Extending PDDL to Model Stochastic Decision Processes

Håkan L. S. Younes
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
lorens@cs.cmu.edu

Abstract

We present an extension of PDDL for modeling stochastic decision processes. Our domain description language allows the specification of actions with probabilistic effects, exogenous events, and actions and events with delayed effects. The result is a language that can be used to specify stochastic decision processes, both discrete-time and continuous-time, of varying complexity. We also propose the use of established logic formalisms, taken from the model checking community, for specifying probabilistic temporally extended goals.

Introduction

A standard domain description language, PDDL (McDermott 2000; Fox & Long 2002b), for deterministic planning domains and the biannual International Planning Competition, first held in 1998, have resulted in a large library of benchmark problems enabling direct comparisons of different deterministic planners. For the 4th International Planning Competition to be held in 2004, there are plans to include a track for probabilistic planners. This will require a domain description language for specifying probabilistic planning domains. In this paper, we propose such a domain description language that in many ways can be seen as an extension of PDDL.

We start by introducing a PDDL-like syntax for specifying actions with probabilistic effects, which allows us to define *Markov decision processes* (MDPs). We then go on to introduce exogenous events, as well as actions and events with random delay. The result is a domain description language that can be used for specifying a wide range of stochastic decision processes, from MDPs to *generalized semi-Markov decision processes* (GSMDPs). A GSMDP can be viewed as the composition of concurrent *semi-Markov decision processes* (SMDPs), and captures the essential dynamical structure of a *discrete event system* (Glynn 1989).

A discrete event system consists of a set of states S and a set events E. At any point in time, the system occupies some state $s \in S$. The system remains in state s until the occurrence of an event $e \in E$, at which point the system instantaneously transitions to a state s' (possibly the same state as s). Our domain description language can be used to specify both continuous-time and discrete-time discrete event systems. By including the process concept from level 5 of PDDL+ (Fox & Long 2002a), we could also specify stochastic hybrid systems, but that is beyond the scope of this paper.

As a formalism for specifying probabilistic goal conditions we propose PCTL (Hansson & Jonsson 1994) for discrete-time domains and CSL (Baier, Katoen, & Hermanns 1999) for continuous-time domains. This permits the specification of planning deadlines and maintenance and prevention goals, in addition to the traditional achievement goals. The benefit of using established logic formalisms for goal specification is that we can take advantage of recent developments in probabilistic model checking for efficient plan verification.

We leave the representation of plans open. The sole focus of this paper is the representation of probabilistic planning domains.

Actions with Probabilistic Effects

An important aspect of stochastic decision processes is that actions can have probabilistic effects. We adopt a model of stochastic actions that is a variation of *factored probabilistic STRIPS operators* proposed by Dearden & Boutilier (1997). A stochastic action a consists of a precondition ϕ and a consequence set $C = \{c_1, \ldots, c_n\}$. Each consequence c_i has a trigger condition ϕ_i with a corresponding effects list $\mathcal{E}_i = \langle p_1^i, E_1^i; \ldots; p_{k_i}^i, E_{k_i}^i \rangle$, where E_j^i is a set of literals and $p_j^i \in [0, 1]$ is a probability associated with the jth literal set. We require that $\sum_{j=1}^{k_i} p_j^i = 1$.

Semantics

In order for an action with precondition ϕ to be applicable in a state s, ϕ must hold in s. A state is a set of atoms that hold and bindings of functional expressions to rational values. We propose that executing an action a whose precondition is not satisfied be given the meaning that a has no effects (cf. Kushmerick, Hanks, & Weld 1995), instead of this being a violation as is the case for deterministic actions in PDDL. The precondition ϕ can in this way be viewed as a factored trigger condition common to all consequences in C. The semantics of stochastic actions can then be stated as follows.

When applying a stochastic action $a = \langle \phi, C \rangle$ to a state s, an effect set is selected for each consequence $c_i \in C$.

Let $x(i) \in [1, k_i]$ denote the index of the selected effect set for c_i, with $x(i)$ being a sample of a random variable X_i such that $\Pr[X_i = j] = p^i_j$. Let the *acting* effect set for a consequence c_i be

$$\tilde{E}^i_{x(i)} = \begin{cases} E^i_{x(i)} & \text{if } s \models \phi \wedge \phi_i \\ \emptyset & \text{otherwise} \end{cases} .$$

The acting effect set for action a is then the union of acting effect sets for the consequences:

$$\tilde{E} = \bigcup_{i=1}^{n} \tilde{E}^i_{x(i)}$$

We divide $\tilde{E}$ into disjoint sets: $\tilde{E}^+$ representing positive literals, $\tilde{E}^-$ representing negative literals, and $\tilde{E}^u$ representing effects updating value bindings for functional expressions. The result of applying the stochastic action a to the state s is a state s' with atoms $\big(atoms(s) \setminus \{p : \neg p \in \tilde{E}^-\}\big) \cup \tilde{E}^+$ and bindings of functional expressions to values updated in accordance with $\tilde{E}^u$.

We require that consequences with mutually consistent trigger conditions have *commutative* effects. This means that the successor state is the same after applying an action to a state s regardless of the order in which the acting effect sets of enabled consequences are applied to s.

Consequences in our stochastic action model is closely related to *action aspects* in the model of Dearden & Boutilier. Each consequence $c_i = \langle \phi_i, \mathcal{E}_i \rangle$ of an action with precondition ϕ corresponds to an action aspect with discriminant set $\{\phi \wedge \phi_i, \neg(\phi \wedge \phi_i)\}$ and effects lists $\mathcal{E}_i$ and $\langle 1, \emptyset \rangle$. The condition that mutually consistent discriminants taken from distinct aspects of an action have effects lists with no common atoms corresponds to our requirement of commutative effects for consequences with mutually consistent trigger conditions.

