A Stochastic Model of Actions and Plans for Anytime Planning under Uncertainty

Sylvie Thiébaux
IRISA, Campus de Beaulieu,
35042 Rennes, France
thiebaux@irisa.fr

Joachim Hertzberg*
Dortmund University, Informatik VIII
44221 Dortmund, Germany
hertz@ls8.informatik.uni-dortmund.de

William Shoaff, Moti Schneider
FIT, CS Department
Melbourne, Fl 32901, USA
{wds, moti}@cs.fit.edu

Abstract. Some real world domains require planners to cope with uncertainty and time pressure. This paper describes a model of reaction plans that is based on (1) an action formalism allowing for incomplete state descriptions and context-dependent action with multiple effect-alternatives, and (2) state descriptions in probabilistic logic. This model is proposed as a basis for anytime planning under uncertainty.

As the field of planning matures, work focuses on the necessities of the real world. In particular, two topics are moving back into focus, which are important for this paper:

Coping with uncertainty. Knowledge about the planning domain is frequently incomplete, or the domain may not be completely controlled by the planner, or actions can have nondeterministic or context-dependent effects. There are a variety of approaches for tackling uncertainty at either planning or execution time; examples include reaction plans and reactive planning [20], decision-theoretic planning methods [12, 14], or execution monitoring and replanning [13, 27, Ch. 11]. Uncertainty is also tackled by theory-oriented work in reasoning about action and change [9, 3, 6].

Implementing time-bounded rationality. Generating optimal plans from first principles takes time. Under time pressure, a planner may wish to produce some plan in a fixed amount of time, and improve the plan quality as more time is allocated. Anytime algorithms [8] can be used to implement time-bounded rationality in the planning framework. They have been applied to solving time-dependent planning problems [2] and are also used for a broad class of planning applications, e.g. [28].

It is useful to combine work on these two topics, the perspective being to build planners that cope with uncertainty, and incrementally increase the plan quality if time permits, e.g., [10, 1].

We have earlier described PASCALE [23], a system for planning under uncertainty, based on a theory of actions with uncertain outcomes [3], which deals explicitly with uncertainty arising from the incompleteness or ambiguity of information. PASCALE's representation of reaction plans allows them to be generated from first principles, as well as revised off or on-line; this proved to be a good basis for exhibiting reactivity.

*On leave from GMD, I3.KI, Schloss Birlinghoven, 53757 Sankt Augustin, Germany.

In this paper, we address the problem of generating PASCALE style plans using an anytime algorithm. The crucial issue is to develop a framework for evaluating the quality of these plans. This framework must have a dynamic flavor: the value of the still unexecuted rest of some plan can be updated as more information becomes available during the execution of its first parts. This update leads to improving parts of the plan on-line, or to selecting another plan that appears to be more promising at the moment.

As a result, we obtain a powerful framework for the anytime generation of plans under possibly incomplete, ambiguous knowledge, and possibly including actions with alternative and context dependent effects, where the framework allows for plan reuse, incremental replanning, and incrementally integrating additional knowledge that reduces uncertainty. All this builds on the firm grounds of logic and probability theory. The framework enables a spectrum of implementations; special cases are, e.g., linear and universal plans [21]. A planner based on this model has been implemented in Standard ML. The main issues concerning this implementation as well as experimental results are reported in a longer version of this paper [24].

Section 1 describes our formalization of actions which blends our previously used formalism with Nilsson's [19] probabilistic logic. Section 2 deals with our representation of reaction plans and with its advantages for planning under uncertainty and time pressure. Section 3 explains how the quality of these plans can be evaluated and sketches how anytime algorithms can be developed within this framework. Section 4 concludes by examining some relations to aspects of previous work.

1. Formalizing Actions In Probabilistic Logic

1.1. Background (Possible Models And Probabilistic Logic)

We first provide an action formalization that handles incomplete or ambiguous information about the recent world state as well as alternative action effects, and that enables us to assess the probability of the world being in a given state after an action has been performed. To this end, we use a possible models variant of Nilsson's probabilistic logic, as a basis for both reasoning about actions in the spirit of [3], and later, decision-theoretic planning.

Given a first order language $\mathcal{L}$ of closed formulae, general information about the domain is expressed in two ways. First, a set $K \subseteq \mathcal{L}$, called *logical background knowledge*, contains the logical constraints known to be true in all world states. Second, *probabilistic background knowledge* is given as a set P of probability values for some sentences in $\mathcal{L}$. It expresses the constraints that must be verified by the probability distribution on world states believed at any time, in absence of information beyond K.

Given a finite subset $L = \{a_1, \ldots, a_n\}$ of ground atoms of $\mathcal{L}$, the world states are represented as sets $\{l_1, \ldots, l_n\}$, where $l_i = a_i$ or else $l_i = \neg a_i$. These sets are interpreted as the conjunction of their elements, and we will often switch between the set and the conjunctive notation. From all such sets, exactly those consistent with K represent *possible* states of the world; they are called *possible models*. Possible models are in fact Herbrand Models of K, restricted to L. For an arbitrary $s \in \mathcal{L}$, we define

Def. 1.1 (Possible models in s) *Let $s \in \mathcal{L}$ and let K be the logical background knowledge. The possible models in s are the elements of the set*

$Poss_K(s) = \{M = \{l_1, \ldots, l_n\} \mid K \cup M \text{ is consistent and } K \cup M \vdash s\}$

$Poss_K(true)$ (which we briefly call $Poss_K$) contains all possible state descriptions, i.e., all possible models, which are mutually exclusive and exhaustive[1]. $Poss_K(s)$ is the subset of $Poss_K$ containing all possible models that make s true. Note that, just as we interpret possible models as conjunctions, a set of possible models should be interpreted as the disjunction of its elements, i.e., as a disjunction of conjunctions.

We can adapt results from Nilsson's probabilistic logic to the above framework, in order to define the probability distribution p over the possible models space that *strictly* reflects the background knowledge. The key result using sets of possible worlds in Nilsson's work is transferred to possible models: the truth probability of a sentence is the sum of the probabilities of the possible models in this sentence. To strictly comply with K and P, p is defined as follows:

(a) A tautology has truth probability 1: $p(true) = 1 = \sum_{M \in Poss_K} p(M)$.

