Towards Feasible Approach to Plan Checking Under Probabilistic Uncertainty: Interval Methods

Raúl Trejo and **Vladik Kreinovich**
Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
{rtrejo,vladik}@cs.utep.edu

Chitta Baral
Dept. of Computer Science & Engineering
Arizona State University
Tempe, AZ 85287-5406, USA
chitta@asu.edu

Abstract

The main problem of *planning* is to find a sequence of actions that an agent must perform to achieve a given objective. An important part of planning is checking whether a given plan achieves the desired objective. Historically, in AI, the planning and plan checking problems were mainly formulated and solved in a *deterministic* environment, when the initial state is known precisely and when the results of each action in each state is known (and uniquely determined). In this deterministic case, planning is difficult, but plan checking is straightforward. In many real-life situations, we only know the probabilities of different fluents; in such situations, even plan checking becomes computationally difficult. *In this paper, we describe how methods of interval computations can be used to get a feasible approximation to plan checking under probabilistic uncertainty.* The resulting method is a natural generalization of 0-approximation proposed earlier to describe planning in the case of partial knowledge. It turns out that some of the resulting probabilistic techniques coincides with heuristically proposed "fuzzy" methods. Thus, we justify these fuzzy heuristics as a reasonable feasible approximation to the (NP-hard) probabilistic problem.

Traditional (deterministic) planning and plan checking

The main problem of *planning* is to find a sequence of actions that an agent must perform to achieve a given objective. An important part of planning is checking whether a given plan achieves the desired objective; this *plan checking* is also called *projection*.

Historically, in AI, the planning and plan checking problems were mainly formulated and solved in a *deterministic* environment, when the initial state is known precisely and when the results of each action in each state is known (and uniquely determined) (Allen et al. 1990).

To formulate the deterministic planning problem precisely we must be able to describe states of the world and actions. States are usually characterized by their properties (*fluents*) $f_1, \ldots, f_n$; the set of all possible fluents will be denoted by $\mathcal{F}$. A state s can thus be defined simply as a set of fluents, meaning the set of all the fluents which are true in this state.

At each moment of time, an agent can execute an action. We will denote the set of all possible actions by $\mathcal{A}$. We have *rules* which describe how an action $a \in \mathcal{A}$ changes a state s; these rules are of the form "a causes F if $F_1, \ldots, F_m$", where F and F_i are *fluent literals*, i.e., fluents or their negations. The set of all such rules is called a *domain description* and denoted by D. The result $res(a,s)$ of applying the action a to the state s is thus defined as
$res(a,s) = \{f \mid (f \in s \& \neg f \notin V_D(a,s)) \vee f \in V_D(a,s)\}$,
where $V_D(a,s)$ denotes the set of conclusions of all rules from D for which all conditions hold in s.

A *plan* α is a sequence of of actions $\alpha = [a_1, \ldots, a_n]$; the result $res(a_n, res(a_{n-1}, \ldots, res(a_1, s)\ldots))$ of applying these actions to the state s will be denoted by $res(\alpha, s)$.

To complete the description of deterministic planning, we must formulate possible objectives. In general, as an objective, we can take a complex combination of elementary properties (fluents) which characterize the final state; for example, a typical objective of an assembling manufacture robot is to reach the state of the world in which all manufactured items are fully assembled. To simplify the description of the problem, we can always add this combination as a new fluent literal (see, e.g. (Allen et al. 1990)); thus, without losing generality, it is sufficient to consider only objectives of the type $f \in \mathcal{F}$.

In these terms, the *planning* problem can be formulated as follows: given a set of fluents $\mathcal{F}$, a goal $f \in \mathcal{F}$, a set of actions $\mathcal{A}$ and a set of rules D describing how these actions affect the state of the world, to find a sequence of actions $\alpha = [a_1, \ldots, a_k]$ that, when executed from the initial state of the world s_0, makes f true. The problem of *plan checking* is, given $\mathcal{F}$, $\mathcal{A}$, a goal, and a sequence of actions α, to check whether the goal becomes true after execution of α in the initial state. In the deterministic case, planning is computationally difficult (NP-hard, see, e.g., a survey (Baral et. al. 1999)), but checking a given plan is feasible: it can be done by a straightforward computation of $res(\alpha, s_0)$ (Allen et al. 1990).

Imperfect sensors: First reason to consider planning and plan checking under probabilistic uncertainty

In real life, we often do not know have a complete knowledge of the initial state s_0. For some fluents $f \in \mathcal{F}$, we know whether f or $\neg f$ are initially true, but for some other fluents f, we may only know the *probability* $p(f, s_0)$ that f is initially true. (The probability $p(\neg f, s_0)$ that $\neg f$ is true is then equal to $1 - p(f, s_0)$.)