Syntax

Stochastic actions can be specified by extending the PDDL syntax for action effects with a probabilistic construct inspired by Bonet & Geffner (2001). Figure 1 shows the proposed extension. The syntax we propose does not allow nested probabilistic statements in effects lists or conditional effects inside probabilistic statements, which is in line with the design decision for PDDL2.1 to disallow nesting of conditional effects. Such language constructs would not add any expressiveness.

As it stands, there is a clear correspondence between the syntax and the representation of stochastic actions introduced above. An effects list is specified as

$$\texttt{(probabilistic } p^i_1 \; E^i_1 \; \ldots \; p^i_{k_i} \; E^i_{k_i}\texttt{)}.$$

The above statement also represents a consequence with a trigger condition $\phi_i = true$. Consequences with non-trivial trigger conditions are specified using conditional effects:

$$\texttt{(when } \phi_i \texttt{ (probabilistic } p^i_1 \; E^i_1 \; \ldots \; p^i_{k_i} \; E^i_{k_i}\texttt{))}$$

Figure 2 gives a specification in the extended PDDL of the stochastic move action used by Dearden & Boutilier as an example. A statement such as

```
<effect>      ::= <d-effect>
<effect>      ::= (and <effect>*)
<effect>      ::= (forall (<typed list(variable)>) <effect>)
<effect>      ::= (when <GD> <d-effect>)
<d-effect>    ::= (probabilistic <prob-eff>+)
<d-effect>    ::= <a-effect>
<prob-eff>    ::= <probability> <a-effect>
<a-effect>    ::= (and <p-effect>*)
<a-effect>    ::= <p-effect>
<p-effect>    ::= (not <atomic formula(term)>)
<p-effect>    ::= <atomic formula(term)>
<p-effect>    ::= (<assign-op> <f-head> <f-exp>)
<probability> ::= Any rational number in the interval [0, 1].
```

Figure 1: PDDL extension for probabilistic effects.

```
(:action move
         :parameters ()
         :effect (and (when (office)
                          (probabilistic 0.9 (not (office))))
                       (when (not (office))
                          (probabilistic 0.9 (office)))
                       (when (and (rain) (not (umbrella)))
                          (probabilistic 0.9 (wet)))))
```

Figure 2: Specification of stochastic move action in probabilistic PDDL.

```
(probabilistic 0.9 (wet))
```

with the probabilities not adding up to 1 is meant as a syntactic sugar for

```
(probabilistic 0.9 (wet) 0.1 (and)),
```

where `(and)` represents an empty effect set.

Numeric effects can be used in combination with probabilistic effects, although this could result in a stochastic process with an infinite state space. We therefore propose the introduction of a bounded integer type, `(integer` *low high*`)`, in addition to the standard PDDL type, `number`, for functional expressions. This provides a straightforward way of ensuring a finite state space. For example,

```
(:functions (power ?x) - (integer 0 10))
```

effectively defines an integer state variable $power_x \in [0, 10]$ for each object x in the domain.

Expressiveness

For now, we assume a simple discrete model of time, where time is progressing in unit steps with each state transition (execution of an action). We can model discrete-time Markov decision processes (MDPs) using stochastic actions as defined in this section. Later on we will consider richer time and action models that will allow us to model more complex stochastic decision processes.

Exogenous Events

Boutilier, Dean, & Hanks (1999) make a distinction between *implicit-event models* where the effects of the environment are factored into the representation of stochastic actions, and *explicit-event models* where change caused by the environment is modeled separately from change caused by actions selected for execution by the decision maker. We have so far only provided the means for specifying implicit-event models. It is sometimes convenient, however, to model environmental effects separate from effects of consciously chosen actions. For this purpose we introduce the notion of an *exogenous event*, $e = \langle \phi, C \rangle$, having the same structure as a stochastic action.

Semantics

An exogenous event e is given the same semantics as a stochastic action, except that the triggering of e is beyond the control of the decision maker. When in a state s, any action chosen for s and all events with a precondition ϕ that holds in s are applied to s, producing a successor state s' at the next time step.

With exogenous events in the domain model it becomes possible to have an action and one or more exogenous events triggering within the same unit time interval, and the successor state s' can then depend on the order in which the action and the events are applied to the current state s. We could require events and actions that can be enabled simultaneously to be commutative, meaning that the successor state is independent of the order in which the events and actions are applied to s, but it would be hard for a domain designer to adhere to this requirement when constructing a large domain with many exogenous events. Instead we choose to assign an equal probability to all orderings of enabled event and actions in a state. So, for example, if an action adding the atomic proposition a and an event deleting a is enabled in a state s, then there is a 0.5 probability of a holding in the next state.

Boutilier, Dean, & Hanks (1999) consider other ways of dealing with simultaneity, but these typically requiring additional information from the domain designer. The fact, though, is that simultaneity almost always is an artifact of using a discrete-time model for an inherently continuous-time stochastic process, and the probability of two events triggering at exactly the same time becomes zero if we work directly with a continuous-time model.

Syntax

We propose using the same syntax for specifying exogenous events as stochastic actions, except that the keyword `:event` is used instead of `:action`. The `:event` keyword was introduced in level 5 of PDDL+ (Fox & Long 2002a) for the specification of deterministic events, and our exogenous events can be viewed as stochastic extensions of PDDL+ level 5 events.

We can break the stochastic move action in Figure 2 into an action modeling intended effects and an exogenous event modeling environmentally triggered effects. Figure 3 shows how this would be done.