(b) p is subject to the constraints in P: $\forall p(s) \in P \;\; p(s) = \sum_{M \in Poss_K(s)} p(M)$.

(c) The entropy of p, i.e., $-\sum_{M \in Poss_K} p(M) \log p(M)$, is maximal, given (a) and (b).

In general, (a) and (b) still induce an infinity of probability distributions. Among them, (c) selects the p with maximal entropy, because this distribution assumes minimal additional information beyond the background knowledge.

Consider an example inspired by [5]. The task of a robot is to manipulate a cup from a fixed position, using actions to be detailed later. The cup can be either on the floor (of) or on a table (ot). When on the floor, the cup can either stand upright (up), or be tipped forward with its mouth facing the robot (fd), or be tipped backward (bk). Experiments take place outside; thus rainy weather (ry) affects the robot's performance. Assuming an appropriate definition of $\mathcal{L}$, the following background knowledge is given:

$$K = \{\mathsf{ot} \leftrightarrow \neg\mathsf{of},\ \mathsf{of} \leftrightarrow \mathsf{up} \vee \mathsf{fd} \vee \mathsf{bk},\ \mathsf{up} \rightarrow \neg\mathsf{fd} \wedge \neg\mathsf{bk},\ \mathsf{fd} \rightarrow \neg\mathsf{up} \wedge \neg\mathsf{bk},\ \mathsf{bk} \rightarrow \neg\mathsf{fd} \wedge \neg\mathsf{up}\}$$
$$P = \{p(\mathsf{ry}) = 0.4,\ p(\mathsf{fd} \vee \mathsf{bk}) = 0.7,\ p(\mathsf{ot}) = 0.2\}$$

$L = \{\mathsf{ry}, \mathsf{fd}, \mathsf{bk}, \mathsf{up}\}$ suffices to represent all relevant aspects of world states as possible models, as the truth or falsity of ot and of is determined by the atoms in L via K. These and the probability distribution p shown in (1) can be computed; lacking further knowledge, they constitute the robot's beliefs.

	$M \in Poss_K$	$p(M)$
M_1	$\{\ \mathsf{ry}, \neg\mathsf{fd}, \neg\mathsf{bk}, \neg\mathsf{up}\ \}$	0.08
M_2	$\{\ \mathsf{ry}, \mathsf{fd}, \neg\mathsf{bk}, \neg\mathsf{up}\ \}$	0.14
M_3	$\{\ \mathsf{ry}, \neg\mathsf{fd}, \mathsf{bk}, \neg\mathsf{up}\ \}$	0.14
M_4	$\{\ \mathsf{ry}, \neg\mathsf{fd}, \neg\mathsf{bk}, \mathsf{up}\ \}$	0.04
M_5	$\{\ \neg\mathsf{ry}, \neg\mathsf{fd}, \neg\mathsf{bk}, \neg\mathsf{up}\ \}$	0.12
M_6	$\{\ \neg\mathsf{ry}, \mathsf{fd}, \neg\mathsf{bk}, \neg\mathsf{up}\ \}$	0.21
M_7	$\{\ \neg\mathsf{ry}, \neg\mathsf{fd}, \mathsf{bk}, \neg\mathsf{up}\ \}$	0.21
M_8	$\{\ \neg\mathsf{ry}, \neg\mathsf{fd}, \neg\mathsf{bk}, \mathsf{up}\ \}$	0.06

(1)

In practice, the possible models and p should be computed once, at the initialization of the planning system. This can be done following the lines given in [4].

We now explain how to handle incomplete information about world states, and then define the result of an action with uncertain outcomes, applied in a state about which information is incomplete.

[1]To ensure these properties with respect not only to L but also to $\mathcal{L}$, we make the *spanningness* assumption stating that L must be sufficient for possible models to represent all relevant aspects of the world: $\forall s \in \mathcal{L} \;\; \forall M \in Poss_K(true) \;\; K \cup M \vdash s$ or else $K \cup M \vdash \neg s$.

1.2. Coping With Uncertainty About World States and Actions

Features of the current world state may be unknown at planning time. We therefore assume that information about this state is given by an arbitrary sentence $s \in \mathcal{L}$, which may describe this state only incompletely. Given that s currently holds, our belief that a possible model M represents the current state is revised. Only possible models in $Poss_K(s)$ may now correspond to the actual state, and the new probability distribution p_s over the possible models space is such that $p_s(M) = p(M \mid s)$, where $p(M \mid s)$ denotes the conditional probability of M given that s holds. This can easily be shown equivalent to

$$p_s(M) = \begin{cases} \frac{p(M)}{\sum_{M' \in Poss_K(s)} p(M')} & \text{if } M \in Poss_K(s) \\ 0 & \text{otherwise.} \end{cases}$$

For example, suppose the robot acquires the information that it is rainy and that the cup is on the table or tipped forward, i.e., $s = \mathsf{ry} \wedge (\mathsf{ot} \vee \mathsf{fd})$. Then

$$Poss_K(s) = \{\{\mathsf{ry}, \neg\mathsf{fd}, \neg\mathsf{bk}, \neg\mathsf{up}\}, \{\mathsf{ry}, \mathsf{fd}, \neg\mathsf{bk}, \neg\mathsf{up}\}\},$$

and the world is represented by the possible models M_1 or M_2 with probability

$$\begin{array}{rcccl} p_s(M_1) & = & \frac{p(M_1)}{p(M_1)+p(M_2)} & = & \frac{0.08}{0.22} \simeq 0.36 \\ p_s(M_2) & = & \frac{p(M_2)}{p(M_1)+p(M_2)} & = & \frac{0.14}{0.22} \simeq 0.64 \end{array}$$

This enforces, e.g., the conclusion that of holds with probability 0.64.