This probability may reflect either the expert's uncertainty in his own opinion (such probabilities are called *subjective*), or the fact that sensors and measuring instruments used to determine whether this fluent holds are never 100% reliable (such probabilities are called *objective*).

Towards the description of the corresponding planning and plan checking problems

Due to this probabilistic uncertainty, we do not know the exact initial state s_0; at best, we know the *probabilities* $P(s = s_0)$ of different initial states s. These probabilities (and similar uncertainty about observations in the future) form the basis of the Partially Observable Markov Decision Processes (POMDP) approach to planning under probabilistic uncertainty (see, e.g., (Heffner et al. 1998; Kaelbling, et al. 1998) and references therein).

Most examples of using POMDP assume that we know the probabilities of all possible initial states. Such a representation is possible for a simple example in which the total number $n_f = \#(\mathcal{F})$ of fluents is small: Indeed, since we define a state s as a subset $s \subseteq \mathcal{F}$ of the set of all the fluents, the number of all possible states is equal to the number of all possible subsets, i.e., to 2^{n_f}. Thus, to describe the probability $p(s)$ of each state, we need to use $2^{n_f} - 1$ real numbers $p(s) \in [0, 1]$ ($2^{n_f} - 1$ since $\sum p(s) = 1$, so one of the values $p(s)$ is uniquely determined if we know the others). Even for $n_f = 20$ fluents, we need more than a million different probabilities. In such situations, we cannot represent this probabilistic uncertainty by describing the probability of every possible initial state s.

In other words, we may know the probability $p(f, s_0)$ of each of n_f fluents $f \in \mathcal{F}$, but we may not know the probability of each of 2^{n_f} possible states $s \subseteq \mathcal{F}$.

In some cases, we know that different fluents are statistically independent; in this case, if we know the initial probabilities $p(f_i, s_0)$ of different fluents $f_i \in \mathcal{F}$, we can determine the probability of every possible state. For example, for $\mathcal{F} = \{f_1, f_2\}$, the probability $p(s_0 = \{f_1\})$ that the initial state s_0 coincides with $\{f_1\}$ is equal to the probability that in the initial state s_0, f_1 is true and f_2 is false, i.e., is equal to $p(s_0 = \{f_1\}) = p(f_1, s_0) \cdot p(\neg f_2, s_0) = p(f_1, s_0) \cdot (1 - p(f_2, s_0))$.

In real life, often, the errors of different sensors are partially caused by the same causes and therefore, are not independent; in most such cases, we do not know the exact extent of this correlation. Thus, even when we know the exact initial probability $p(f, s_0)$ of each fluent $f \in \mathcal{F}$, we may have several different probability distributions P on the set of all initial states; the set of all possible distributions coincides with the set of all distributions P for which the probability of each fluent f is equal to the given value: $P(f) = p(f, s_0)$. In this case, we will say that the probability distribution P is *consistent* with the given probabilities $p(f, s_0)$.

Even the probabilities $p(f, s_0)$ of different fluents may not be known exactly. Indeed, a natural way to get these probabilities is to apply statistical techniques to the records of sensor errors and/or expert judgments. From these statistical estimates, however, we do not get the *exact* values of these probabilities, we can only get *approximate* values. So, in reality, for a fluent f, often, we do not even know the exact value of the initial probability $p(f, s_0)$, we only know the *interval* $\mathbf{p}(f, s_0) = [p^-(f, s_0), p^+(f, s_0)]$ which contains the actual (unknown) value of $p(f, s_0)$ (see, e.g., (Givan et al. 1997). In this case, possible values of the probability $p(\neg f, s_0)$ that $\neg f$ hold is s_0 form the interval

$$\mathbf{p}(\neg f, s_0) = [1 - p^+(f, s_0), 1 - p^-(f, s_0)].$$

Comment. An even more general approach, related to Dempster-Shafer formalism, allows for the possibility that for some fluents $f_1, \ldots, f_k$, we do not even the intervals $\mathbf{p}(f_i, s_0)$; instead, we may know, e.g., only the probability $\mathbf{p}(f_1 \vee \ldots \vee f_k, s_0)$. Heuristic methods for this approach (based on independence-type assumptions) were considered, e.g., in (Lowrance et al. 1990); guaranteed estimates which are not based on any heuristic assumptions are described in (Doan et al. 1996).

