A Stochastic Model of Actions and Plans for Anytime Planning under Uncertainty[1]

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

Joachim Hertzberg[2]
Dortmund University
Informatik VIII
44221 Dortmund, Germany
hertz@ls8.informatik.uni-dortmund.de

William Shoaff
FIT
CS Department
Melbourne, Fl 32901, USA
wds@cs.fit.edu

Moti Schneider
FIT
CS Department
Melbourne, Fl 32901, USA
moti@cs.fit.edu

[1]To appear in *Int. J. of Intelligent Systems.* Thanks to Gerd Brewka, Marie-Odile Cordier, Tom Gordon, Gerd Grosse, Eric Jacopin, Jürgen Paulokat, Eric Rutten, Josef Schneeberger, Qiang Yang, and the anonymous referees for commenting on a previous version of this paper. A shorter version of this paper has appeared in [Horz, 1993].

[2]On leave from GMD, I3.KI, Schloss Birlinghoven, 53757 Sankt Augustin, Germany.

Abstract

Building planning systems that operate in real domains requires coping with both uncertainty and time pressure. This paper describes a model of reaction plans, which are generated using a formalization of actions and of state descriptions in probabilistic logic, as a basis for anytime planning under uncertainty.

The model has the following main features. At the *action* level, we handle incomplete and ambiguous domain information, and reason about alternative action effects whose probabilities are given. On this basis, we generate reaction *plans* that specify different courses of action, reflecting the domain uncertainty and alternative action effects; if generation time was insufficient, these plans may be left unfinished, but they can be reused, incrementally improved, and finished later. At the *planning* level, we develop a framework for measuring the quality of plans that takes domain uncertainty and probabilistic information into account using Markov chain theory; based on this framework, one can design anytime algorithms focusing on those parts of an unfinished plan first, whose completion promises the most "gain". Finally, the plan quality can be updated during execution, according to additional information acquired, and can therefore be used for on-line planning.

1 Motivation

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. In general, knowledge about the planning domain is incomplete, or the domain may not be completely controlled by the planner or actions can have nondeterministic or context-dependent effects, to name just some forms of uncertainty. There is a variety of approaches for tackling some or many forms of uncertainty at planning or execution time; examples include reaction plans and reactive planning (see [Schoppers, 1989] for a discussion and references), conditional planning [Warren, 1976; Peot and Smith, 1992], decision-theoretic planning methods [Feldman and Sproull, 1977; Haddawy and Hanks, 1990], or execution monitoring and replanning [Fikes *et al.*, 1972; Wilkins, 1988, Ch. 11]. Moreover, uncertainty is also tackled by theory-oriented work in reasoning about action and change, e.g. [Dean and Kanazawa, 1988; Brewka and Hertzberg, To appear; Cordier and Siegel, 1992], with some texts, e.g. [Hanks, 1990; Dean and Wellman, 1991], explicitly relating theoretical concepts of reasoning about action to planning.

Implementing time-bounded rationality. Generating optimal plans from first principles takes time. Under time pressure, a planner can produce some plan in a fixed amount of time, and improve the plan quality as more time is allocated. Anytime algorithms [Dean and Boddy, 1988] can be used to implement time-bounded rationality in the planning framework. They have been applied to solving time-dependent planning problems [Boddy and Dean, 1989] and are also used for a broad class of planning applications, e.g. [Zilberstein and Russel, 1992].

Obviously, 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. Examples of such works are [Drummond and Bresina, 1990; Beetz and McDermott, 1992].

In a previous paper [Thiébaux and Hertzberg, 1992], we have described PASCALE, a system for planning under uncertainty. This system is based on a particular theory of actions with uncertain outcomes [Brewka and Hertzberg, To appear], 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.

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 re-planning, and incrementally integrating additional knowledge that reduces uncertainty. All this builds on the firm grounds of logic and probability theory. Note that the framework does not presuppose that all the forms of uncertainty actually occur in every planning application. It enables a whole spectrum of implementations; special cases being, e.g., classical linear plans and universal plans [Schoppers, 1987].

The paper is based on [Thiébaux, 1992], which contains additional details. It is organized as follows. Section 2 describes our formalization of actions which blends our previously used formalism with Nilsson's [1986] probabilistic logic. Section 3 deals with the representation of reaction plans and with its advantages for planning under uncertainty and time pressure. Section 4 explains how the quality of these plans can be evaluated, building on Markov chain theory, and sketches how anytime algorithms can be developed within the framework. Section 5 discusses the implementation of a system based on our model, and describes experimental results. Section 6 concludes by examining some relations to previous work.

2 Formalizing Actions In Probabilistic Logic

This section describes our action formalization. We show how it handles incomplete or ambiguous information about the recent world state as well as alternative action effects, and how it enables us to assess the probability of the world being in a given state after an action has been performed. To this end, the formalization uses a possible models variant of Nilsson's probabilistic logic, as a basis for both reasoning about actions in the spirit of [Brewka and Hertzberg, To appear], and later, decision-theoretic planning.

2.1 Background on Possible Models And Probabilistic Logic

We first set up the possible models framework our formalization is based on, recall some basics of Nilsson's probabilistic logic, and introduce an example-domain that will be used throughout the rest of the paper.

Given a first order language $\mathcal{L}$ of closed formulae[1], 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. For instance, P might express that in absence of information about today's weather in Paris, we believe that it has 30% chance to be fine.

[1]Note that actually using an undecidable $\mathcal{L}$ will almost certainly cause practical problems. We use a propositional $\mathcal{L}$ in all examples here.

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, those and only those consistent with the constraints in K represent *possible* states of the world; they are called *possible models*. Possible models are in fact Herbrand Models of K, restricted to L. We use possible models instead of possible worlds to avoid syntax-dependent and unintuitive results obtained, e.g., in [Ginsberg and Smith, 1988]. For an arbitrary $s \in \mathcal{L}$, we define

Def. 2.1 (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*
$Poss_K(s) = \{M = \{l_1, \ldots, l_n\} \mid K \cup M \text{ is consistent and } K \cup M \vdash s\}$

$Poss_K(true)$ contains all possible state descriptions, i.e., all possible models. In the following, $Poss_K$ is used as a shorthand for $Poss_K(true)$. $Poss_K(s)$ 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.