We now turn to the computation of the belief about the state resulting from the performance of an action. We allow actions to produce alternative outcomes with some probabilities, e.g., the action of tossing a coin, and actions applied in different contexts may produce differing outcomes, e.g., the action of toggling a light's switch switches the light on if it was off, and vice versa. The general form of an action a is then

$$\begin{array}{rcll} a & = & [\; pre_1 & \mid \; (Post_1^1, \pi_1^1), \ldots, (Post_1^{l(1)}, \pi_1^{l(1)}); \\ & & \vdots & \\ & & pre_m & \mid \; (Post_m^1, \pi_m^1), \ldots, (Post_m^{l(m)}, \pi_m^{l(m)}) \;]. \end{array} \tag{2}$$

For each context i, the precondition pre_i is an arbitrary formula from $\mathcal{L}$, the postconditions $Post_i^j$ are subsets of possible models, and π_i^j is the probability that executing the action in the context i leads to $Post_i^j$. We assume the pre_i are mutually exclusive and exhaustive, so that, when the action is applied, the unique context whose precondition holds determines the possible outcomes. We furthermore assume that, for each context, the postconditions are exhaustive and mutually exclusive; the meaning of the latter will be discussed later.

Consider the action *table2up* for moving the cup from the table to its upright position on the floor. If the weather is fine, this succeeds 80% of the time; otherwise the cup falls to its tipped forward position. When it is rainy, the cup gets slippery, decreasing the success probability to 60%. To ensure the exhaustivity of the preconditions, a default context with the empty set as postcondition captures the intuition that the action changes nothing when it is unapplicable, i.e., when ot does not hold.

$$\begin{array}{rcrll} \mathit{table2up} & = & [\; \neg\mathsf{ry} \wedge \mathsf{ot} & \mid & (\{\mathsf{up}\}, 0.8), \; (\{\mathsf{fd}\}, 0.2); \\ & & \mathsf{ry} \wedge \mathsf{ot} & \mid & (\{\mathsf{up}\}, 0.6), \; (\{\mathsf{fd}\}, 0.4); \\ & & \neg\mathsf{ot} & \mid & (\{\}, 1) \qquad\qquad\qquad\quad] \end{array}$$

Under some assumptions about L discussed in [3], the approach solves both the frame and ramification problem; it is unnecessary to specify that the weather is unaffected and that the cup is not on the table any more. Unspecified features are inferred via K, capturing the intuition that a possible model M' that results from a possible model M by applying an action that makes postcondition $Post$ true, contains $Post$ but differs as little as possible from M. M' is said to be *maximally Post-conforming with* M:

Def. 1.2 (Maximal *Post*-conformity) *Let M and M' be elements of $Poss_K$, and $Post$ be an action postcondition. M' is maximally Post-conforming with M iff $M \cap M'$ is (set inclusion) maximal under the constraint that $Post \subseteq M'$. The set of all such models M' is noted $C_K(Post, M)$.*

In our example, we have $C_K(\{\mathsf{fd}\}, M_1) = \{M_2\}$. In general, there may be multiple maximally conforming models, since a postcondition can be achieved by several minimal changes in the world. Recalling that we interpret sets of possible models as disjunctions, we define the probability of M' resulting from the achievement of $Post$ from M as $p(M' \mid C_K(Post, M))$. $C_K(Post, M)$ is then considered as the information available about the resulting state, and the probability is computed as explained in Section 2.2.

Thus elements of $C_K(Post, M)$ are the possible models resulting from the achievement of a unique postcondition $Post$ starting from a unique possible model M. From this, we can define the result of applying an action in a state about which information is given as an arbitrary formula.

Def. 1.3 (Result of an action) *Let $s \in \mathcal{L}$, and a be an action as defined in (2). The possible models resulting from the application of a in a state where s holds are the elements of the following set*
$R_K(a, s) = \{M' \in Poss_K \mid \exists M \in Poss_K(s)$ such that $M' \in C_K(Post_i^j, M)$, where $K \cup M \vdash pre_i$ and $j \in \{1, \ldots, l(i)\}\}$.

For example, $R_K(table2up, s) = \{\{\mathsf{ry}, \neg\mathsf{fd}, \neg\mathsf{bk}, \mathsf{up}\}, \{\mathsf{ry}, \mathsf{fd}, \neg\mathsf{bk}, \neg\mathsf{up}\}\}$ is the result of applying $table2up$ in state $s = \mathsf{ry} \wedge (\mathsf{ot} \vee \mathsf{fd})$: For M_1 in $Poss_K(s)$, the second context is selected, whose postconditions lead respectively to M_4 and M_2 shown in (1); for M_2 in $Poss_K(s)$ the default context is selected, which means that nothing changes.

Our belief about s and the probabilities of the outcomes of a enable us to compute the probability distribution $p_{(a,s)}$ over the possible models that result from performing a in s. If $p_a(M', M)$ denotes the probability that a changes the world from possible model M to M', then clearly

$$p_{(a,s)}(M') = \textstyle\sum_{M \in Poss_K(s)} p_a(M', M)\ p_s(M).$$

But how is $p_a(M', M)$ calculated? Given a possible model M and the context i whose precondition holds in M, we assume that for any two postconditions $Post_i^{j_1}$ and $Post_i^{j_2}$, we have $C_K(Post_i^{j_1}, M) \cap C_K(Post_i^{j_2}, M) = \emptyset$. This is our mutual exclusivity assumption on postconditions, whose explanation was previously postponed. If this property holds, then the probability that executing a in M leads to M', where $K \cup M \vdash pre_i$, is

$$p_a(M', M) = \textstyle\sum_{j=1}^{l(i)} p(M' \mid C_K(Post_i^j, M))\ \pi_i^j.$$

When applying, e.g., $table2up$, the maximally conforming models are unique, and our belief that a possible model results from applying $table2up$ in s is

$$\begin{aligned} p_{(table2up,s)}(M_4) &= (1 \times \pi_2^1 + 0 \times \pi_2^2) \times p_s(M_1) + 0 \times \pi_3^1 \times p_s(M_2) \\ &\simeq 0.6 \times 0.36 \simeq 0.22, \\ p_{(table2up,s)}(M_2) &= (0 \times \pi_2^1 + 1 \times \pi_2^2) \times p_s(M_1) + 1 \times \pi_3^1 \times p_s(M_2) \\ &\simeq 0.4 \times 0.36 + 1 \times 0.64 \simeq 0.78. \end{aligned}$$

In this section, we have exemplified that our action formalization copes with uncertainty at planning time, such as incompleteness or ambiguity in world state descriptions; it also copes with ambiguity and context dependency of actions effects. [3, 22] describe the formalization in detail. Furthermore, probability assessments, once combined with utility functions, will constitute a preference ordering which will enable us to choose among plan alternatives under time pressure. We now explain how this action formalization can be used to generate and improve reaction plans in an anytime fashion.