The cases of complete information about the fluent and the complete lack of information about this fluent can be described as a particular case of the interval formulation: $\mathbf{p}(f, s_0) = [1, 1]$ means that we are 100% sure that f is true; $\mathbf{p}(f, s_0) = [0, 0]$ means that we are 100% sure that f is false; and $\mathbf{p}(f, s_0) = [0, 1]$ means that we have no information about f.

In this realistic case of interval probabilities $\mathbf{p}(f, s_0)$ ($f \in \mathcal{F}$), we have to consider all probability distributions P which are *consistent* with these interval probabilities, i.e., all probability distributions P for which the probability of each fluent f belongs to the corresponding interval: $P(f) \in \mathbf{p}(f, s_0)$.

If we have a complete domain description D, then, for each initial state s and for each action plan α, we can determine the resulting state $res(\alpha, s)$. Thus, for every probability distribution P which is consistent with the given interval probabilities, and for every objective fluent f, we can determine the probability $p(f, res(\alpha, s_0))$ that this objective will be satisfied in the final state, as the sum $\sum P(s)$ of the probabilities of all initial states s for which f holds in $res(\alpha, s)$. Since the interval probabilities do not determine the probability distribution uniquely, we may have different probability distributions P which are consistent with this data; for different distributions, we may get different values $p(f, res(\alpha, s_0))$. Let us denote the interval of all possible values of such probabilities by $\mathbf{p}(f, res(\alpha, s_0)) = [p^-(f, res(\alpha, s_0)), p^+(f, res(\alpha, s_0))]$.

Since we do not have the complete knowledge of the initial state, we, therefore, cannot be 100% sure that the result of applying a given sequence of actions will always lead to a given objective; at best, we can hope that the probability of achieving a given objective is high enough, i.e., that this probability is higher than a given number p_0. So, in this probabilistic context, the planning problem means finding a plan α for which $p^-(f, res(\alpha, s_0)) \geq p_0$. Correspondingly, a plan checking problem means checking whether this inequality holds for a given plan α.

Summarizing: with probabilistic uncertainty in sensors, the planning problem takes the following form. We are given a set of fluents $\mathcal{F}$, a set of actions $\mathcal{A}$, a domain description D (i.e., set of rules which describe how actions affect the state), the initial interval probabilities $\mathbf{p}(f, s_0)$ for all $f \in \mathcal{F}$, the objective $f \in \mathcal{F}$, and the desired success probability p_0. Our goal is to find a sequence of actions α such that the probability $p(f, res(\alpha, s_0))$ of the objective f being true after execution of α is guaranteed to be greater or equal than p_0.

A natural step in solving the planning problem is checking the given plan. The problem of *plan checking* is, given a planning problem and a candidate plan α, to check whether the probability $p(f, res(\alpha, s_0))$ is guaranteed to be greater or equal than p_0, i.e., whether $p^-(f, res(\alpha, s_0)) \geq p_0$.

For probabilistic uncertainty, even plan checking is NP-hard; so, we need a good approximate plan checking algorithm

In contrast to the deterministic planning where plan checking is easy, the probabilistic plan checking problem is NP-hard: indeed, in (Baral et. al. 1999), it is shown, in effect, that this problem is NP-hard even if we only allow intervals of the type $[0,0]$, $[1,1]$, and $[0,1]$ (see also (Littman et al. 1998)). It is therefore desirable to find good approximate algorithms for plan checking.

A natural way to check the success of a plan is to estimate the probability $p^-(f, res(\alpha, s_0))$, and then compare the resulting estimate with the desired probability value p_0. Since it is NP-hard to check whether $p^- \geq p_0$, it is, therefore, NP-hard to compute the value p^-; thus, at best, we can look for a feasible algorithm for computing a good approximation $\widetilde{p}^-$ for the desired difficult-to-compute probability bound p^-.

This word "approximation" may mean two things: it may mean that our algorithm misses a successful plan (i.e., it is unable to confirm that a plan is successful), and it may mean that the approximate algorithm erroneously declares a bad plan to be successful.

There are many heuristic techniques which provide us with approximate values of probabilities; e.g., techniques based on fuzzy logic have been successfully combined with more traditional AI techniques to produce an award-winning robot (Congdon et al. 1993; Saffiotti et al. 1995). For a robot, usually, errors of both types are equally bad, so we just try to minimize the total number of such errors.

