-cs.nyu.edu/faculty/davise/commonsense01/`.

P. Mateus, A. Pacheco, J. Pinto, A. Sernadas, and C. Sernadas. Probabilistic situation calculus. *Annals of Mathematics and Artificial Intelligence*, 32(1/4):393–431, 2001.

J. 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`.

J. Pinto. Compiling ramification constraints into effect axioms. *Computational Intelligence*, 15(3):280–307, 1999.

J. Pinto, A. Sernadas, C. Sernadas, and P. Mateus. Non-determinism and uncertainty in the Situation Calculus. In *Proceedings of the 12th International Florida AI Research Symposium, FLAIRS'99*, pages 454–460. AAAI Press, 1999.

J. Pinto, A. Sernadas, C. Sernadas, and P. Mateus. Non-determinism and uncertainty in the Situation Calculus. *International Journal of Uncertainty, Fuzziness and Knowledge Based Systems*, 8(2), 2000.

D. Poole. Decision theory, the Situation Calculus, and conditional plans. *Linköping Electronic Articles in Computer and Information Science*, 3(8), 1998. URL = `http://www.ep.liu.se/ea/cis/1998/008/`.

R. 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.

R. Reiter. Knowledge in action: Logical foundations for describing and implementing dynamical systems. Book Draft, available from `http://www.cs.utoronto.ca/cogrobo`, 2000.

S. Resnick. *A Probability Path.* Birkhäuser Boston Inc., Boston, MA, 1999.

S. Wolfram. *The Mathematica Book.* Wolfram Media, 3 edition, 1996.