2. Structured Reaction Plans

We now show how to represent plans that are, first, reactive to the sources of uncertainty predicted by the action formalization, that can, second, be generated from first principles if time permits, and that, third, can be easily reused or incrementally extended if time is tight. We do not yet exploit probabilistic information or deal with anytime planning issues; this will be done in the next section. In fact, we just summarize key issues from [23] here.

A *plan* is a bipartite directed graph with two types of nodes: T-nodes representing tasks, i.e., occurrences of actions in a plan, and M-nodes representing possible models. This is to be interpreted as follows: a T-node T preceded by some M-node M means that the plan specifies that T is to be applied whenever the plan execution finds itself in a situation described by M; M preceded by T represents the possible model M that may result as one of the effects from applying T.

Given a *planning problem* defined by an initial situation s, a goal formula g, a background knowledge K and P, and a set of actions A, the root of a plan for this problem is a task built from the dummy action $start = [\ true \mid (M_1, p_s(M_1)), \ldots, (M_n, p_s(M_n))]$ such that $\{M_1, \ldots, M_n\} = Poss_K(s)$. By construction, the successors of *start* are all M-nodes in $Poss_K(s)$. The leaves of a plan are M-nodes, which represent the possible world states at the end of executing this plan. Some of them might match the goal (M matches g iff $M \in Poss_K(g)$), but since the planner may not have generated a complete plan for all alternatives, there is no requirement that every leaf match the goal. Each non-leaf M-node M in the plan must not match g, and must have one unique T-node successor corresponding to an action $a \in A$. The successors of this T-node, in turn, are all possible models in $R_K(a, M)$.

As a last property of plans, we consider their *validity:* from each node, there must be at least one path to some leaf. As an informal lemma, note that a valid plan cannot include a task that is not applicable in the state represented by its possible model predecessor. The reason is that an intuitively non-applicable task changes nothing and would create an blind alley in the plan.

Here is an example. Starting from our initial situation $s = \mathsf{ry} \wedge (\mathsf{ot} \vee \mathsf{fd})$, we want to achieve the goal $g = \mathsf{up}$. Available actions are *table2up* as previously described, as well as *back2up* for moving the cup from its bk to its up position, and *spin* for spinning a tipped cup, shown in Figure 1. The example plan $\mathcal{P}_1$ for this problem is also shown in Figure 1. $\mathcal{P}_1$ is to be interpreted as follows: if the cup is initially fd, then *spin* it until the desired bk position is obtained and apply *back2up*; if the cup is initially ot, apply *table2up*, and if the cup becomes fd, then go on as before. Note that $\mathcal{P}_1$'s single leaf matches g; therefore, it is guaranteed to achieve the goal under the sources of uncertainty predicted by the action formalization, where "guaranteed" means that the probability of being in the goal model approaches 1 as the length of

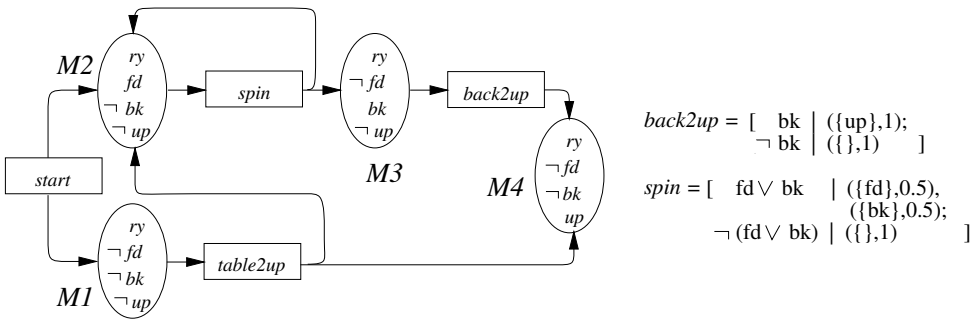


Figure 1: Plan $\mathcal{P}_1$ for the cup example, and the *spin* and *back2up* actions

execution sequences grows. The requirement that a non-leaf M-node has exactly one T-node successor ensures that the execution is deterministic. On the other hand, it enforces that only analogs of linear plans can be represented.

Setting aside, for the moment, how this plan representation could benefit from probability information, the very structure of the plans yields some interesting features for planning under uncertainty and time pressure.

First, there is more support for *execution monitoring,* than in other approaches to encoding reactivity such as situated control rules (SCRs) rules. A plan can obviously be translated into a set of SCRs, by interpreting M-nodes as IF parts, and the T-nodes as THEN parts. But our plans allow us to focus more easily on the momentarily relevant rules by representing explicitly what is expected to happen next, according to the domain model. They also exclude what would be conflicts between different applicable rules, allowing us to choose the next execution step deterministically. Moreover, our plans have the same capabilities as SCRs for handling unpredictable or unpredicted events at execution. E.g., suppose that when applying *table2up*, the cup serendipitously drops to the bk position on the floor. The execution monitor can locate the corresponding unexpected M-node M_3 in $\mathcal{P}_1$, and resume the execution with *back2up*. A useful feature of SCRs that is not yet offered by our plan representation is that SCRs explicitly define the weakest preconditions ensuring that the goal can still be reached. Hence, they offer economic sensing, and also situation generality: they might be applicable to more situations than originally discovered by the planner.

Second, *replanning* with this plan representation can inherit from previous planning, making *incremental planning* possible. This helps mainly in two cases: when the execution monitor is confronted with an unexpected situation for which the plan includes no corresponding M-node, and when time did not suffice for generating a plan that is complete according to the domain model. To exemplify the latter case, suppose no reaction has yet been planned for the case where M_1 represents the initial world state. Hence, M_1 is a second leaf of the plan, and *table2up* is not included yet. Incremental planning in reaction to M_1 would then simply consist in inserting *table2up* after M_1 and connecting it to its possible models successors M_2 and M_4, thereby profiting from a whole part of what is already generated. If replanning is necessary due to an unexpected event, it suffices to insert a new M-node representing the unexpected situation as a successor of the *start* task, and to proceed as for the incremental planning case.