In many real-life applications, missing a successful plan is bad, but selecting a failing plan as supposedly successful can be disastrous: e.g., when we plan to send astronauts on a space mission, it is bad but still tolerable if we miss a possibility of a cheaper mission and thus erroneously overestimate the mission's cost, but it would be a disaster to send a mission with a low success probability under the erroneous assumption that this mission's success probability is high. Due to this fact, we are interested in a plan checking algorithm which will never overestimate the success probability. In other words, we want to guarantee that our estimate $\widetilde{p}^-$ never exceeds the actual value p^-: $\widetilde{p}^- \leq p^-$.

In the following text, we will show, among other things, that the above-mentioned fuzzy techniques (which were not originally intended to provide such guarantees) can, in fact, lead to guaranteed estimates.

Imperfect actuators: Second reason to consider planning and plan checking under probabilistic uncertainty

In the above text, we took into consideration that sensors can be imperfect, but we still assumed that the actuators are perfect, i.e., that the results of each action in a given state are uniquely determined by our choice of this action. In reality, of course, actuators are also imperfect; as a result, if we apply, several times, the same action a to the same state s, we may get different results. For example, if we want a robot to move forward, we send to it, at several consequent moments of time, a signal to go forward. Due to actuator errors, a robot usually deviates from the desired trajectory, and this deviation may change from time to time (the robot "wobbles"). In other words, instead of deterministic rules of the type "a **causes** F **if** $F_1, \ldots, F_m$", we now have probabilistic rules of the type

$$a \textsf{ causes } F \textsf{ with probability } p \textsf{ if } F_1, \ldots, F_m.$$

Similarly to the sensor uncertainty, we may not know the exact values of the corresponding probabilities; instead, we only know the *interval* of possible values. In this case, the probabilistic rules take the following form:

$$a \textsf{ causes } F \textsf{ with probability } p \in \mathbf{p} \textsf{ if } F_1, \ldots, F_m$$

for some given probability interval $\mathbf{p}$. To illustrate this idea, let us give three simple examples: a deterministic action corresponds to $\mathbf{p} = [1,1]$; a non-deterministic statement that an action a *may* cause F can be described by $\mathbf{p} = [0, 1]$, and coin toss leads to $F =$ "heads" with probability $\mathbf{p} = [0.5, 0.5]$.

These additional interval probabilities make the planning and plan checking problems even more complicated. In POMDP description, these probabilities are taken into consideration; however, it is assumed that the probabilities corresponding to consequent actions are independent. This is indeed true in some real-life situations when the actuator errors are purely random. In real life, many actuators also have a systematic error component, i.e., a component which leads to a strong correlation between probabilities corresponding to consequent actions (this assumption is also made in the interval version of POMDP, described, e.g., in (Draper et al. 1994)). Thus, in our description, we do not want to assume this independence; instead, similarly to sensor uncertainty, we consider all possible probability distributions which are consistent with the given interval probabilities.

This framework – no independence assumption at all, probabilities are allowed to change over time within the intervals, etc. – may yield overly conservative estimates for the probabilities (especially for long plans), but it is unavoidable in the situations when we need to *guarantee* that the plan succeeds with a given probability.

Taking actuator imperfection into consideration makes the plan checking problem even more complex. However, from the computational viewpoint, for realistic (polynomial-length) plans, we can reformulate this new uncertainty in the equivalent form of initial state's uncertainty. The motivation behind this reduction is very simple and natural: In the deterministic description, to find the post-action state of the system, it is sufficient to know its pre-action state. In the more realistic situation, to determine the post-action state uniquely, we must also know the pre-action state of the actuator. Thus, to reduce the new description to the previous one, we can add, to the the original set of fluents $\mathcal{F}$ which describe the state of the system itself, additional fluents which describe the state of the actuator. Namely, for each action a_i from the actions sequence (plan) α, and for each rule from D (of the type "a_i causes F with probability $p \in \mathbf{p}$ if $F_1, \ldots, F_m$") in which the result of this action a_i is not uniquely determined, we add a new fluent f_r whose meaning is that this rule leads to F if this fluent is true. Then, the original probabilistic rule is replaced by the deterministic rule "a causes F if $F_1, \ldots, F_m, f_r$", where the initial interval probability of the new fluent f_r is equal to $\mathbf{p}$.

Similarly, we can take care of *exogenous actions*, i.e., of the situations in which, with a certain probability, a state can change by itself, without any (regular) action being performed.

Due to the possibility of this reduction, in the following text, we will, without losing generality, restrict ourselves to the situations in which the results of actions are deterministic, and the only uncertainty is in the initial state.

The idea of 0-approximation: a motivation for our algorithm

Our feasible plan-checking algorithm for planning under probabilistic uncertainty will be a generalization of the 0-approximation algorithm developed in (Baral and Son 1997) for the case of partial knowledge, i.e., in our terms, for the case when for each fluent $f \in \mathcal{F}$, the initial probability interval is equal to $[0, 0]$, $[1, 1]$, or $[0, 1]$.