Expressiveness

The addition of exogenous events does not add to the expressiveness of our specification language: we can still only model discrete-time MDPs. At this point it is merely added for the convenience of the modeler, but once we consider more complex processes we will see that the effects of exogenous events cannot in a reasonable way be factored into the effects of actions.

```
(:action move
        :parameters ()
        :effect (and (when (office)
                           (probabilistic 0.9 (not (office))))
                     (when (not (office))
                           (probabilistic 0.9 (office)))))

(:event make-wet
        :parameters ()
        :precondition (and (rain) (not (umbrella)))
        :effect (probabilistic 0.9 (wet)))
```

Figure 3: Partial specification of explicit-event model in probabilistic PDDL.

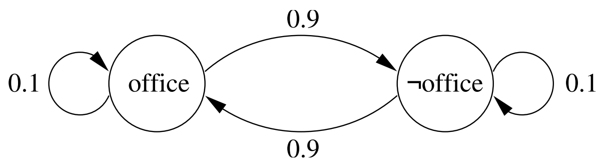


Figure 4: State-transition model for executing the stochastic move action of Figure 3.

Delayed Actions and Events

Consider the stochastic move action of Figure 3 in isolation. Figure 4 shows the state-transition model for executing this action both when in the office and when not in the office. The stochastic move action is executed at every time step and has a 0.9 probability of succeeding each time. The time spent in a state before the action succeeds is a random variable with a geometric distribution $G(0.9)$. Instead of thinking of the stochastic move action as being executed at every time step, we can think of it as being executed once in each state but with a delayed effect. The delay is in this case a random variable with distribution $G(0.9)$. Figure 5 shows the state-transition model for a delayed move action, where probability distributions instead of probabilities are associated with each state transition.

A delayed action a is a triple $\langle \phi, C, F(t) \rangle$, where ϕ is the enabling condition of a, C is the consequence set of a defined in the same way as for stochastic actions, and $F(t)$ is the cumulative distribution function (cdf) for the delay from when a is enabled until it triggers. We require that $F(0) = 0$, meaning that an action must trigger strictly after it becomes enabled. Delayed exogenous events are defined analogously. A regular stochastic action (exogenous event) can be viewed as a delayed action (event) with a cdf satisfying the condition $F(1) = 1$, i.e. it always triggers within one time unit from when it is enabled,but with no additional assumptions being made regarding the shape of $F(t)$.

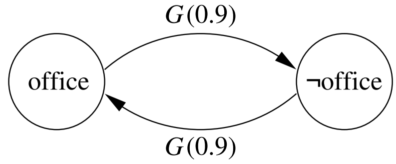


Figure 5: State-transition model for executing a delayed move action.

```
<delayed-action-def> ::= (:delayed-action <name>
                          :parameters (<typed list<variable>)
                          :delay <delay-distribution>
                          [:condition <GD>]
                          [:effect <effect>])
<delayed-event-def>  ::= (:delayed-event <name>
                          :parameters (<typed list<variable>)
                          :delay <delay-distribution>
                          [:condition <GD>]
                          [:effect <effect>])
<delay-distribution> ::= Any distribution s.t. Pr[delay ≤ 0] = 0.
```

Figure 6: PDDL extension for delayed actions and events.

Semantics

For now, let us assume that all delay distributions are *memoryless*. We will consider general delay distribution in the next section. The probability distribution of a random variable X is memoryless if $\Pr[X > t + \Delta t | X > t] = \Pr[X > \Delta t]$ for all $t, \Delta t \geq 0$. For an action a with a memoryless delay distribution this means that if a has been enabled for t time units without triggering, then the remaining delay has the same distribution as if a had just been enabled. The geometric distribution mentioned earlier in this section is a memoryless distribution, and so is its continuous analog: the exponential distribution. The semantics of delayed actions and events with memoryless delay distributions is as follows.

Assume we are entering state s at time t. Any action chosen by the decision maker to be enabled in s and all events with a condition ϕ holding in s race to trigger first. Let e^* be the event or action with the shortest delay in s and let d^* be the delay of e^* in s. We then get the successor state s' at time $t + d^*$ by applying e^* to s. An action or event enabled in s may still be enabled in s', but this is inconsequential when all delay distributions are memoryless. If an action or event did not trigger in s and is not enabled in s', then that action or event is simply canceled. If multiple events or actions have minimum delay d^* in s, then all those events and actions are simultaneously applied to s to produce s' at time $t + d^*$.

Syntax

The proposed syntax for specifying delayed actions and events is given in Figure 6. Delayed actions can be viewed as a stochastic variation of the deterministic durative actions available in PDDL+. A delayed action with a deterministic delay distribution $D(x)$ and enabling condition ϕ corresponds to a durative action with duration x, invariant condition ϕ, and effects associated with the end of the durative action.

Figure 7 shows the partial specification of an explicit-event model with delayed actions and events.

Expressiveness

By just considering memoryless delay distributions, we are nevertheless only able to model MDPs. With geometric delay distributions we have a discrete-time MDP, while with exponential delay distributions we have a continuous-time MDP. Moreover, exogenous events still do not add expressiveness as they can easily be factored into the representation of actions. We can combine all actions and events enabled in a state s into one single action.

```
(:delayed-action move
                 :parameters ()
                 :delay (geometric 0.9)
                 :effect (and (when (office)
                                (not (office)))
                              (when (not (office))
                                (office))))

(:delayed-event make-wet
                :parameters ()
                :delay (geometric 0.9)
                :precondition (and (rain) (not (umbrella)))
                :effect (wet))
```

Figure 7: Partial specification of explicit-event model with delayed actions and events.

```
(:delayed-action move
                 :parameters ()
                 :delay (geometric 0.99)
                 :effect (and (when (office)
                                (probabilistic 10/11
                                               (not (office))))
                              (when (not (office))
                                (probabilistic 10/11 (office)))
                              (when (and (rain) (not (umbrella)))
                                (probabilistic 10/11 (wet))
```

Figure 8: Partial specification of implicit-event model with delayed move action.