Finally, these plans can be *reused as a default behavior* if time to generate optimal ones from first principles is lacking. A plan achieving a goal g (i.e., a plan whose leaves all match g) can be reused to achieve h if $Poss_K(g) \subseteq Poss_K(h)$. E.g., $\mathcal{P}_1$ can be reused to achieve of. However, reusing does not require the plan to be finished with

respect to a new problem. There may be possible models of the initial situation of the new problem which are not included in the M-nodes of a reused plan, and for which one must replan. In general, solutions to problem instances are not simply found in plan libraries. But even then, parts of old plans that are "useful" for a new problem can still be extracted. A detailed discussion, however, lies beyond this paper's scope.

In sum, our plan representation shows several interesting features for planning under uncertainty and time pressure. The plan representation combined with the action formalization enables us to build reactive planners that can, like PASCALE, incrementally increase their reaction quality. However, the representation itself does not provide a way of chosing purposefully among plan alternatives, and an anytime planning algorithm using it would have no information about to which events it should rationally plan a reaction first. The topic of the next section is how to provide this information.

3. The Quality of Reaction Plans

As introduced in Section 2, we have information about the probability of a possible model representing an initial situation or resulting as the effect of an action. We will now exploit this probability information to develop a framework that allows us to define quality measures on plans. The framework is based on Markov chain theory; it provides dynamic quality measures that can be used as input by existing anytime algorithms. Furthermore, it allows one to select the parts of an unfinished plan to be extended first, thereby constituting a basis for designing special-purpose anytime algorithms.

3.1. Basic Results About Markov Chains

We first recall basic results from Markov chain theory [18], and then explain how these results can be used to define the quality of a plan. Markov chains are stochastic processes used to describe dynamic systems whose probabilistic transition through a set of states at a given instant $t+1$ depends only on the state at the immediate preceding instant t, and not on the states the chain passed through prior to time t. Furthermore, if the transition probabilities do not depend on t (i.e., remain stationary over time), the Markov chain is said to be *stationary*. Stationary chains can by definition be represented by a single transition matrix relating the probability of the succeeding state to the current state. More formally:

Def. 3.1 ((Stationary) Markov chain, transition matrix) *A Markov chain is a family of random variables* $\{X_t, t = 0, 1, \ldots\}$ *taking values in a set of states* S*, such that the conditional probability distribution* Π *of the state at time* $t+1$ *verifies for all* t*:*

$$\forall s_0, \ldots, s_{t+1} \in S.\ [\Pi(X_{t+1} = s_{t+1} \mid X_t = s_t, \ldots, X_0 = s_0) = \Pi(X_{t+1} = s_{t+1} \mid X_t = s_t)]$$

A Markov chain is stationary if and only if for all t*:*

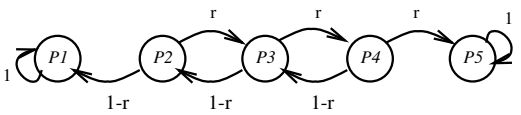
$$\forall s_i, s_j \in S.\ [\Pi(X_{t+1} = s_j \mid X_t = s_i) = m_{ij}]$$

The transition matrix for a stationary Markov chain is the matrix with entries m_{ij}*.*

A classical example of a process that can be modeled as a stationary Markov chain is a random walk. A particle moves between 5 points $p_1, \ldots, p_5$ on a line. At each step, it can go from one point to the right with probability r, and from one point to the left

with probability $1-r$. It moves until it reaches one of the two boundaries of the line and remains at this boundary. The corresponding stationary chain and its transition matrix are shown below.

$$\begin{array}{c|cccc|cc} & p_2 & p_3 & p_4 & p_1 & p_5 \\ p_2 & 0 & r & 0 & 1-r & 0 \\ p_3 & 1-r & 0 & r & 0 & 0 \\ p_4 & 0 & 1-r & 0 & 0 & r \\ \hline p_1 & 0 & 0 & 0 & 1 & 0 \\ p_5 & 0 & 0 & 0 & 0 & 1 \end{array}$$



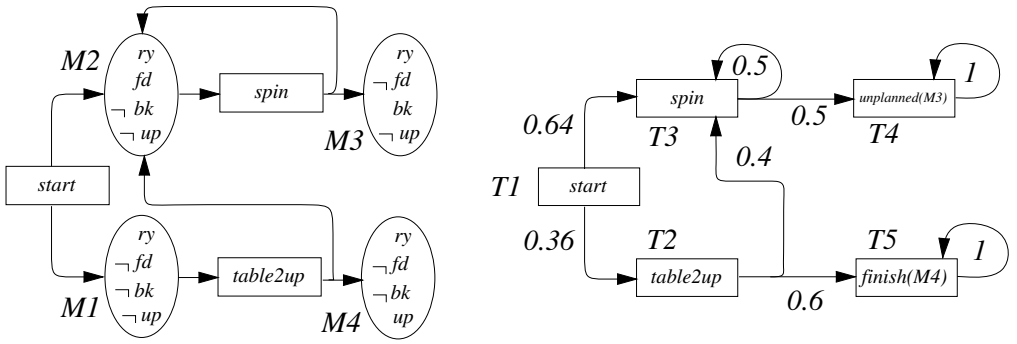
We are interested here in a special type of stationary chain called *absorbing* chain. This is a stationary chain with two types of states: transient states, which can be left on at least one path that never return, and absorbing states, which cannot be left once entered. It is easy to see that the chain modeling the random walk is absorbing. p_1 and p_5 are absorbing states, while the others are transient. The transition matrix of an absorbing chain can be divided into 4 submatrices as shown below, where the submatrix Q denotes transitions from transient to transient states, R denotes transitions from transient to absorbing states, I is the identity matrix, and O consists only of 0's.