In terms of these intervals, the 0-approximation algorithm can be described as follows: To check whether a plan $\alpha = [a_1, \ldots, a_n]$ is successful, for each moment of time $t = 1, \ldots, n$, and for each fluent $f \in \mathcal{F}$, we estimate the interval $\mathbf{p}(f, s_t)$ of possible values of this fluent's probability in the state $s_t = res(a_t, res(a_{t-1}, \ldots, res(a_1, s) \ldots))$. To be more precise, for each t and f, we compute the *enclosure* $\widetilde{\mathbf{p}}(f, s_t) \supseteq \mathbf{p}(f, s_t)$. We start with the known values $\widetilde{\mathbf{p}}(f, s_0) = \mathbf{p}(f, s_0)$; after the estimates $\widetilde{\mathbf{p}}(f, s_t)$ are found for a certain t, we compute the estimates for s_{t+1} as follows: Let $V^+(a, s_t)$ denote the set of all fluent literals F for which D contains a rule "a causes F if $F_1, \ldots, F_m$" for which all the conditions F_i are definitely true (have probability intervals $\widetilde{\mathbf{p}}(F_i, s_t) = [1, 1]$). Let $V^-(a, s_t)$ denote the set of all fluent literals F for which D contains a rule "a causes F if $F_1, \ldots, F_m$" for which all the conditions F_i may be true (have probability intervals $\widetilde{\mathbf{p}}(F_i, s_t) \neq [0, 0]$). Then, for every fluent $f \in \mathcal{F}$:

- We assign $\widetilde{\mathbf{p}}(f, s_{t+1}) := [1, 1]$ if

$f \in V^+(a_{t+1}, s_t) \vee (\widetilde{\mathbf{p}}(f, s_t) = [1, 1] \& \neg f \notin V^-(a_{t+1}, s_t))$.

- We assign $\widetilde{\mathbf{p}}(f, s_{t+1}) := [0, 0]$ if

$\neg f \in V^+(a_{t+1}, s_t) \vee (\widetilde{\mathbf{p}}(f, s_t) = [0, 0] \& f \notin V^-(a_{t+1}, s_t))$.

- In all other cases, we take $\widetilde{\mathbf{p}}(f, s_{t+1}) := [0, 1]$.

It is proven that this algorithm indeed produces an enclosure and thus, if we get $\widetilde{\mathbf{p}}(f, s_n) = [1, 1]$ at the final state, we are thus guaranteed that this plan works.

The 0-approximation algorithm is feasible: its computation time grows linearly with the length of the plan and with the size of the domain description. Since the plan checking problem is NP-hard, it is not surprising that sometimes, this algorithm errs – fails to realize that a given plan is successful.

The new algorithm based on interval computations

In our new algorithm, we will also start with the original estimates $\widetilde{\mathbf{p}}(f, s_0) = \mathbf{p}(f, s_0)$ and produce the values $\widetilde{\mathbf{p}}(f, s_t) \supseteq \mathbf{p}(f, s_t)$ for $t = 1, 2, \ldots, n$.

According to the semantics of the rules, a fluent f holds in the next moment of time iff either it is caused by some rule, or it was true in the previous moment of time, and its negation was not caused by any rule. Thus, in order to find out whether f holds at the moment $t+1$ (after applying the action a_{t+1}), we first need to describe all the rules in which the action a_{t+1} causes either f or $\neg f$. Let us denote by k the total number of rules in which a_{t+1} causes f, and let us denote the conditions of the i-th such rule by $F_{i,1}, \ldots, F_{i,n_i}$. Similarly, let us denote by ℓ the total number of rules in which a_{t+1} cause $\neg f$, and let us denote the conditions of the j-th such rule by $G_{j,1}, \ldots, G_{i,m_j}$. In terms of these notations, f holds at the moment of time $t+1$ iff the following formula B holds at moment t:

$$(F_{1,1} \& \ldots \& F_{1,n_1}) \vee \ldots \vee (F_{k,1} \& \ldots \& F_{k,n_k}) \vee$$
$$\{f \& \neg[(G_{1,1} \& \ldots \& G_{1,m_1}) \vee \ldots \vee (G_{\ell,1} \& \ldots \& G_{\ell,m_\ell})]\}.$$