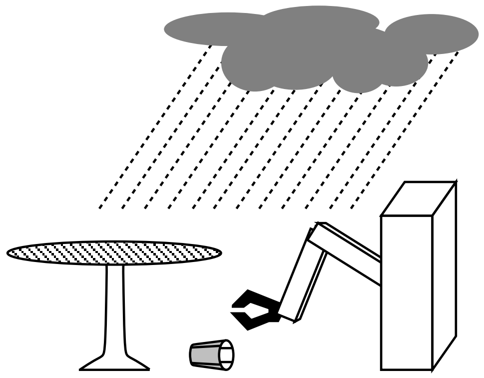
Nilsson's probabilistic logic is based on the fact that, the sets of possible worlds being mutually exclusive and exhaustive, the probability of a sentence is the sum of the probabilities of the sets of possible worlds in which this sentence is true. In order to transfer to the possible models framework the key results using sets of possible worlds in Nilsson's work, we must establish a bijection between possible models and sets of possible worlds. For that purpose, we make the *spanningness* assumption stating that L must be sufficient for possible models to represent all relevant aspects of the world:

$$\forall s \in \mathcal{L} \;\; \forall M \in Poss_K \;\; K \cup M \vdash s \text{ or else } K \cup M \vdash \neg s$$

This assumption ensures that knowing about the truth value of the elements of L is sufficient to infer, via K, the truth value of any formula in $\mathcal{L}$. Hence, the possible models being mutually exclusive and exhaustive, there is the required bijection from the set $Poss_K$ of possible models to the set of sets of possible worlds consistent with K generated by the language $\mathcal{L}$.

Given the spanningness assumption, 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, the probability distribution p over the possible models space is defined, as for sets of possible worlds, as follows:

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



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

Figure 1: The cup domain, $Poss_K$ and p

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, defined as $-\sum_{M \in Poss_K} p(M) \log p(M)$, is maximal subject to a. and b.

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

Consider an example-domain inspired by [Chrisman and Simmons, 1991] and shown in the left-hand side of Figure 1. The task of a robot is to manipulate a cup from a fixed position, using several 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) might affect the robot's performance. Assuming an appropriate definition of $\mathcal{L}$, the following background knowledge is given:

$$K = \left\{ \begin{array}{c} \mathsf{ot} \leftrightarrow \neg\mathsf{of}, \\ \mathsf{of} \leftrightarrow \mathsf{up} \vee \mathsf{fd} \vee \mathsf{bk}, \\ \mathsf{up} \rightarrow \neg\mathsf{fd} \wedge \neg\mathsf{bk}, \\ \mathsf{fd} \rightarrow \neg\mathsf{up} \wedge \neg\mathsf{bk}, \\ \mathsf{bk} \rightarrow \neg\mathsf{fd} \wedge \neg\mathsf{up} \end{array} \right\} \qquad P = \left\{ \begin{array}{c} p(\mathsf{ry}) = 0.4, \\ p(\mathsf{fd} \vee \mathsf{bk}) = 0.7, \\ p(\mathsf{ot}) = 0.2 \end{array} \right\}$$

$L = \{\mathsf{ry}, \mathsf{fd}, \mathsf{bk}, \mathsf{up}\}$ suffices to represent all relevant aspects of world states as possible models[2]. These and the probability distribution p shown in the right-hand side of Figure 1 can be computed; lacking further knowledge, they constitute the robot's beliefs.

This completes the prerequisites we will use in subsequent sections. We next 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.

[2]It is unnecessary to include ot and of in L, since their value can be deduced from those of the other atoms.

2.2 Coping With Uncertainty About World States

At planning time, many features of the current world state might be unknown. We therefore assume that information about the state of the world is given as an arbitrary sentence $s \in \mathcal{L}$, which might not completely describe this state. Given that s currently holds, our belief that a possible model M represents this state is revised. Only possible models in $Poss_K(s)$ might now correspond to the actual state, and the new probability distribution p_s over the possible models space is computed according to Bayesian conditioning[3]. Thus, $p_s(M) = p(M \mid s)$, where $p(M \mid s)$ denotes the conditional probability of M given that s holds. Using Bayes's theorem, 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

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

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

2.3 Coping With Uncertainty About Actions

We now examine 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. Independently from that, 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

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

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[4], and π_i^j is the probability that executing the action

[3] Bayesian conditioning can be seen as the probabilistic counterpart of belief revision in the sense of [Gärdenfors, 1988].

[4] Alternative action outcomes are given as alternative sets (conjunctions) of literals built of ground atoms from L, *not* as an arbitrary formula capturing all these outcomes at once. This is a difference to Winslett's [1988] approach, which is necessary to handle alternative action outcomes properly.

in the context i leads to $Post_i^j$. For simplicity of presentation, we assume that 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.

For example, 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 having the empty set as postcondition captures the intuition that the action changes nothing when it is not applicable, i.e., when ot does not hold. We define:

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

Under some assumptions about L discussed in [Brewka and Hertzberg, To appear], the approach solves both frame and ramification problems; 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*[5] *with* M:

Def. 2.2 (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 might be multiple maximally conforming models, because a postcondition can be achieved by several minimal changes in the world. Computing successor world descriptions by minimally changing the predecessor world description so as to account for the postconditions and re-establish consistency with the logical background knowledge has first been introduced into planning by Drummond [1986], with his notion of reconciliation. Reconciliation is performed on arbitrary sets of atomic propositions, which correspond, in our framework, to partial Herbrand models of K. As discussed in [Brewka and Hertzberg, To appear], reasoning using partial models, and making no distinction between essential propositions (those of L) and propositions whose truth value can be deduced from those of the formers, can lead to unintuitive results.

[5] In [Thiébaux and Hertzberg, 1992], we use the term maximally *Post*-conform instead of maximally *Post*-conforming.

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. 2.3 (Result of an action) *Let $s \in \mathcal{L}$, and a be an action as defined in (1). 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, applying $table2up$ in state $s = \mathsf{ry} \wedge (\mathsf{ot} \vee \mathsf{fd})$ yields

$$R_K(table2up, s) = \{\{\mathsf{ry}, \neg\mathsf{fd}, \neg\mathsf{bk}, \mathsf{up}\}, \{\mathsf{ry}, \mathsf{fd}, \neg\mathsf{bk}, \neg\mathsf{up}\}\}.$$

For M_1 in $Poss_K(s)$, the second context is selected, whose postconditions lead respectively to M_4 and M_2 shown in Figure 1; for M_2 in $Poss_K(s)$ the default context is selected, which means that nothing changes.

Upon learning that a is applied in s, our belief about the possible models space is updated. We compute the probability distribution $p_{(a,s)}$ over the possible models that result from performing a in s using a rule similar to Lewis's imaging[6]. 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[7]. If this property is verified, then the probability that executing a in M leads to M', where $K \cup M \vdash pre_i$, is

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

[6]Lewis's [1976] imaging can be viewed as the probabilistic counterpart of updates in the sense of [Katsuno and Mendelzon, 1991]. It corresponds to the transfer of the probability mass of a possible model to its closest neighbours according to the event that has happened. In our case, these closest neighbours are the maximally conforming models. [Dubois and Prade, 1993] studies the probabilistic and possibilistic counterparts of belief revision and updates in detail.