For a discrete-time model, let $G(p_i)$ be the distribution of the ith action or event enabled in a state s. The probability of at least one action or event triggering in s after one time unit is $p = 1 - \prod_i (1 - p_i)$, and the probability of the ith action or event triggering after one time unit given that something triggers is p_i/p. We can therefore represent all actions and events enabled in s with a single action having delay distribution $G(p)$, and with the effects of the ith action or event weighted by p_i/p. Figure 8 shows the implicit-event representation of the action and event in Figure 7.

A similar transformation of an explicit-event model to an implicit-event model can be made for continuous-time models. Let $E(\lambda_i)$ be the delay distribution of the ith action or event enabled in a state s, with λ_i being the *rate* of the exponential delay distribution. The rate at which some action or event triggers in s is $\lambda = \sum_i \lambda_i$, and the probability of the ith action or event triggering before any other action or event is λ_i/λ. We can use this information to construct a single action representing all actions and events enabled in s. Part of a continuous-time explicit-event model is given in Figure 9 and the corresponding implicit-event model is given in Figure 10.

General Delay Distributions

Although memoryless distributions can be used to adequately model many real-world phenomena, they are many times insufficient for accurately capturing certain aspects of stochastic processes. Hardware failure, for example, is often more accurately modeled using a Weibull distribution rather than an exponential distribution (Nelson 1985).

Figure 11 shows an example domain with general delay distributions for actions and events. Here, we have associated a uniform delay distribution with the move action and a Weibull distribution with the make-wet event.

```
(:delayed-action move
                 :parameters ()
                 :delay (exponential 3)
                 :effect (and (when (office)
                                    (not (office)))
                              (when (not (office))
                                    (office))))

(:delayed-event make-wet
                :parameters ()
                :delay (exponential 2)
                :precondition (and (rain) (not (umbrella)))
                :effect (wet))
```

Figure 9: Partial specification of continuous-time explicit-event model with delayed actions and events.

```
(:delayed-action move
                 :parameters ()
                 :delay (exponential 5)
                 :effect (and (when (and (office)
                                          (rain) (not (umbrella)))
                                 (probabilistic 0.6 (not (office))
                                                0.4 (wet)))
                              (when (and (office)
                                          (or (not (rain))
                                              (umbrella)))
                                 (probabilistic 0.6
                                                (not (office))))
                              (when (and (not (office))
                                          (rain) (not (umbrella)))
                                 (probabilistic 0.6 (office)
                                                0.4 (wet)))
                              (when (and (not (office))
                                          (or (not (rain))
                                              (umbrella)))
                                 (probabilistic 0.6 (office)))))
```

Figure 10: Partial specification of continuous-time implicit-event model with delayed move action.

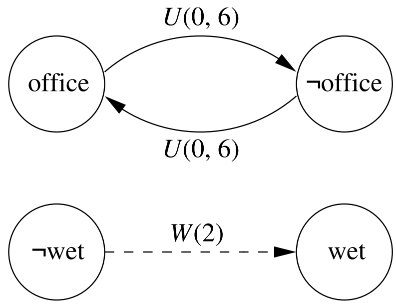


Figure 12: State-transition model for move action (above) and make-wet event (below).

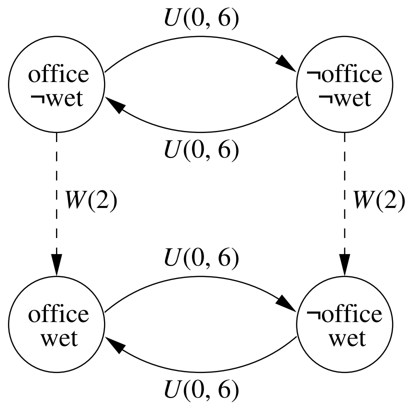


Figure 13: Composite state-transition model for move action and make-wet event.

The state-transition model for the action and event considered separately is depicted in Figure 12. Each state-transition model corresponds to a *semi-Markov process* (SMP; Howard 1971). However, when viewing the domain model as a whole—as the composition of concurrent SMPs (Figure 13)—we have what is called a *generalized semi-Markov process* (GSMP; Glynn 1989).

GSMPs differ from SMPs in that the delay distribution of an enabled event can depend not only on the current state but on the entire path taken to that state. Consider the state-transition model in Figure 13. Assume that we start out not being in the office and not being wet (upper right state). The move action and the make-wet event are both enabled. Say that the make-wet event happens to trigger before the move action after having been in the current state for 3 time units, causing a transition to the lower right state where the move action is still enabled. Because the move action already has been enabled for 3 time units in the previous state without triggering, the delay distribution for the move action in the new state is in effect $U(0, 3)$. If, on the other hand, we had entered the lower right state by executing the move action when being wet in the office (lower left state), then the delay distribution of the move action would have been $U(0, 6)$. This history dependence occurs because we are not using memoryless distributions.

```
(:delayed-action move
                 :parameters ()
                 :delay (uniform 0 6)
                 :effect (and (when (office)
                                    (not (office)))
                              (when (not (office))
                                    (office))))

(:delayed-event make-wet
                :parameters ()
                :delay (weibull 2)
                :precondition (and (rain) (not (umbrella)))
                :effect (wet))
```

Figure 11: Partial specification of continuous-time explicit-event model with general delay distributions.

Generalized Semi-Markov Processes

GSMPs, first introduced by Matthes (1962), have the following components:

- A set S of states.
- A set E of events.
- For each state $s \in S$, a set $E(s) \subset E$ of events enabled in s.
- For each pair $\langle s, e \rangle \in S \times E(s)$, a probability distribution $p(s'; s, e)$ over S giving the probability of the next state being s' if event e triggers in state s.
- For each event e, a cdf $F(t; e)$, s.t. $F(0; e) = 0$, giving the probability that e has triggered t time units after it was last enabled.