Thus, if we know the enclosures $\widetilde{\mathbf{p}}(F, s_t)$ for all fluent literals at time t, in order to find the enclosure $\widetilde{\mathbf{p}}(f, s_{t+1})$ for the probability interval $\mathbf{p}(f, s_{t+1})$, it is sufficient to be able to find the enclosure for the above Boolean combination B. Since the Boolean combination B consists of sequential application of propositional connectives $\&$, $\vee$, and $\neg$, it is therefore sufficient to be able to solve the following three auxiliary problems:

- given the enclosure $\widetilde{\mathbf{p}}(F)$ for $\mathbf{p}(F)$, compute the enclosure $\widetilde{\mathbf{p}}(\neg F)$ for $\mathbf{p}(\neg F)$;
- given the enclosures $\widetilde{\mathbf{p}}(F_i)$ $(1 \le i \le n)$ for the intervals $\mathbf{p}(F_i)$, compute the enclosure $\widetilde{\mathbf{p}}(F_1 \& \ldots \& F_n)$ for $\mathbf{p}(F_1 \& \ldots \& F_n)$;
- given the enclosures $\widetilde{\mathbf{p}}(F_i)$ $(1 \le i \le n)$ for the intervals $\mathbf{p}(F_i)$, compute the enclosure $\widetilde{\mathbf{p}}(F_1 \vee \ldots \vee F_n)$ for $\mathbf{p}(F_1 \vee \ldots \vee F_n)$.

The following propositions solve these problems:

Proposition 1. *If* $\mathbf{p}(F) \subseteq \widetilde{\mathbf{p}}(F)$, *then* $\mathbf{p}(\neg F) \subseteq \widetilde{\mathbf{p}}(\neg F) \stackrel{\text{def}}{=} [1 - \widetilde{p}^+(F), 1 - \widetilde{p}^-(F)]$.

Proposition 2. *If* $\mathbf{p}(F_i) \subseteq \widetilde{\mathbf{p}}(F_i)$ *for* $i = 1, \ldots, n$, *then* $\mathbf{p}(F_1 \& \ldots \& F_n) \subseteq \widetilde{\mathbf{p}}(F_1 \& \ldots \& F_n) \stackrel{\text{def}}{=} [\max(0, \widetilde{p}^-(F_1) + \ldots + \widetilde{p}^-(F_n) - n + 1), \min(\widetilde{p}^+(F_1), \ldots, \widetilde{p}^+(F_n))]$, *and* $\mathbf{p}(F_1 \vee \ldots \vee F_n) \subseteq \widetilde{\mathbf{p}}(F_1 \vee \ldots \vee F_n) \stackrel{\text{def}}{=} [\max(\widetilde{p}^-(F_1), \ldots, \widetilde{p}^-(F_n)), \min(1, \widetilde{p}^+(F_1) + \ldots + \widetilde{p}^+(F_n))]$.

One can prove that the estimates provided by Propositions 1, 2 are indeed the narrowest possible.

By using these estimates, we can find, step-by-step, the enclosure for $\mathbf{p}(f, s_{t+1})$ and thus, the desired enclosure for the interval $\mathbf{p}(f, s_n)$ which describes the probability of success in the final state.

Let us give a simple example. Let a domain description D consist of the following three rules: a causes f if g, h; a causes f if k; a causes $\neg f$ if j, k. The objective is f, the checked plan consists of the single action a. In this case, the validity of f at the final moment of time s_1 is equivalent to the validity of the following propositional formula at the moment s_0: $(g \& h) \vee k \vee \{f \& \neg[j \& k]\}$. If initially, $\mathbf{p}(f, s_0) = [0.1, 0.2]$, $\mathbf{p}(g, s_0) = [0.7, 0.9]$, $\mathbf{p}(h, s_0) = [0.6, 0.7]$, $\mathbf{p}(j, s_0) = [0.2, 0.3]$, and $\mathbf{p}(k, s_0) = [0.7, 0.9]$, then we can compute the enclosure $\widetilde{\mathbf{p}}(f, s_1)$ as follows:

- $\widetilde{\mathbf{p}}(g \& h, s) =$ $[\max(0, 0.7 + 0.6 - 1), \min(0.9, 0.7)] = [0.3, 0.7]$;
- $\widetilde{\mathbf{p}}(j \& k, s_1) =$ $[\max(0, 0.2 + 0.4 - 1), \min(0.3, 0.6)] = [0, 0.3]$;
- $\widetilde{\mathbf{p}}(\neg[j \& k], s) = [1 - 0, 1 - 0.3] = [0.7, 1]$;
- $\widetilde{\mathbf{p}}(f \& \neg[j \& k], s) =$ $[\max(0, 0.1 + 0.7 - 1), \min(0.2, 1)] = [0, 0.2]$;
- $\widetilde{\mathbf{p}}((g \& h) \vee k) \vee \{f \& \neg[j \& k]\}) =$ $[\max(0.3, 0.7, 0), \min(1, 0.7 + 0.9 + 0.2)] = [0.7, 1]$.