[7]Imposing that $Post_i^{j_1} \not\subseteq Post_i^{j_2}$ for all two postconditions $Post_i^{j_1}$ and $Post_i^{j_2}$ is a too strong condition for ensuring the mutual exclusivity of possible models resulting from the achievement of the two postconditions. Our criterion considers as valid a context of the form $[\ l \mid (\{\neg l\}, \pi), (\{\}, 1-\pi)\ ]$ even if $\{\} \subseteq \{\neg l\}$, because $\neg l$ is achieved with the first postcondition, while l remains with the second.

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

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

This example ends the description of our action formalization, about which details can be found in [Brewka and Hertzberg, To appear; Thiébaux, 1992]. We have exemplified that this 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. 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.

3 Structured Reaction Plans

We firstly 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. We are summarizing key issues from [Thiébaux and Hertzberg, 1992] here. Our description will be informal; the reader is referred to [Thiébaux, 1992] for a more formal treatment.

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 subplan 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

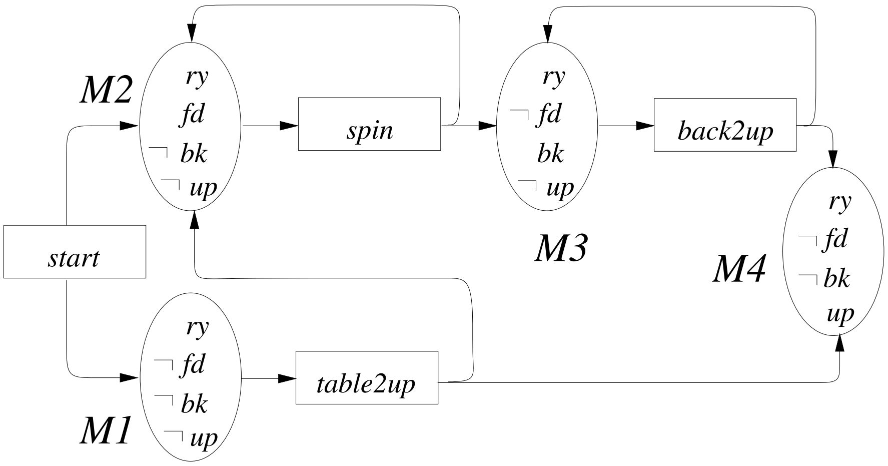


Figure 2: Plan $\mathcal{P}_1$ for the cup example

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 a blind alley in the plan. For example, applying *table2up* in an M-node where $\neg\mathsf{ot}$ holds is prohibited in a valid plan: the default context, i.e., the context denoting that the action is intuitively not applicable when $\neg\mathsf{ot}$ holds, would be selected, leading to the same M-node as unique successor, which would create a blind alley.

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, defined as follows:

$$\begin{array}{rcllll} back2up & = & [& \neg\mathsf{ry} \wedge \mathsf{bk} & | & (\{\mathsf{up}\}, 1); \\ & & & \mathsf{ry} \wedge \mathsf{bk} & | & (\{\mathsf{up}\}, 0.8),\ (\{\}, 0.2); \\ & & & \neg\mathsf{bk} & | & (\{\}, 1) \quad] \\ spin & = & [& \mathsf{fd} \vee \mathsf{bk} & | & (\{\mathsf{fd}\}, 0.5),\ (\{\mathsf{bk}\}, 0.5); \\ & & & \neg(\mathsf{fd} \vee \mathsf{bk}) & | & (\{\}, 1) \quad] \end{array}$$

The example plan $\mathcal{P}_1$ for this problem is shown in Figure 2. $\mathcal{P}_1$ 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* until it works; 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 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. In this paper, we do not discuss sensing the external world in order

These plans can be generated with any forward search strategy through the space of valid partial plans, starting with a plan embryo consisting of the *start* task and its M-nodes successors, and expanding partial plans at some of their leaves. Provided that $\mathcal{L}$ is decidable, such a search always terminates, owing to the finiteness of the state space: possible models occur only once in a plan, even if generated by different tasks. Furthermore, restricting the search to valid plans reduces the search space considerably. We have not yet defined an appropriate version of regression [Waldinger, 1977] for our action format, so that we presently do not generate plans in a backward fashion. In fact, it is highly questionable whether backward planning will make a profit when planning under uncertainty and time pressure.

It is important to understand that this plan representation provides a model of what can be predicted to happen at execution, according to a problem description. Per se, it does not intend to make all the information needed for a realistic execution explicit. In particular, even if the plan's transitions are labeled with complete possible models, we do not advocate sensing whole possible models to monitor the execution. Let us include hints to related work from which ideas can be drawn for bringing our representation closer to what is needed for sensing.

A first possibility would be to reduce the M-nodes to the *weakest* preconditions needed for ensuring that the goal can still be achieved, as is done for other approaches to encoding reactivity such as situated control rules [Drummond, 1989], triangle tables [Fikes *et al.*, 1972], and TR-programs [Nilsson, 1992]. This would offer two advantages: with respect to sensing, plans would tell explicitly the smallest sets of preconditions to be sensed; with respect to situation generality, they might apply to more situations than originally discovered by the planner. We believe that it is possible to modify our plans, once they have been generated, to retain just the strictly necessary part of the M-nodes. However, note that this would require regression, which is likely to be very expensive for an action formalism allowing for the expressivity of ours. Furthermore, explicitly representing uncertainty in our plans (branches, loops) implies that this modification process will cost additional effort compared to what is needed for traditional rule based approaches. Even then, monitoring exactly the weakest preconditions of actions may still be inadequate, because sensor actions have their own costs and side effects—in fact, they may be so costly that it is preferable to choose the next action to be applied based on probabilistic information.

[Chrisman and Simmons, 1991] proposes a solution for generating optimal plans given a static sensing policy, that might be adapted to our framework. Another approach would be to generate an optimal sensing policy, given a plan. Finding tractable ways of achieving situation generality and economic sensing is future research. In this paper, we assume that the execution monitor is responsible for updating its world model via sensing, so as to transit appropriately in the plans.

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,* compared to other approaches to encoding reactivity such as situated control rules [Drummond, 1989]. A plan can obviously be translated into a set of such rules, 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 situated control rules for handling unpredictable or unpredicted events at execution. E.g., suppose that when applying *table2up*, the cup drops on the floor, but is serendipitously spun to the bk position by a blast of wind. Obviously, the execution monitor can locate the corresponding unexpected M-node M_3 in $\mathcal{P}_1$, and resume the execution by directly executing *back2up*. On the other hand, situated control rules offer the advantages of economic sensing and situation generality, which have been addressed above.

Second, *replanning* with this plan representation can inherit subsequently 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 might 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 this section, we have seen that our plan representation shows several interesting features for planning under uncertainty and time pressure. Plans are reactive to predicted sources of uncertainty, and are a good basis to react to unpredicted events occurring at execution. Under time constraints, it is unnecessary to work on the whole possible models space at once, since plans can be incompletely generated and incrementally extended or reused later. The plan representation together 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 not have any information about to which events it should rationally plan a reaction first. The topic of the next section is to develop a framework that makes this possible.