A generalized semi-Markov *decision* process (GSMDP) is a GSMP with a subset of E being controllable events (i.e. actions).

By allowing general delay distributions, we can specify GSMDPs using the proposed extensions of PDDL. The preconditions of actions and events determine the sets $E(s)$, the probability distributions $p(s'; s, e)$ can be derived from conditional probabilistic effects, and the distributions $F(t; e)$ correspond directly to the delay distributions of actions and events.

The semantics of a GSM(D)P is best described in terms of discrete event simulation (Shedler 1993). We associate a real-valued clock $c(e)$ with each event $e \in E$ that indicates the time remaining until e is scheduled to occur. The system starts in some initial state s with events $E(s)$ enabled. For each enabled event $e \in E(s)$, we sample a duration according to the cdf $F(t; e)$ and set $c(e)$ to the sampled value. Let e^* be the event in $E(s)$ with the shortest duration, and let $c^* = c(e^*)$. The event e^* becomes the triggering event in s. When e^* triggers, we sample a next state s' according to the probability distribution $p(s'; s, e^*)$ and update the clock for each event $e \in E(s')$ enabled in the next state as follows:

- If $e \in E(s) \setminus \{e^*\}$, then subtract c^* from $c(e)$.
- If $e \notin E(s) \setminus \{e^*\}$, then sample a new duration according to the cdf $F(t; e)$ and set $c(e)$ to the sampled value.

An event enabled in s but not in s' will have its clock reset next time it becomes enabled. The first condition above highlights the fact that GSM(D)Ps are non-Markovian, as the durations for events are not independent of the history. The system evolves by repeating the process of finding the triggering event in the current state, and updating clock values according to the scheme specified above.

Summary of Expressiveness

Figure 14 shows the hierarchy of stochastic decision processes that can be specified using our proposed probabilistic extension of PDDL. The most general class is GSMDPs, which allow for concurrency, general delay distributions, and probabilistic effects. A GSMDP is an SMDP if no action or event can be enabled in consecutive states without triggering, and it is an MDP if all delay distributions are memoryless (Glynn 1989).

With general delay distributions, there is no longer an easy way to factor the effects of exogenous events into the effects of actions. The delay distribution of the combined action would be the distribution of the minimum of the individual delay distributions. The minimum of exponential distributions with rates λ_i is simply an exponential distribution with rate $\sum_i \lambda_i$, but for general distributions there is typically no simple distribution for the minimum. Neither is it in general possible to obtain a closed-form expression for the probability of a specific event triggering first. Exogenous events is therefore more than just a modeling convenience once we allow general delay distribution.

A GSMDP can be approximated with an MDP by approximating each general delay distribution with a phase-type distribution. Figure 15 shows two commonly used phase-type distributions. The phase of each approximating distribution becomes part of the state-space for the MDP, which potentially can lead to state-space explosion. It is therefore desirable to approximate a general distribution with a phase-type distribution having few phases, while still matching at least the first three moments of the general distribution (Osogami & Harchol-Balter 2003).

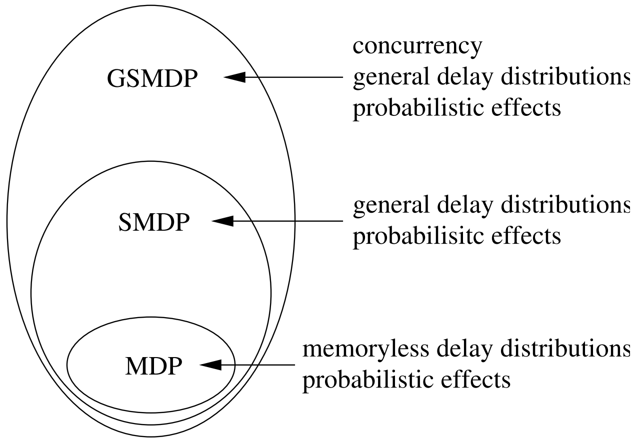
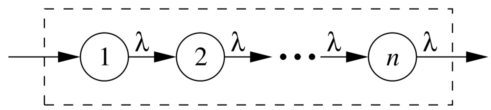
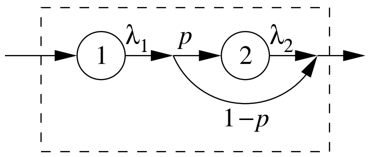


Figure 14: A hierarchy of stochastic decision processes.



(a) An n-phase Erlang distribution.



(b) A two-phase Coxian distribution.

Figure 15: Examples of phase-type distributions.

We can model both discrete-time and continuous-time stochastic decision processes. German (2000) discusses techniques for approximating a continuous-time Markov process with a discrete-time Markov process that can be used in case a discrete-time model is preferred.

Probabilistic Planning Problems

A probabilistic planning problem is commonly specified as an initial state s_0 (or initial distribution over states), a set of goal states G, and a probability threshold p. A plan is considered a solution to a problem if the set of paths from s_0 to states in G has probability $p' \geq p$ (Farley 1983; Blythe 1994; Goldman & Boddy 1994; Kushmerick, Hanks, & Weld 1995; Lesh, Martin, & Allen 1998). Drummond &

Bresina (1990) suggest the need for maintenance and prevention goals, in addition to goals of achievement traditionally considered in probabilistic planning, and propose a branching temporal logic for specifying temporally extended goals. Dean *et al.* (1995) embrace a similar view, but take a decision-theoretic approach with goals encoded using utility functions.