$$\begin{array}{c|cc} & trans. & abs. \\ trans. & Q & R \\ \hline abs. & O & I \end{array}$$

These submatrices can be used to compute quantitative information about the process modeled as an absorbing chain. The matrix $I-Q$ always has an inverse N, called the *fundamental matrix,* where $N=(I-Q)^{-1}=\sum_{k=0}^{\infty} Q^k$. The definition of N implies that its ij^{th} element is the average number of times transient state j will be entered before an absorbing state is reached, given that we are in transient state i. Furthermore, the ij^{th} element of matrix $N \times R$ is the probability of reaching absorbing state j, given that we are in transient state i. Note that this probability of reaching an absorbing state can also be viewed as the average number of times an absorbing state will be entered before the process becomes stable. In the following, we will characterize an absorbing Markov chain by the matrix $(N \quad N \times R)$ whose leftmost columns are those of the fundamental matrix N of the chain, and whose rightmost columns are those of the product of N by the submatrix R of the chain.

3.2. Plan Quality

The starting point of our use of Markov chain theory for the definition of the quality of a plan is to consider a plan as an absorbing Markov process. We associate a plan with an absorbing Markov chain whose state set is a set of tasks. Transient states of the chain correspond to the tasks in the plan. Its absorbing states shall denote that the plan execution is finished, i.e., when the current world situation is represented by an M-node leaf of the plan, then the state of the execution remains as it is. Therefore, we artificially introduce two types of absorbing Markov states: *unplanned* states are dummy tasks applied in final M-nodes that do not match the goal, and *finish* states are other dummy tasks applied in M-nodes that do match the goal. The probability law Π of the Markov chain follows the probabilities provided by the action formalization.



	T_1	T_2	T_3	T_4	T_5
T_1	0	0.36	0.64	0	0
T_2	0	0	0.4	0	0.6
T_3	0	0	0.5	0.5	0
T_4	0	0	0	1	0
T_5	0	0	0	0	1

Figure 2: Incomplete plan $\mathcal{P}_2$ and its associated Markov chain with its transition matrix

Def. 3.2 (Markov chain associated with a plan) *Let s, g, K, P, A define a planning problem; let $\mathcal{P}$ be a plan for this problem, characterized by its set of T-nodes $\mathcal{T}$, its set of non-leaf M-nodes $\mathcal{M}$, its set of leaf M-nodes $\mathcal{M}_L$, and the function $pre_{\mathcal{P}}$ mapping a T-node to its M-node predecessor in $\mathcal{P}$. Let, moreover, $\mathcal{T}' = \{unplanned(M) \mid M \in \mathcal{M}_L \setminus Poss_K(g)\} \cup \{finish(M) \mid M \in \mathcal{M}_L \cap Poss_K(g)\}$, and let the function pre over $\mathcal{T} \cup \mathcal{T}'$ such that*

$$pre(T) = \begin{cases} pre_{\mathcal{P}}(T) & for\ T \in \mathcal{T} \\ M & for\ T = unplanned(M) \in \mathcal{T}' \\ M & for\ T = finish(M) \in \mathcal{T}' \end{cases}$$

The Markov chain associated with $\mathcal{P}$, noted $chain(\mathcal{P})$, is the family $\{X_t, t = 0, 1, \ldots\}$ of random variables ranging over the set of tasks $\mathcal{T} \cup \mathcal{T}'$, such that the conditional probability distribution Π of X_{t+1} is defined as

$$\Pi(X_{t+1} = T' \mid X_t = T) = \begin{cases} p_s(pre(T')) & for\ T = start \\ p_{(T,pre(T))}(pre(T')) & for\ T \notin \mathcal{T}' \cup \{start\} \\ 1 & for\ T \in \mathcal{T}'\ and\ T' = T \\ 0 & for\ T \in \mathcal{T}'\ and\ T' \neq T \end{cases}$$

Figure 2 shows the incomplete cup domain plan $\mathcal{P}_2$, $chain(\mathcal{P}_2)$, and its transition matrix.

Translating the above-reported results of Markov chain theory to our framework, we find that for any plan $\mathcal{P}$, the ij^{th} element of the matrix ($N\ N \times R$) characterizing $chain(\mathcal{P})$ represents the average number of times task j will be executed before the plan execution ends, given that we are currently executing task i. Note that, for j being an *unplanned* or $finish$ task, this also represents the probability that plan execution ends in task j, given that i is currently executed.

These results allow us to estimate the quality (or utility) of a plan prior to execution, and to update this estimation during execution, according to the actual evolution of the environment. We assume that each task in the plan is given a numerical *utility* which will mostly depend on the cost of the action from which the task is built and on the reward for being in its possible model predecessor. E.g., if utility is understood

as goal-achievement probability, a step utility function [14] should be used, that maps the *finish* tasks to 1 and other tasks to 0. If partial goal-satisfaction is of interest, one can use a noisy step function assigning the highest value to *finish* tasks, some lower positive values to *unplanned* tasks that reflect the proximity to the goal of their possible model predecessor, and 0 to the other tasks. If interested in minimizing the cost of plan execution, one can use a utility function assigning the cost (a negative value) of their corresponding action to the respective tasks, where the cost of an *unplanned* task would heuristically depend on its possible model predecessor. Setting aside the problem of building a multi-attribute utility function from individual attributes utilities, which is dealt with in [26], the plan quality is defined from the utility of tasks as follows.