4 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.

4.1 Basic Results About Markov Chains

We first recall basic results from Markov chain theory [Kemeny and Snell, 1960], and then explain how these results can be used to define the quality of a plan. Markov chains are stochastic processes used to described 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. 4.1 ((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*

$$\forall t \geq 0 \;\; \forall s_0, \ldots, s_{t+1} \in S \;\; \Pi(X_{t+1} = s_{t+1} \mid X_t = s_t, \ldots, X_0 = s_0) = \Pi(X_{t+1} = s_{t+1} \mid X_t = s_t)$$

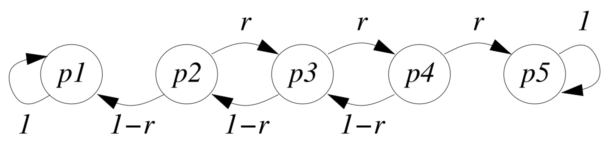
A Markov chain is stationary if and only if

$$\forall t \geq 0 \;\; \forall s_i, s_j \in S \;\; \Pi(X_{t+1} = s_j \mid X_t = s_i) = m_{ij}$$

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.

$$\begin{array}{c} \\ p_2 \\ p_3 \\ p_4 \\ \hline p_1 \\ p_5 \end{array} \begin{array}{c} \begin{array}{ccc|cc} p_2 & p_3 & p_4 & p_1 & p_5 \end{array} \\ \left(\begin{array}{ccc|cc} 0 & r & 0 & 1-r & 0 \\ 1-r & 0 & r & 0 & 0 \\ 0 & 1-r & 0 & 0 & r \\ \hline 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{array}\right) \end{array}$$



We are interested here in a special type of stationary chain called *absorbing chain.* An *absorbing* chain 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 random walk is an absorbing chain. p_1 and p_5 are absorbing, while the other states 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.

$$\begin{array}{r} \\ \mathit{trans.} \\ \hline \mathit{abs.} \end{array} \begin{array}{c} \begin{array}{c|c} \mathit{trans.} & \mathit{abs.} \end{array} \\ \left(\begin{array}{c|c} Q & R \\ \hline O & I \end{array}\right) \end{array}$$

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.

4.2 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. More formally:

Def. 4.2 (Markov chain associated with a plan) *Let s, g, K, P and 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*

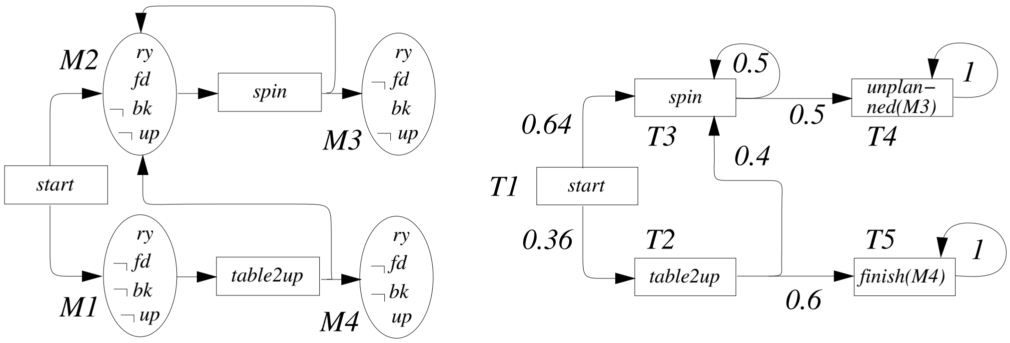
$$pre(T) = \begin{cases} pre_{\mathcal{P}}(T) & for\ T \in \mathcal{T} \\ M & for\ T = unplanned(M) \in \mathcal{T}' \\ M & for\ T = finish(M) \in \mathcal{T}' \end{cases}$$

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 3 shows the incomplete plan $\mathcal{P}_2$ for the cup example, $chain(\mathcal{P}_2)$, and its transition matrix. Note that it is the very validity property that ensures that the tasks in a plan are indeed *transient* states of the associated Markov chain, and cannot be absorbing.

Translating the results of Markov chain theory presented above to our framework, we find that for any plan $\mathcal{P}$, the ij^{th} element of the matrix $(N \quad N \times R)$ characterizing $chain(\mathcal{P})$



$$
\begin{array}{c}
 \\ T_1 \\ T_2 \\ T_3 \\ \hline T_4 \\ T_5
\end{array}
\begin{array}{c}
\begin{array}{ccc|cc} T_1 & T_2 & T_3 & T_4 & T_5 \end{array} \\
\left(\begin{array}{ccc|cc}
0 & 0.36 & 0.64 & 0 & 0 \\
0 & 0 & 0.4 & 0 & 0.6 \\
0 & 0 & 0.5 & 0.5 & 0 \\
\hline
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{array}\right)
\end{array}
$$

Figure 3: Incomplete plan $\mathcal{P}_2$ and its associated Markov chain with its transition matrix

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 task 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 [Haddawy and Hanks, 1990] 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 [Wellman and Doyle, 1992], the plan quality is defined from the utility of tasks as follows.

Def. 4.3 (Plan Quality) *Let $\mathcal{P}$ be a plan, and $(N \;\; N \times R)$ the matrix characterizing chain($\mathcal{P}$). Let u be a utility function on tasks and U be a column vector such that U_j is the utility associated with task $j \in \mathcal{P}$ by u. The quality of $\mathcal{P}$ according to u, given that task $i \in \mathcal{P}$ is executed, is the i^{th} element of the vector $(N \;\; N \times R) \times U$. This vector is noted $Qual(\mathcal{P}, u)$ and its i^{th} element is noted $Qual_i(\mathcal{P}, u)$.*

Hence, $Qual(\mathcal{P}, u)$ yields an a-priori estimation of $\mathcal{P}$'s quality by considering its first element $Qual_1(\mathcal{P}, u)$ (i.e., the element corresponding to the $start$ task), as well as updates of this estimation, given the task currently executed. As an example, consider the simple case where quality is defined as goal-achievement probability. Hence, u assigns 1 to $finish$ tasks and 0 to others. $Qual(\mathcal{P}_2, u)$ is calculated as follows, where N and R are calculated from Π as explained in Section 4.1, and where Π is itself calculated from the probabilities provided by the action formalization, as defined in Definition 4.2:

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

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 quality estimation must be recalculated. Computing the quality vector has complexity $\mathcal{O}(n^3)$, where n is the number of tasks in the plan considered.

Given a planning problem P and a utility function on tasks u, we say that a plan $\mathcal{P}$ for P is *a-priori* better, according to u, than another plan $\mathcal{P}'$ for P if and only if $Qual_1(\mathcal{P}, u) > Qual_1(\mathcal{P}', u)$. A plan for P is a-priori optimal according to u if and only if there is no a-priori better plan for P, according to u.