Thus, if $p_0 \le 0.7$, this plan is successful; otherwise, we cannot guarantee its success with a given probability.

The soundness of the above algorithm can be formulated in precise terms:

Definition 1. *By a planning problem, we mean the tuple* $\langle \mathcal{F}, \mathcal{A}, D, \{\mathbf{p}(f, s_0)\}_{f \in \mathcal{F}}, f, p_0 \rangle$, *where:*

- $\mathcal{F}$ *is a finite set whose elements are called fluents;*
- $\mathcal{A}$ *is a finite set whose elements are called actions;*
- D *is a finite set of expression of the type "*a causes F if $F_1, \ldots, F_m$*", where* F *and* F_i *are fluent literals, i.e., fluents or their negations;*
- *each* $\mathbf{p}(f, s_0)$ *is a sub-interval of the interval* $[0, 1]$*;*
- $f \in \mathcal{F}$ *is called objective, and* $p_0 \in [0, 1]$.

A sequence of actions $\alpha = [a_1, \ldots, a_n]$ *is called a plan.*

By a plan checking problem we mean a pair consisting of a planning problem and a plan α.

We say that a probability distribution P *on the set of all initial states is consistent with the planning problem if* $P(f) \in \mathbf{p}(f, s_0)$ *for every fluent* f.

We say that a plan is successful if $p(f, res(\alpha, s_0)) \ge p_0$ *for every probability distribution* P *with is consistent with the planning problem.*

Proposition 3. *For every plan checking problem, for every probability distribution* P *on the set of all initial states which is consistent with the given interval probabilities, the probability* $P(f, res(\alpha, s_0))$ *is contained in the interval* $\widetilde{\mathbf{p}}(f, s_n)$ *computed by the above algorithm.*

(Proof is by induction over the length of the plan, similarly to the proof about 0-approximation.)

Corollary. *If the above algorithm tells that the plan is successful (i.e., if* $\widetilde{p}^-(f, s_n) \ge p_0$*), then this plan is indeed successful.*

Definition 2. *We say that a planning problem corresponds to incomplete information if for every fluent* f, *the interval* $\mathbf{p}(f, s_0)$ *is equal to* $[0, 0]$, $[1, 1]$, *or* $[0, 1]$.

Proposition 4. *For planning problems corresponding to incomplete information, the above algorithm coincides with the* 0*-approximation algorithm.*

Comments.

1) For degenerate intervals $\widetilde{\mathbf{p}}(F_i) = [p_i, p_i]$, we get $[\max(p_1 + p_2 - 1, 0), \min(p_1, p_2)]$ as $\widetilde{\mathbf{p}}(F_1 \& F_2)$, and $[\max(p_1, p_2), \min(1, p_1 + p_2)]$ for $\vee$. Both lower and upper bounds are particular cases of the operations used in the fuzzy approach (Congdon et al. 1993; Saffiotti et al. 1995); thus, we get a new justification for this approach.

2) The above step-by-step approach to getting guaranteed estimates can be viewed as a particular case of *interval computations*, i.e., computations in which the

input is only known with interval uncertainty (see, e.g., (Hammer et al. 1993; Kearfott et al. 1996)). Interval computations have been used, together with more traditional AI techniques, to produce a robot which won 1st place at the AAAI'97 robot competition (Baral and Son 1997a; Baral et al. 1998; Morales et al. 1998).

3) Estimates obtained by using interval computations are often overestimations, because when we compute the probability intervals for the next moment of time, we assume that the previous intervals are "independent", while in reality, they come from the same source and may therefore be dependent. For example, if we have two rules "a causes f if f_1" and "a causes f if $\neg f_1$", with $p(f_1, s_0) = [0.6, 0.7]$ and with f initially false, then after the action a, f is always true ($\mathbf{p}(f, s_1) = [1, 1]$). In this case, our algorithm finds f as $f_1 \vee \neg f_1$, so we get $\widetilde{\mathbf{p}}(\neg f_1, s_0) = [0.3, 0.4]$, and $\widetilde{\mathbf{p}}(f, s_1) = [\max(0.6, 0.3), \min(0.6+0.4, 1)] = [0.6, 1]$. To take this dependence into consideration, we can use *generalized* (affine) interval computations (Hansen 1975).