Def. 3.3 (Plan Quality) *Let $\mathcal{P}$ be a plan, and $(N\ N \times R)$ the matrix characterizing chain($\mathcal{P}$). Let U be a column vector such that U_j is the utility associated with task j. The quality of $\mathcal{P}$, given that task i is executed, is the i^{th} element of the vector $U(\mathcal{P}) = (N\ N \times R) \times U$.*

Hence, $U(\mathcal{P})$ yields an a-priori estimation of $\mathcal{P}$'s quality by considering its element corresponding to the *start* task, as well as updates of this estimation, given the task currently under execution. For the simple case where quality is defined as goal-achievement probability, $U(\mathcal{P}_2)$ is calculated as follows, where N and R are calculated from Π as explained in Section 3.1, and where Π is itself calculated from the probabilities provided by the action formalization, as given in Definition 3.2:

$$\begin{array}{c} \\ T_1 \\ T_2 \\ T_3 \end{array} \begin{array}{c} \begin{array}{ccccc} T_1 & T_2 & T_3 & T_4 & T_5 \end{array} \\ \left(\begin{array}{ccccc} 1 & 0.36 & 1.56 & 0.78 & 0.22 \\ 0 & 1 & 0.8 & 0.4 & 0.6 \\ \underbrace{0 \quad 0 \quad 2}_{N} & & & \underbrace{1 \quad 0}_{N \times R} & \end{array} \right) \end{array} \times \begin{array}{c} \\ \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \end{array} \right) \\ \underbrace{\quad}_{U} \end{array} = \begin{array}{c} \\ \left(\begin{array}{c} 0.22 \\ 0.6 \\ 0 \end{array} \right) \\ \underbrace{\quad}_{U(\mathcal{P}_2)} \end{array}$$

The a-priori estimation of the quality is 22%. During execution, additional knowledge about the task currently executed becomes available, reducing uncertainty about what may happen during the rest of this execution. E.g., if we know that the task currently executed is T_2, i.e., *table2up*, then $\mathcal{P}_2$'s quality estimation increases to 60%. When $\mathcal{P}_2$ gets extended, e.g., by introducing *back2up* between $M3$ and $M4$, which leads to $\mathcal{P}_1$, the matrices and the quality estimation must be recalculated. Computing quality has complexity $\mathcal{O}(n^3)$, n being the number of tasks in the plan.

3.3. Use and Design of Anytime Algorithms

The a-priori quality estimation enables off-line planning using a general-purpose anytime algorithm, such as those based on expectation driven iterative refinement, e.g., [25]. The updated estimations are suitable to incrementally improve an incomplete plan during its execution, using the same general-purpose algorithms. The planner can interact with the execution monitor, and work with the updated estimation corresponding to the task currently executed. This implicitly focuses the anytime algorithm on the plan part whose improvement will be of most use in the rest of the execution.

The framework also suggests a rational exploration of both state and search space, thereby facilitating the design of special-purpose anytime algorithms for off-line or on-line planning. The following algorithm, which can be viewed as a reformulation of the projection algorithm in [10] without considering quantitative time, plans for the most

probable evolution of the environment first. Given an incomplete plan to be improved on-line, the state space is explored by selecting the non-goal leaf that is maximally probable to be reached, given the task currently executed. This leaf can be computed according to the recent matrix $N \times R$.[2] Once this leaf is expanded, its most probable successor is further expanded, until the currently expanded path reaches a goal, which leads then to a new selection. To focus on useful improvements with respect to the remainder of the execution, the selection process is re-performed whenever a new task gets executed. To select among plans resulting from an expansion, the search space is explored using an interruptible best-first search informed by the quality estimation, and that guarantees quality to increase monotonically as a function of computation time. Hence, when averaged over an infinite number of runs, the reward obtained during the execution of a plan is higher than if we were to execute a plan generated for a shorter computation time. [24] presents this algorithm in more detail.

Apart from on-line plan generation, the ability to update the quality estimation has an advantage for a reactive executor alone: if many reaction plans are available for a problem, one can start execution with the a-priori best plan but interrupt its execution and switch to another plan, whenever this one becomes better. [17] uses this approach: the planner continually selects the behavior with maximal utility among those available.

The model presented here aims at a combination anytime planner/reactive executor. As in [11], it can sometimes be assumed that a reactive executor can be designed that uses user-provided default plans, thereby having some probability of solving a problem. The role of the anytime planner is to increase this competence. If the planner was fast enough to generate a complete plan for the problem, then the reactive executor can use it to achieve the goal. If no plan at all could be generated by the planner, then the reactive executor uses its default behavior. The interesting case is the intermediate one where only a partial plan could be generated: the reactor starts by executing this plan, and ends the execution with its default behavior. This case leads to an important problem: since the partial plan may not be a prefix of any solution, it might decrease the executor's performance. [11] presents the RFS type of search (Reaction-First Search) which is designed to solve this problem. It releases partial plans, that will, on average, monotonically increase the executor's performance. Our model provides another way of dealing with such a combination planner/reactive executor. It suffices that the planner uses a quality measure based on a utility function on tasks that maps $finish$ tasks to 1, $unplanned(M)$ tasks to the goal-achievement probability of the best default plan available for solving the problem starting from M, and other tasks to 0. RFS does not need to calculate goal-achievement probabilities in order to guaranty monotonic improvement, but on the other hand, the method sketched here can trivially be extended to an arbitrary utility function.

4. Instances of the Framework

In this paper, we view planning as a choice under uncertainty and time pressure, for which symbolic planning and decision theory serve two complementary purposes. As [15] points out, *"Symbolic planning provides a computational theory of plan generation ... Decision theory provides a normative model of choice under uncertainty"*.

[2] Given that we are executing task T (which can be the *start* task for off-line planning), the non-goal leaf M that is maximally probable to be reached is that, for which the element of $N \times R$ corresponding to T and $unplanned(M)$ is maximal.

Within our model, symbolic planning enables to search for a plan under uncertainty that stems from incomplete information about the start situation, from context dependency of actions, and from alternative action effects. We did not go into detail concerning these issues as they are presented elsewhere [23].

The new issue is to introduce probabilities in order to both guide the search and chose among feasible plan alternatives. These probabilities are handled like in Nilsson's probabilistic logic for reasoning within single world situations; we use Markov chain theory to calculate the probability of tasks in a plan being executed and, eventually, final states of a plan being reached, given that the plan execution has proceeded in some direction. Based on the notion of utility of tasks in a plan being executed, we have then defined plan quality, which allows us to estimate where to extend a given incomplete plan with most effect on the plan quality. This is crucial for a rational anytime planning algorithm; and it allows for rationally jumping to another plan at execution time if the quality of the plan currently under execution turns out to be lower than the estimated quality of some other suitable off-the-shelf plan.