On-line, given that the world is in state M, a plan $\mathcal{P}$ for P is *a-posteriori* better according to u, than another plan $\mathcal{P}'$ for P if and only if one of the following conditions holds:

- $\mathcal{P}$ prescribes a task to be applied in M while $\mathcal{P}'$ does not
- $\mathcal{P}$ prescribes to apply task i in M and $\mathcal{P}'$ prescribes to apply task j, and $Qual_i(\mathcal{P}, u) > Qual_j(\mathcal{P}', u)$.

Given that the world is in state M, a plan for P is a-posteriori optimal according to u if and only if there is no a-posteriori better plan for P according to u.

Similarly, one can define an optimal plan as an a-priori optimal plan, which is a-posteriori optimal given any world state. Hence an optimal plan has to be a universal plan.

4.3 Applications of the Model

4.3.1 Use and Design of Anytime Algorithms

Anytime algorithms [Dean and Boddy, 1988] are algorithms that return an answer for any allocation of computation time and are expected to return better answers when given more time.

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., [Wah and Chu, 1990]. Furthermore, the updated estimations are suitable to incrementally improve an incomplete plan on-line, 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 spaces, thereby facilitating the design of special-purpose anytime algorithms for off-line or on-line planning. The following anytime algorithm, which can be viewed as a reformulation of the projection algorithm in [Drummond and Bresina, 1990] without considering quantitative time, plans for the most probable evolution of the environment first.

The search starts with a plan embryo containing the *start* task and the M-nodes corresponding to the initial situation of the problem. The state space is explored by selecting the non-goal leaf that is maximally probable to be reached, supposing that the start task is currently executed. This leaf can be computed according to the recent matrix $N \times R$.[8] Once this leaf is expanded (by inserting a certain task T and its M-nodes successors), the non-goal leaf that is maximally probable to be reached from T is further expanded, until the currently expanded path reaches the goal, which leads then to a new selection from the *start* task. To select among plans resulting from an expansion, the search space is explored using a simple interruptible best-first search informed by the a-priori estimation of the quality, and that guarantees monotonically increasing performance as a function of time. Pseudo-code for the algorithm is presented in Figure 4, and experimental results concerning this algorithm are given Section 5. The algorithm can also be used to improve an incomplete plan off-line. It suffices to start the search with this plan.

Note that this algorithm trivially ensures monotonic improvement of the a-priori quality of the plan, because the plan **best-plan** that will be released at the deadline is set to a newly expanded plan if and only if this latter is a-priori better than the former. A-priori quality being itself an expectation taken over world states, this does not ensure that the reward received by an agent executing a plan will always be higher than the one received when executing a plan generated for an earlier deadline. It will be only higher *when averaged over an infinite number of executions.* Trivially ensuring monotonic improvement is a possible

[8] Given that we are currently executing task C, the non-goal leaf M that is maximally probable to be reached is that, for which the element of $N \times R$ corresponding to C and $unplanned(M)$ is maximal.

```
function off-line-plan (problem,deadline,utility-func) =
   best-plan := empty-plan(problem);
   current-task := start;
   search-space := [ best-plan ];
   current-time := start-timer();
   while search-space ≠ [] interrupt-when current-time ≥ deadline
         current-plan := head(search-space);
         search-space := tail(search-space);
         if    quality(current-plan,current-task,utility-func) >
               quality(best-plan,current-task,utility-func)
         then best-plan := current-plan fi;
         T := last-inserted-task(current-plan);
         L := select-maximal-probable(current-plan,T,problem,current-task);
         if    L ≠ nil
         then successors := expand(current-plan,L,problem);
               successors := order(successors,utility-func,current-task);
               search-space := append(successors,search-space) fi
   end;
   return best-plan
```

empty-plan(p) builds a plan embryo containing the start task and the possible models of the initial situation of problem p.

quality(P, C, u) returns the quality of plan P according to the utility function on tasks u, given that the currently executed task is C.

last-inserted-task(P) returns the lastly inserted task in plan P.

select-maximal-probable(P, T, p, C) returns the leaf of plan P that is maximally probable to be reach from task T, unless one of the leaves that can be reached from T matches the goal of problem p. In that case, returns the leaf that is maximally probable to be reach from C and which does not match this goal. If there is none, then returns *nil*.

expand(P, l, p) returns the list of valid plans resulting from the expansion of plan P at leaf l, using the available actions of problem p.

order(P, u, C) order the list of plans P in decreasing order of quality according to the utility function on tasks u, given that task C is currently executed.

Figure 4: A simple off-line anytime planning algorithm

strategy in designing an anytime algorithm. This strategy is not enforced at all by our framework; it is just introduced here as one instance of the framework. It can be disadvantageous for domains requiring quick reaction. It implies that if the utility function is such that one must generate a complete path (or even a complete plan) to the goal before seeing quality increase, no partial path (or even no partial plan) will ever be released. Reciprocally, if the utility function assigns a heuristic value to an *unplanned* task that overestimate its gain, a complete plan might never be generated even if one exists.

The algorithm requires only a few modifications to be suitable for the on-line improvement of an incomplete plan. First, it must not backtrack on an already executed task. Second, current-task must not be fixed to the *start* task, but must vary according to the current task of the execution. Last, in order to focus on useful improvements with respect to the remainder of the execution, the selection process must be performed each time a new task gets executed.

4.3.2 Improving Reactive behavior

Apart from on-line plan generation, the ability to update the quality estimation has another advantage for a reactive executor alone: if many reaction plans are available for the current 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. Kanazawa and Dean [1989] follow this approach: their 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 [Drummond *et al.*, 1993; Beetz and McDermott, 1992], it can be reasonably assumed, for a broad class of applications, that a reactive executor can be designed that uses default user-provided 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 plans.

The interesting case is the intermediate one where only an incomplete plan could be generated: the reactor can begin by executing this plan, and ends the execution with its default plans. This case leads to an important problem: since the incomplete plan might not lead to a solution, i.e., be a prefix of no solution, it might decrease the executor's performance. [Drummond *et al.*, 1993] presents the RFS type of search (Reaction-First Search) which is designed to solve this problem. By exploring the possible behaviors of the reactive executor and constraining them, the search releases incomplete sequences of actions, that will, when averaged on an infinite number of runs, monotonically increase the executor's performance, performance being understood as goal-achievement probability.

Our model provides another way of dealing with such a combination planner/reactive executor, that is not limited to a definition of quality as goal-achievement probability. Let the problem-related utility being expressed via a utility function on tasks u. In order to improve the behavior of the reactive executor, it suffices that the planner computes quality according to the utility function on tasks that maps:

- non-*unplanned* tasks T to the value $u(T)$
- *unplanned*(M) tasks to the a-posteriori quality, according to u, of the a-posteriori best default plan, given that the world is in state M.