We propose a definition of probabilistic planning problems closely related to that of Drummond & Bresina, adopting PCTL (Hansson & Jonsson 1994) and its continuous-time analog CSL (Baier, Katoen, & Hermanns 1999) as a logic for specifying probabilistic temporally extended goals. We define a planning problem as an initial state s_0 and a CSL (PCTL) formula ϕ, and a solution is a plan that makes ϕ true in s_0.

Probabilistic Temporally Extended Goals

The syntax of CSL (PCTL) is defined as

$$\phi ::= \mathit{true}\big|a\big|\phi \wedge \phi\big|\neg\phi\big|\mathcal{P}_{\bowtie p}\left(\phi\,\mathcal{U}^{\leq t}\,\phi\right)\big|\mathcal{P}_{\bowtie p}\left(\phi\,\mathcal{U}\,\phi\right),$$

where a is an atomic proposition, $p \in [0,1]$, $t \in \mathbb{R}_{\geq 0}$ ($t \in \mathbb{Z}_{\geq 0}$ for PCTL), and $\bowtie \in \{\geq, >\}$.

Regular logic operators have their usual semantics. A probabilistic formula $\mathcal{P}_{\bowtie p}\left(\rho\right)$ holds in a state s if and only if the set of paths starting in s and satisfying the path formula ρ is p' and $p' \bowtie p$. A path of a stochastic process is a sequence of states and holding times:

$$\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \ldots$$

A path formula $\phi_1\ \mathcal{U}^{\leq t}\,\phi_2$ ("time-bounded until") holds over a path σ if and only if ϕ_2 holds in some state s_i such that $\sum_{j=0}^{i-1} t_j \leq t$ and ϕ_1 holds in all state s_j for $j < i$. The formula $\phi_1\,\mathcal{U}\,\phi_2$ ("until") holds over a path σ if and only if $\phi_1\,\mathcal{U}^{\leq t}\,\phi_2$ holds over σ for some $t \geq 0$.

We can derive other common logic and path operators, for example:

$$\begin{aligned}
\mathit{false} &\equiv \neg\mathit{true}\\
\phi_1 \vee \phi_2 &\equiv \neg(\neg\phi_1 \wedge \neg\phi_2)\\
\mathcal{P}_{\bowtie p}\left(\Diamond^{\leq t}\,\phi\right) &\equiv \mathcal{P}_{\bowtie p}\left(\mathit{true}\,\mathcal{U}^{\leq t}\,\phi\right)\\
\mathcal{P}_{\geq p}\left(\phi_1\,\mathcal{W}^{\leq t}\,\phi_2\right) &\equiv \neg\mathcal{P}_{>1-p}\left(\neg\phi_2\,\mathcal{U}^{\leq t}\,\neg(\phi_1 \vee \phi_2)\right)\\
\mathcal{P}_{>p}\left(\phi_1\,\mathcal{W}^{\leq t}\,\phi_2\right) &\equiv \neg\mathcal{P}_{\geq 1-p}\left(\neg\phi_2\,\mathcal{U}^{\leq t}\,\neg(\phi_1 \vee \phi_2)\right)\\
\mathcal{P}_{\bowtie p}\left(\Box^{\leq t}\,\phi\right) &\equiv \mathcal{P}_{\bowtie p}\left(\phi\,\mathcal{W}^{\leq t}\,\mathit{false}\right)
\end{aligned}$$

Intuitively, $\Diamond^{\leq t}\,\phi$ ("eventually") holds if ϕ becomes true within t time units, $\phi_1\,\mathcal{W}^{\leq t}\,\phi_2$ ("weak until") holds if either ϕ_1 remains true for t time units or ϕ_2 becomes true within t time units with ϕ_1 holding until then, and $\Box^{\leq t}\,\phi$ ("continuously") holds if ϕ continuously holds for t time units. These path operators can be defined without a time-bound in an analogous way.

Table 1 gives a few examples of goals that we can express using PCTL/CSL. In addition to regular achievement goals, we are able to specify goals with deadlines, safety constraints over execution paths, maintenance goals, and prevention goals.

Relation to Decision-Theoretic Planning

We can encode a probabilistic goal $\mathcal{P}_{\bowtie p}\left(\phi_1\,\mathcal{U}\,\phi_2\right)$ for a decision process M as a Markovian reward function for a modified decision process M'. This is done by making every state for which $\neg\phi_1 \vee \phi_2$ holds in M absorbing in M' and assigning reward one in M' to those states that satisfy ϕ_2 in M. All other states are assigned reward zero. The expected total undiscounted reward for M' then equals the probability of $\phi_1\,\mathcal{U}\,\phi_2$ holding in M. For a time-bounded formula $\mathcal{P}_{\bowtie p}\left(\phi_1\,\mathcal{U}^{\leq t}\,\phi_2\right)$, the time-bound serves as a planning horizon.

Bacchus, Boutilier, & Grove (1996; 1997) use PLTL, a past-tense variation of the linear temporal logic (LTL; Emerson *et al.* 1990), to specify desired plan behavior, and associate rewards with PLTL formulas. This enables the specification of non-Markovian rewards. A similar approach is suggested by Thiébaux, Kabanza, & Slaney (2002), who present a formalism for expressing non-Markovian rewards adapted to anytime solution methods.

Decision Epochs

For discrete-time MDPs decision epochs occur at every time point, meaning that the decision maker is allowed to select an action for execution at regular intervals of unit length. If we count movements from a state to itself as state transitions (caused by the triggering of an event with no effects on the current state), then we can say that decision epochs for discrete-time MDPs occur after every state transition.

A natural extension of this decision model to continuous-time and non-Markovian decision processes is to have decision epochs occur only at state transitions. This is the typical decision model for SMDPs (Howard 1971), and is also the model used by Younes, Musliner, & Simmons (2003) for GSMDPs.