Recently, methods based on the policy iteration algorithm have been proposed which do not need to explore the whole state space at once, but restrict the planner's attention to judiciously chosen larger and larger sets of situations [7]. This approach uses deliberation scheduling methods in order to allocate computation time to various iterative refinement routines. However, it employs a low level representation of the state space that must be entirely described as a stochastic automaton. Our framework differs from this latter by laying stress on flexible symbolic representations. Indeed, the very focus of our paper is on the representational issue, not on anytime planning and deliberation scheduling themselves: how to represent actions and plans in order to exploit anytime planning, deliberation scheduling, and decision-theory. It would then be interesting to see how the above techniques can be used in the framework of our representation.

An academic advantage of our model is its high expressiveness, which allows a variety of seemingly different plan formats to be generated or reformulated in its terms and thus made comparable. As examples, let us mention linear (STRIPS type) plans, universal plans [21], decision trees [12], situated control rules [10], and PASCALE plans [23].

A non-instance of our framework is, e.g., the planner in [17], whose *causal model* of the world is more expressive than ours. It explicitly considers changes in the world that do not need to be caused by single actions of the planner. However, optimal plan *selection* with respect to the current state of knowledge is NP-hard with this model. The planner uses a revolving approach, trading plan quality for computation time. It suffers from a limited horizon problem, and hence cannot make long-term predictions.

Practically, expressiveness is dangerous, because it yields computation cost. Thus, if your domain is completely deterministic and completely known, we recommend that you apply something simpler than what we propose here. On the other hand, if your domain is very uncertain, then it would be nonsense to calculate a plan to the tiniest detail. For such cases, algorithms with a rational anytime behavior seem most promising.

Acknowledgements

Thanks to Gerd Brewka, Marie-Odile Cordier, Tom Gordon, Gerd Grosse, Eric Jacopin, Jürgen Paulokat, Eric Rutten, Josef Schneeberger, and Qiang Yang for commenting on drafts.

References

[1] M. Beetz and D. McDermott. Declarative goals in reactive plans. In *[16]*, pages 3–12, 1992.

[2] M. Boddy and T.L. Dean. Solving time-dependent planning problems. In *Proc. IJCAI-89*, pages 979–984, 1989.

[3] G. Brewka and J. Hertzberg. How to do things with worlds: on formalizing actions and plans. *J. Logic and Computation*, to appear. Previous version as TASSO-Report No. 11, GMD, 1990.

[4] P. Cheeseman. A method of computing generalized bayesian probability values for expert systems. In *Proc. IJCAI-83*, pages 198–202, 1983.

[5] L. Chrisman and R. Simmons. Sensible planning: focusing perceptual attention. In *Proc. AAAI-91*, pages 756–761, 1991.

[6] M.O. Cordier and P. Siegel. A temporal revision model for reasoning about world change. In *Proc. KR-92*, pages 732–739, 1992.

[7] T. Dean, L. P. Kaelbling, J. Kirman, and A. Nicholson. Planning under time constraints in stochastic domains. In *Proc. AAAI-93*, pages 574–579, 1993.

[8] T.L. Dean and M. Boddy. An analysis of time-dependent planning. In *Proc. AAAI-88*, pages 49–54, 1988.

[9] T.L. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Comput. Intell.*, 5(3):142– 150, 1988.

[10] M. Drummond and J. Bresina. Anytime synthetic projection: maximizing the probability of goal satisfaction. In *Proc. AAAI-90*, pages 138–144, 1990.

[11] M. Drummond, K. Swanson, J. Bresina, and R. Levinson. Reaction-first search. In *Proc. IJCAI-93*, pages 1408–1414, 1993.

[12] J.A. Feldman and R.F. Sproull. Decision theory and artificial intelligence II: the hungry monkey. *Cogn. Sci.*, 1:158–192, 1977.

[13] R.E. Fikes, P.E. Hart, and N.J. Nilsson. Learning and executing generalized robot plans. *J. Art. Intell.*, 3:251–288, 1972.

[14] P. Haddawy and S. Hanks. Issues in decision-theoretic planning: symbolic goals and numeric utilities. In K.P. Sycara, editor, *Proc. DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*, pages 48–58, San Diego, CA, 1990.

[15] P. Haddawy and S. Hanks. Representations for decision-theoretic planning: utility functions for deadline goals. In *Proc. KR-92*, pages 71–82, 1992.

[16] J. Hendler, editor. *Artificial Intelligence Planning Systems: Proceedings of the First International Conference (AIPS92)*, Morgan Kaufmann, 1992.

[17] K. Kanazawa and T.L. Dean. A model for projection and action. In *Proc. IJCAI-89*, pages 985–990, 1989.

[18] J. Kemeny and L. Snell. *Finite Markov Chains*. Van Nostrand, 1960.

[19] N.J. Nilsson. Probabilistic logic. *J. Art. Intell.*, 28(1):71–87, 1986.

[20] M.J. Schoppers. In defense of reaction plans as caches. *AI Magazine*, 10(4 (Winter)):51–60, 1989.

[21] M.J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proc. IJCAI-87*, 1987.

[22] S. Thiébaux. *Anytime Reaction Planning in Probabilistic Logic*. Master's thesis, Florida Institute of Technology, June 1992.

[23] S. Thiébaux and J. Hertzberg. A semi-reactive planner based on a possible models action formalization. In *[16]*, pages 228–235, 1992.

[24] S. Thiébaux, J. Hertzberg, W. Shoaff, and M. Schneider. A stochastic model of actions and plans for anytime planning under uncertainty. *Int. J. of Intelligent Systems*, 1993. in press.

[25] B.W. Wah and L.C. Chu. *TCA**. a time constrained approximate A^* search algorithm. In *Proc. 1990 IEEE International Workshop on Tools for Artificial Intelligence*, pages 314–320, 1990.

[26] M.P. Wellman and J. Doyle. Modular utility representation for decision-theoretic planning. In *[16]*, pages 236–242, 1992.

[27] D. Wilkins. *Practical Planning. Extending the Classical AI Planning Paradigm*. Morgan Kaufmann, 1988.

[28] S. Zilberstein and S.J. Russel. Efficient ressource-bounded reasoning in AT-RALPH. In *[16]*, pages 260–266, 1992.