By using such a quality measure, the algorithm presented in Figure 4 will attempt to replace an *unplanned* task by a newly generated task (or subplan) if and only if this outperforms the default behavior produced when the *unplanned* task is encountered.

This strategy is different from RFS's. Rather than exploring first the default behaviors and constraining them, the planner directly attempts to generate incomplete plans after the execution of which the default plans will be of higher utility than they initially were. RFS performs better than this method if a large number of default plans is available, provided that these plans already have, on average, a high quality for solving the problem. On the other hand, our method is more appropriate than RFS if the default plans have a low utility. RFS does not need to calculate qualities to guaranty monotonic improvement, but is limited to a definition of quality as goal-achievement probability.

5 Implementation and Experimental Results

PASCALE2, a domain-independent planning system based on our model, has been implemented in Standard ML on top of the `qwertz` toolbox [Gordon *et al.*, in preparation], a software toolbox for building planning systems developed at GMD.

The most time-critical task in PASCALE2 is that of computing the possible models and the probability distribution p in accord with the background knowledge. Fortunately, this task is to be performed only once per domain, when this domain is first presented to PASCALE2. The task of computing all possible models is the most costly of the whole procedure, being exponential in the cardinality of L. Once this step has been performed, it is fairly simple to determine an approximation of p, following the lines given in [Cheeseman, 1983]. If P is, as before, the probabilistic background knowledge, this computation amounts to solving a non-linear system of k equations, where $k = card(P) + 1$, i.e., the number of sentences, including *true*, for which an a-priori probability is available.

One cannot compute the exact solution of such a non-linear system, but only an approximation as accurate as needed. This can be done using Newton's method, whose complexity is $\mathcal{O}((k^3 + m)i)$, where m is the complexity of computing the Jacobian of the system, and i the number of Newton's iterations required to obtain the desired accuracy. Since a relatively low accuracy is required, and since Newton's method converges quadratically, i is small. Unfortunately, m can be as large as the number of possible models times k^3 in this system.

The theoretical complexity of solving this system is then $\mathcal{O}(k^3 2^n i)$. However, in general, the number of possible models will be much lower than 2^n, and from our practical experience, the time required to compute p can be neglected, compared to that of computing the possible models.

The single reason for computing all possible models in advance is that this is necessary for obtaining the probability distribution p. A solution that can be applied to a broad class of domains is to have a special-purpose procedure generate the models, and PASCALE2 compute p. This is the solution we usually choose when confronted to large domains. Note that if no a-priori probability is available for a domain, then the whole computation should be skipped. Possible models of the initial situation of a problem as well as multiple maximally conforming models will then be equiprobable, but the probabilities of alternative action effects will still be taken into account.

Since PASCALE2 is still in its infancy, we use a straightforward algorithm for computing the result of applying action a in situation s, namely that one directly derived from the definitions given in Section 2. We

- compute the possible models in s,
- select the appropriate context of a for each of these models,
- compute the possible models in each of the postconditions of the selected context,
- filter the models that are maximally conforming,
- compute the union of the maximally conforming models for each model in s and each postcondition of the corresponding selected context in a,
- and finally compute the resulting probability distribution.

The performance of the algorithm is acceptable because the possible models in s as well as those in the postconditions of a can be computed by just updating the available complete set of possible models, using, in our case, a sequent-calculus based theorem prover. Moreover, the availability of the possible models set makes validating action preconditions faster.

However, we plan to use the more efficient algorithm described in [Chou and Winslett, 1991], which can trivially be adapted to our framework.[9] Instead of computing all possible models in a postcondition $Post$ and filtering maximally $Post$-conforming models with a possible model M in s, the latter algorithm incrementally updates $M \cup Post$ so as to make it consistent with the logical background knowledge K. For this purpose, it modifies a minimal subset of M, by looking to see inhowfar M is inconsistent with $K \cup Post$. The drawback

[9][Del Val, 1992] also proposes algorithms for updating databases, based on a syntactic characterization of updates. However, in the presence of a large logical background knowledge, those seem to be less efficient than the algorithm given in [Chou and Winslett, 1991].

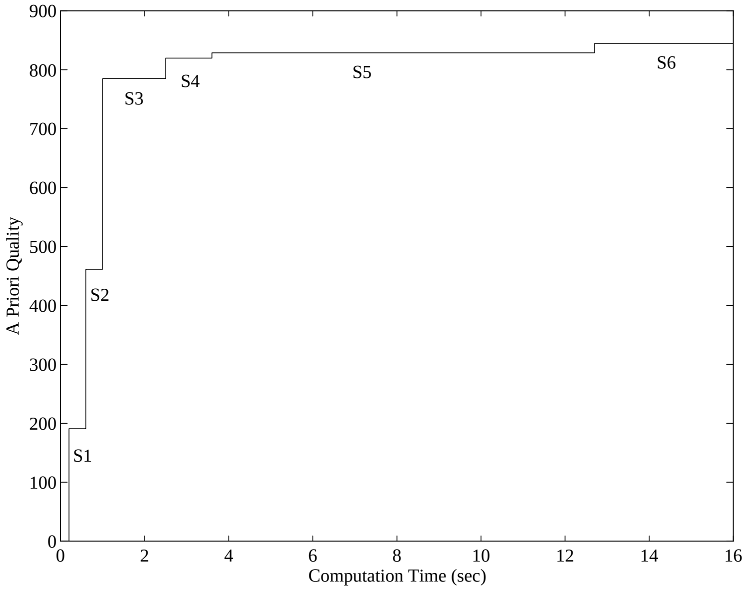


Figure 5: A-priori plan quality as a function of computation time

of this algorithm is that it must be given a complete set of counter-models of K. However, the complexity of computing all counter-models is much lower than that of computing all possible models, provided that K primarily consists of implications, which is the natural case.

From our experiences, the exploration of the finite search space of valid plans does not cause much performance problems. Naturally enough, if the planner is given a highly incomplete initial situation and if actions are highly stochastic, plans will branch excessively. However, the practical disadvantage of this branching factor is reduced by the fact that possible models are uniquely represented in a plan. In the current stage of implementation, PASCALE2's anytime algorithms library only consists of simple algorithms for exploring the search and state spaces, such as that of Figure 4. Part of our future implementation work will be devoted to building more elaborate algorithms, and to their compilation into anytime algorithms with an optimal performance profile [Zilberstein and Russel, 1992].