Alternatively, we could allow policies that associate with each state s an action selection function $\pi(s,t)$, with the currently enabled action being a function of the time since the last state transition. This decision model is, for example, used by Doshi (1979). Decision epochs occur at every point in time with this model, making the number of decision epochs uncountable for continuous-time decision processes.

Plan Verification

Given a plan π, a stochastic domain model $\mathcal{M}$, and a planning problem $\langle s_0, \phi\rangle$, we accept π as a solution if ϕ holds in s_0 for the stochastic process $\mathcal{M}$ controlled by π. We can take advantage of recent developments in probabilistic verification to verify if a plan is a solution to a planning problem.

Interest in verification of probabilistic systems has been on the rise in the last ten years. Symbolic methods for probabilistic verification of discrete-time (Hansson & Jonsson 1994) and continuous-time (Baier, Katoen, & Hermanns 1999; Baier *et al.* 2000; Katoen *et al.* 2001) Markov processes have been proposed. PRISM (Kwiatkowska, Norman, & Parker 2002a; 2002b) is a fast symbolic probabilistic model checker for both PCTL and CSL formulae.

Goal description	Formula
reach office with probability at least 0.9	$\mathcal{P}_{\geq 0.9}\left(\Diamond\, \text{office}\right)$
reach office within 5 time units with probability at least 0.9	$\mathcal{P}_{\geq 0.9}\left(\Diamond^{\leq 5}\, \text{office}\right)$
reach office with probability at least 0.9 along paths where the recharging station can be reached within 17 time units with probability at least 0.5	$\mathcal{P}_{\geq 0.9}\left(\mathcal{P}_{\geq 0.5}\left(\Diamond^{\leq 17}\, \text{recharging}\right)\, \mathcal{U}\, \text{office}\right)$
maintain stability for at least 8.2 time units with probability at least 0.7	$\mathcal{P}_{\geq 0.7}\left(\Box^{\leq 8.2}\, \text{stable}\right)$
avoid becoming wet with probability at least 0.8	$\mathcal{P}_{\geq 0.8}\left(\Box\, \neg \text{wet}\right)$

Table 1: Probabilistic goals expressible in CSL and PCTL.

Model checking algorithms for more complex models have been proposed by Infante López, Hermanns, & Katoen (2001) (SMPs) and Kwiatkowska *et al.* (2000) (stochastic timed automata with clocks governed by general distributions). While verification of CSL properties without a time-bound is no harder for SMPs than for Markov processes, the proposed symbolic methods for verifying time-bounded properties of more general processes rely on techniques that are prohibitively complex, and for GSMPs no symbolic methods exist at all.

Younes & Simmons (2002) have developed an efficient sampling-based approach to verifying time-bounded probabilistic properties of general discrete event systems. For GSMPs without any restrictions on the class of delay distributions that can be used, there are currently no alternatives to sampling-based approaches. Without exhaustive sampling, we can never be certain that the result returned by a sampling-based approach is correct, but Younes & Simmons use statistical hypothesis testing techniques to bound the probability of an incorrect verification result.

Discussion

We have presented an extension of PDDL for modeling stochastic decision processes of varying complexity. Our representation of actions with probabilistic effects is in essence the same as that of Dearden & Boutilier (1997). In this paper we have tied this representation to a PDDL-like syntax. We have also extended the representation of stochastic actions to include actions with random delay, which allows us to specify SMDPs and GSMDPs.

We have extended the classical representation of probabilistic planning problems by using PCTL and CSL for specifying goals. This allows us to express, for example, deadlines and maintenance and prevention goals in addition to the traditional achievement goals. We have discussed work in probabilistic model checking that could be utilized for efficient plan verification.

The focus of this paper has been on the representation of planning problems and not on the representation or generation of plans. Much of the work in probabilistic and decision theoretic planning assumes an MDP model. Boutilier, Dean, & Hanks (1999) provides an excellent overview of planning with MDP models. Howard (1971) provides a good introduction to SMPs and SMDPs, and presents dynamic programming algorithms for solving decision-theoretic SMDP planning problems. Probabilistic planning with GSMDP domain models have been considered recently by Younes, Musliner, & Simmons (2003) who propose a sampling-based algorithm for generating stationary policies for GSMDPs, and decision-theoretic extensions of this work is discussed by Ha & Musliner (2002).

Acknowledgments. The author acknowledges Reid Simmons for helping him clarify his thoughts on issues discussed in this paper through many long and fruitful discussions. The author also thanks Michael Littman and the anonymous reviewers for helpful comments.

This material is based upon work supported by DARPA and ARO under contract no. DAAD19–01–1–0485, and a grant from the Royal Swedish Academy of Engineering Sciences (IVA). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the sponsors.

References

Bacchus, F.; Boutilier, C.; and Grove, A. 1996. Rewarding behaviors. In *Proc. Thirteenth National Conference on Artificial Intelligence*, 1160–1167. AAAI Press.

Bacchus, F.; Boutilier, C.; and Grove, A. 1997. Structured solution methods for non-Markovian decision processes. In *Proc. Fourteenth National Conference on Artificial Intelligence*, 112–117. AAAI Press.

Baier, C.; Haverkort, B. R.; Hermanns, H.; and Katoen, J.-P. 2000. Model checking continuous-time Markov chains by transient analysis. In *Proc. 12th International Conference on Computer Aided Verification*, volume 1855 of *LNCS*, 358–372. Springer.

Baier, C.; Katoen, J.-P.; and Hermanns, H. 1999. Approximate symbolic model checking of continuous-time Markov chains. In *Proc. 10th International Conference on Concurrency Theory*, volume 1664 of *LNCS*, 146–161. Springer.

Blythe, J. 1994. Planning with external events. In *Proc. Tenth Conference on Uncertainty in Artificial Intelligence*, 94–101. Morgan Kaufmann Publishers.