4) If we are sure that all the probabilities are independent, then we can use a feasible technique – Monte-Carlo simulations (see, e.g., (Kreinovich et al. 1994)).

Conclusions

In this paper, we show that methods of interval computations can be used to get a feasible approximation to plan checking under probabilistic uncertainty. The resulting method is a natural generalization of 0-approximation proposed earlier to describe planning in the case of partial knowledge. It turns out that some of the resulting probabilistic techniques coincides with heuristically proposed "fuzzy" methods. Thus, we justify these fuzzy heuristics as a reasonable feasible approximation to the (NP-hard) probabilistic problem.

Acknowledgments

This work was supported in part by NASA grant NCC5-209, by NSF grants DUE-9750858 and CDA-9522207, by United Space Alliance grant NAS 9-20000 (PWO C0C67713A6), by the US Air Force grant F49620-95-1-0518, and by the National Security Agency grant MDA904-98-1-0561.

The authors are thankful to anonymous referees for valuable comments.

References

J. Allen, J. Hendler, A. Tate, *Readings in Planning*, Morgan Kauffman, San Mateo, CA, 1990.

C. Baral *et al.*, "From theory to practice: The UTEP robot in AAAI 96 and AAAI 97 robot contests", *Proc. 2nd International Conference on Autonomous Agents (Agents'98)*, 1998, pp. 32-38.

C. Baral, V. Kreinovich, and R. Trejo, "Computational complexity of planning and approximate planning in presence of incompleteness", in: *Proc. IJCAI'99*, Vol. 2, pp. 948–953 (full text to appear in *Artificial Intelligence*).

C. Baral and T. Son, "Approximate reasoning about actions in presence of sensing and incomplete information", In: *Proc. of International Logic Programming Symposium (ILPS'97)*, 1997, pp. 387–401.

C. Baral and T. Son, "Regular and special sensing in robot control – relation with action theories", *Proc. AAAI 97 Workshop on Robots, Softbots, and Immobots – Theories of Action, Planning and Control*, 1997a.

C. Congdon *et al.*, "Carmel vs. Flakey: A comparison of two winners," *AI Magazine*, 1993, Vol. 14, No. 1, pp. 49–57.

A. Doan and P. Haddawy, "Sound Abstraction of Probabilistic Actions in the Constraint Mass Assignment Framework ", *Proc. UAI'96*, pp. 228–235.

D. Draper and S. Hanks, "Localized Partial Evaluation of Belief Networks", *Proc. UAI'94*.

R. Givan, S. Leach, and T. Dean, "Bounded parameter Markov decision processes", *Proc. 4th European Conference on Planning*, Toulouse, France, 1997.

R. Hammer *et al.*, *Numerical Toolbox for Verified Computing I*, Springer-Verlag, 1993.

E. R. Hansen, "A generalized interval arithmetic", In: K. Nickel (ed.), *Interval mathematics*, Lecture Notes in Computer Science, 1975, Vol. 29, pp. 7–18.

H. Geffner and B. Bonet, "High-Level Planning and Control with Incomplete Information Using POMDPs", *Proc. AIPS-98 Workshop on Integrating Planning, Scheduling and Execution in Dynamic and Uncertain Environments*, 1998.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains", *Artificial Intelligence*, 1998, Vol. 101, pp. 99–134.

R. B. Kearfott and V. Kreinovich (eds.), *Applications of Interval Computations*, Kluwer, Dordrecht, 1996.

V. Kreinovich *et al.*, "Monte-Carlo methods make Dempster-Shafer formalism feasible." In R. R. Yager *et al.* (Eds.), *Advances in the Dempster-Shafer Theory of Evidence*, Wiley, N.Y., 1994, pp. 175–191.

M. L. Littman, J. Goldsmith, and M. Mundhenk, "The Computational Complexity of Probabilistic Planning", *JAIR*, 1998, Vol. 9, pp. 1–36.

J. D. Lowrance and D. E. Wilkins, "Plan evaluation under uncertainty," *Proc. Workshop on Innovative Approaches to Planning, Scheduling and Control*, Morgan Kaufmann, San Francisco, 1990, pp. 439–449.

D. Morales and Tran Cao Son, "Interval Methods in Robot Navigation", *Reliable Computing*, 1998, Vol. 4, No. 1, pp. 55–61.

A. Saffiotti, K. Konolige, and E. H. Ruspini, "A multivalued-logic approach to integrating planning and control", *Artificial Intelligence*, 1995, Vol. 76, No. 1–2, pp. 481–526.