Experiments with PASCALE2 were conducted on a Sparc IPX workstation. Figure 5 shows the quality of the plans generated by the algorithm of Figure 4 as a function of computation time, for our cup example. The example was augmented with the action *wait*, for waiting during a fixed amount of time with a 0.1 probability that the weather changes during waiting:

$$\begin{array}{lcll} wait & = & [& \mathsf{ry} \mid (\{\neg\mathsf{ry}\}, 0.1),\ (\{\}, 0.9); \\ & & & \neg\mathsf{ry} \mid (\{\mathsf{ry}\}, 0.1),\ (\{\}, 0.9) \quad] \end{array}$$

The problem-related utility was expressed using a utility function on tasks mapping the *start* task to 0, *finish* tasks to 1500 (i.e., reaching the goal provides a gain of 1500), tasks built from *spin* to -200, tasks built from *back2up* and *table2up* to -300, tasks built from *wait* to -5, and *unplanned*(M) tasks to a number roughly estimating the proximity of M with the goal. Note that *wait* has a much lower cost than the other actions because the experiment was interested in minimizing the physical resources such as energy, rather than execution time.

Figure 5 shows that the plan quality increases step by step with the timeline. These steps corresponds to different plans $S1$ through $S6$ depicted in Figure 6. The planner first releases the two incomplete plans $S1$ and $S2$, the latter having a 0.64 goal-achievement probability. The following plans $S3$ through $S6$ all achieve the goal with probability 1. The complete plan $S3$, which is simply our plan example $\mathcal{P}_1$, is found after 1 second computation time. Thereafter, less resource-consuming plans $S4$, $S5$ and $S6$ that include *wait* actions are successively found, $S6$ being a-priori optimal for the problem considered. Note that with a more powerful function for evaluating the proximity of *unplanned* M-nodes with the goal, the algorithm does converge directly towards $S6$, which is then found within 2 seconds. Experiments on larger blocks-world type benchmark problems also gave satisfactory results.[10] Moreover, we expect the run-time performance of PASCALE2 to be considerably improved by using Chou and Winslett's algorithm. PASCALE2 has not yet been tested on real-world problems.

6 Summary and Further Related Work

To conclude, let us first summarize what we have achieved and then briefly discuss the relevance of our results.

In this paper, we aim at viewing planning as a choice under uncertainty and time pressure, for which symbolic planning and decision theory act as two complementary functions. As Haddawy and Hanks [Haddawy and Hanks, 1992] point out,

> Symbolic planning provides a computational theory of plan generation ... Decision theory provides a normative model of choice under uncertainty.

Within our model, symbolic planning enables the search of a reaction plan under uncertainty stemming 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 [Thiébaux and Hertzberg, 1992].

The new issue is basically 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 to reason within single world situations; we use Markov chain theory to

[10] Moreover, the current version of the system does not use tricky implementations of sets, and matrix manipulation is far from being as optimal as with the existing numerical analysis tools.

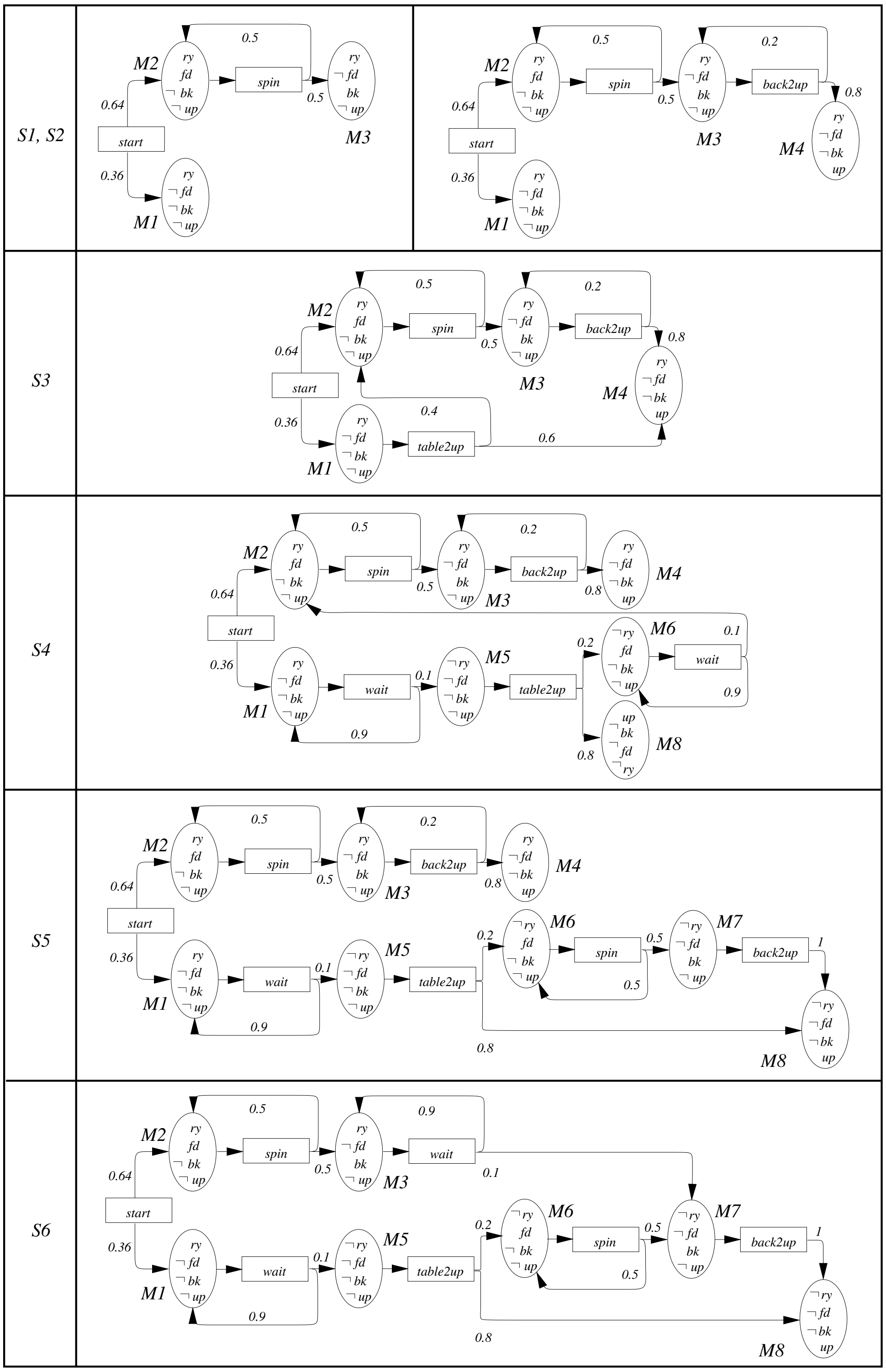


Figure 6: Plan solutions $S1$ through $S6$ found for increasing allocations of computation time

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 estimated quality of the plan currently under execution turns out to be lower than that of some other suitable off-the-shelf plan. Gathering statistics about plan quality as a function of time in our system will also enable future versions of PASCALE2 to reason about the value of computation [Horovitz, 1988; Russel and Wefald, 1989; Zilberstein and Russel, 1992], i.e., to reason about when to stop planning and about which computational action to perform.