Bonet, B., and Geffner, H. 2001. GPT: A tool for planning with uncertainty and partial information. In *Proc. IJCAI-01 Workshop on Planning with Uncertainty and Incomplete Information*, 82–87.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research* 11:1–94.

Dean, T.; Kaelbling, L. P.; Kirman, J.; and Nicholson, A. 1995. Planning under time constraints in stochastic domains. *Artificial Intelligence* 76(1–2):35–74.

Dearden, R., and Boutilier, C. 1997. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence* 89(1–2):219–283.

Doshi, B. T. 1979. Generalized semi-Markov decision processes. *Journal of Applied Probability* 16(3):618–630.

Drummond, M., and Bresina, J. 1990. Anytime synthetic projection: Maximizing the probability of goal satisfaction. In *Proc. Eighth National Conference on Artificial Intelligence*, 138–144. AAAI Press.

Emerson, E. A.; Mok, A. K.; Sistla, A. P.; and Srinivasan, J. 1990. Quantitative temporal reasoning. In *Proc. 2nd International Conference on Computer Aided Verification*, volume 531 of *LNCS*, 136–145. Springer.

Farley, A. M. 1983. A probabilistic model for uncertain problem solving. *IEEE Transactions on Systems, Man, and Cybernetics* 13(4):568–579.

Fox, M., and Long, D. 2002a. PDDL+: Modelling continuous time-dependent effects. In *Proc. 3rd International NASA Workshop on Planning and Scheduling for Space*.

Fox, M., and Long, D. 2002b. PDDL2.1: An extension to PDDL for expressing temporal planning domains. Technical Report 20/02, University of Durham, Durham, UK.

German, R. 2000. *Performance Analysis of Communication Systems: Modeling with Non-Markovian Stochastic Petri Nets*. New York, NY: John Wiley & Sons.

Glynn, P. W. 1989. A GSMP formalism for discrete event systems. *Proceedings of the IEEE* 77(1):14–23.

Goldman, R. P., and Boddy, M. S. 1994. Epsilon-safe planning. In *Proc. Tenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers.

Ha, V., and Musliner, D. J. 2002. Toward decision-theoretic CIRCA with application to real-time computer security control. In *Papers from the AAAI Workshop on Real-Time Decision Support and Diagnosis Systems*, 89–90. AAAI Press. Technical Report WS-02-15.

Hansson, H., and Jonsson, B. 1994. A logic for reasoning about time and reliability. *Formal Aspects of Computing* 6(5):512–535.

Howard, R. A. 1971. *Dynamic Probabilistic Systems*, volume II. New York, NY: John Wiley & Sons.

Infante López, G. G.; Hermanns, H.; and Katoen, J.-P. 2001. Beyond memoryless distributions: Model checking semi-Markov chains. In *Proc. 1st Joint International PAPM-PROBMIV Workshop*, volume 2165 of *LNCS*, 57–70. Springer.

Katoen, J.-P.; Kwiatkowska, M.; Norman, G.; and Parker, D. 2001. Faster and symbolic CTMC model checking. In *Proc. 1st Joint International PAPM-PROBMIV Workshop*, volume 2165 of *LNCS*, 23–38. Springer.

Kushmerick, N.; Hanks, S.; and Weld, D. S. 1995. An algorithm for probabilistic planning. *Artificial Intelligence* 76(1–2):239–286.

Kwiatkowska, M.; Norman, G.; Segala, R.; and Sproston, J. 2000. Verifying quantitative properties of continuous probabilistic timed automata. In *Proc. 11th International Conference on Concurrency Theory*, volume 1877 of *LNCS*, 123–137. Springer.

Kwiatkowska, M.; Norman, G.; and Parker, D. 2002a. PRISM: Probabilistic symbolic model checker. In *Proc. 12th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, volume 2324 of *LNCS*, 200–204. Springer.

Kwiatkowska, M.; Norman, G.; and Parker, D. 2002b. Probabilistic symbolic model checking with PRISM: A hybrid approach. In *Proc. 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 2280 of *LNCS*, 52–66. Springer.

Lesh, N.; Martin, N.; and Allen, J. 1998. Improving big plans. In *Proc. Fifteenth National Conference on Artificial Intelligence*, 860–867. AAAI Press.

Matthes, K. 1962. Zur Theorie der Bedienungsprozesse. In *Transactions of the Third Prague Conference on Information Theory, Statistical Decision Functions, Random Processes*, 513–528. Publishing House of the Czechoslovak Academy of Sciences.

McDermott, D. 2000. The 1998 AI planning systems competition. *AI Magazine* 21(2):35–55.

Nelson, W. 1985. Weibull analysis of reliability data with few or no failures. *Journal of Quality Technology* 17(3):140–146.

Osogami, T., and Harchol-Balter, M. 2003. A closed-form solution for mapping general distributions to minimal PH distributions. Technical Report CMU-CS-03-114, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Shedler, G. S. 1993. *Regenerative Stochastic Simulation*. Boston, MA: Academic Press.

Thiébaux, S.; Kabanza, F.; and Slaney, J. 2002. Anytime state-based solution methods for decision processes with non-markovian rewards. In *Proc. Eighteenth Conference on Uncertainty in Artificial Intelligence*, 501–510. Morgan Kaufmann Publishers.

Younes, H. L. S., and Simmons, R. G. 2002. Probabilistic verification of discrete event systems using acceptance sampling. In *Proc. 14th International Conference on Computer Aided Verification*, volume 2404 of *LNCS*, 223–235. Springer.

Younes, H. L. S.; Musliner, D. J.; and Simmons, R. G. 2003. A framework for planning in continuous-time stochastic domains. In *Proc. 13th International Conference on Automated Planning and Scheduling*. AAAI Press. Forthcoming.