The work on Markov decision processes can be profitably used in the planning framework. Koenig [Koenig, 1991] has shown how to elegantly model a planning problem as a Markov decision problem, and to derive an anytime algorithm producing universal plans from the policy iteration algorithm [Howard, 1960]. A major drawback is that the algorithm must work on the complete state space at once, which is avoided by our model. Recently, new 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 [Dean *et al.*, 1993]. This approach is particularly interesting when planning under time pressure. It makes use of several iterative refinement routines from which a global anytime algorithm can be built that allocates computation time to the various 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 AI approaches to representing actions. 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-theoretic methods. It would then be interesting to see how Dean's algorithms can be used in the framework of our representation.

Garcia [Garcia, 1993] has developed time-independent non-linear and hierarchical planning algorithms, based on a formalization of actions similar to ours. This formalization takes into account context-dependent effects, and plans can be revised on-line, according to additional information acquired. However, actions must be deterministic, and hence, plans remain classical partially ordered sets of actions. We believe that this work can be used as a basis for integrating hierarchical planning to our approach.

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. To generate them within our model, everything about the world must be known, all actions are context free and unambiguous. The reactive execution of such a plan is comparable to that of a STRIPS triangle table [Fikes *et al.*, 1972]. See [Thiébaux and Hertzberg, 1992] for details.

Universal plans, as proposed in [Schoppers, 1987]. To generate them, nothing about the initial situation is known, all actions are context free and unambiguous.

Decision trees, as proposed by [Feldman and Sproull, 1977]. No restrictions need to be made to generate them.

Situated control rules as from [Drummond, 1989; Drummond and Bresina, 1990], see Sections 3 and 4.

PASCALE **plans,** as described in [Thiébaux and Hertzberg, 1992]: to generate them, possible models always have equiprobability.

Our work differs from the traditional approaches to generating reaction plans, e.g. [Schoppers, 1987; Drummond, 1989; Nilsson, 1992], in that the effects of actions are not assumed to be deterministic for the purpose of planning.

A non-instance of our framework is, e.g., the planner by Kanazawa and Dean [1989], 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 within this model. The planner uses a revolving approach, that trades plan quality for computation time. It suffers from a limited horizon problem, and hence cannot make long-term predictions. Extending our model to handle external events at planning time as in Kabanza's [1990] world automata is part of our future research.

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.

References

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

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

[Brewka and Hertzberg, To appear] 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.

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

[Chou and Winslett, 1991] T. Chou and M. Winslett. Immortal: a model-based belief revision system. In *Proc. KR-91*, pages 99–109, 1991.

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

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

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

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

[Dean and Wellman, 1991] T.L. Dean and M.P. Wellman. *Planning and Control.* Morgan Kaufmann, 1991.

[Dean *et al.*, 1993] 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.

[Del Val, 1992] A. Del Val. Computing knowledge base updates. In *Proc. KR-92*, pages 740–750, 1992.

[Drummond, 1986] M. Drummond. A representation of action and belief for automatic planning systems. In M. Georgeff and A. Lansky, editors, *Proc. Reasoning About Actions and Plans*, pages 189–211, Morgan Kaufmann, 1986.

[Drummond, 1989] M. Drummond. Situated control rules. In *Proc. KR-89*, pages 103–113, 1989.

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

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

[Dubois and Prade, 1993] D. Dubois and H. Prade. Belief revisions and updates in numerical formalisms. an overview, with new results for the possibilistic framework. In *Proc. IJCAI-93*, pages 620–625, 1993.

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

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

[Garcia, 1993] F. Garcia. *Révision des croyances et révision du raisonnement pour la planification.* PhD thesis, Ecole Nationale Supérieure de l'Aéronautique et de l'Espace, Toulouse, France, February 1993.

[Gärdenfors, 1988] P. Gärdenfors. *Knowledge in Flux.* MIT Press, 1988.

[Ginsberg and Smith, 1988] M.L. Ginsberg and D.E. Smith. Reasoning about action I: a possible worlds approach. *J. Art. Intell.*, 35:165–195, 1988.

[Gordon *et al.*, in preparation] T. Gordon, J. Hertzberg, and A. Horz. `qwertz`. a toolbox for building AI planners. in preparation.

[Haddawy and Hanks, 1990] 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.

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

[Hanks, 1990] S. Hanks. Practical temporal projection. In *Proc. AAAI-90*, pages 158–163, 1990.

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

[Horovitz, 1988] E. Horovitz. Reasoning under varying und uncertain resource constraints. In *Proc. AAAI-88*, pages 111–116, 1988.

[Horz, 1993] A. Horz, editor. *Beiträge zum 7. Workshop "Planen und Konfigurieren".*, Arbeitspapiere der GMD Nr. 723, January 1993.

[Howard, 1960] R.A. Howard. *Dynamic Programming and Markov Processes.* MIT Press, 1960.

[Kabanza, 1990] F. Kabanza. Synthesis of reactive plans for multi-path environments. In *Proc. AAAI-90*, pages 164–169, 1990.

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

[Katsuno and Mendelzon, 1991] H. Katsuno and A.O. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proc. KR-91*, pages 387–394, 1991.

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

[Koenig, 1991] S. Koenig. *Optimal Probabilistic and Decision-Theoretic Planning using Markovian Decision Theory*. Master's thesis, University of California at Berkeley, May 1991.

[Lewis, 1976] D.K. Lewis. Probabilities of conditionals and conditionals probabilities. *The Philosophical Rev.*, (85):297–315, 1976.

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

[Nilsson, 1992] N.J. Nilsson. *Toward Agent Programs with Circuit Semantics*. Technical Report STAN-CS-92-1412, Stanford University, 1992.

[Peot and Smith, 1992] M.A. Peot and D.E. Smith. Conditional nonlinear planning. In *[Hendler, 1992]*, pages 189–197, 1992.

[Russel and Wefald, 1989] S. Russel and E. Wefald. On optimal game-tree search using rational meta-reasoning. In *Proc. IJCAI-89*, pages 334–340, 1989.

[Schoppers, 1987] M.J. Schoppers. Universal plans for reactive robots in unpredictable environments. In *Proc. IJCAI-87*, pages 1039–1046, 1987.

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

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

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

[Wah and Chu, 1990] 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.

[Waldinger, 1977] R. Waldinger. Achieving several goals simultaneously. *Machine Intelligence*, 8:94–136, 1977.

[Warren, 1976] D. H. D. Warren. Generating conditional plans and programs. In *AISB Summer Conference, Edinburgh*, pages 344–354, 1976.

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

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

[Winslett, 1988] M. Winslett. Reasoning about action using a possible models approach. In *Proc. AAAI-88*, pages 89–93, 1988.

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