Probabilistic Plan Verification through Acceptance Sampling

Håkan L. S. Younes[*]
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, U.S.A.
lorens@cs.cmu.edu

David J. Musliner
Automated Reasoning Group
Honeywell Laboratories
3660 Technology Drive
Minneapolis, MN 55418, U.S.A.
musliner@htc.honeywell.com

Abstract

CIRCA is an architecture for real-time intelligent control. The CIRCA planner can generate plans that are guaranteed to maintain system safety, given certain timing constraints. To prove that its plans guarantee safety, CIRCA relies on formal verification methods. However, in many domains it is impossible to build 100% guaranteed safe plans, either because it requires more resources than available, or because the possibility of failure simply cannot be eliminated. By extending the CIRCA world model to allow for uncertainty in the form of probability distribution functions, we can instead generate plans that maintain system safety with high probability. This paper presents a procedure for probabilistic plan verification to ensure that heuristically-generated plans achieve the desired level of safety. Drawing from the theory of quality control, this approach aims to minimize verification effort while guaranteeing that at most a specified proportion of good plans are rejected and bad plans accepted.

Introduction

Realistic domains for autonomous agents present a broad spectrum of uncertainty, including uncertainty in external events and in the outcome of internally-selected actions. Planning to achieve system goals and maintain safety in the face of this uncertainty can be highly challenging. We can attempt to generate a universal plan (cf. (Schoppers 1987)), taking every contingency into account, but with limited resources this may be futile. A more advantageous approach may be to quantify the uncertainty and incorporate this information into the reasoning process. This enables us to set an arbitrary failure threshold for plans, and we can reject plans with failure probability above the threshold. Furthermore, the additional information can be used to focus our planning efforts on situations we are more likely to encounter (cf. (Atkins, Durfee, & Shin 1996)).

In this paper we introduce a probabilistic extension to CIRCA—an architecture for real-time control—(Musliner, Durfee, & Shin 1993). The original planner in CIRCA builds reactive control plans that achieve system goals and maintain system safety, subject to strict time bounds and models of the dynamic external world (the environment). While the original CIRCA model includes nondeterminism in the outcome of actions and uncertainty about the timing and occurrence of externally-caused transitions, it does not have quantified uncertainty information. The extended model includes quantified uncertainty in the form of probability distributions on the timing of different transitions. This allows CIRCA to build plans that are not completely guaranteed to prevent failure—plans may allow for the possibility of failure, as long as the failure probability is below some threshold. The world model of the probabilistic extension corresponds to a generalized semi-Markov process.

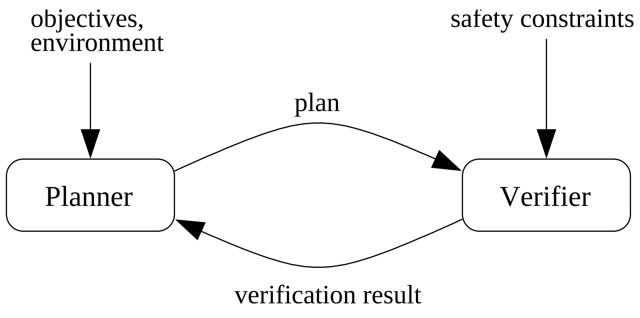


Figure 1: Interaction between planner and verifier.

To verify that a potential plan meets the necessary safety threshold, we have developed an efficient probabilistic plan verifier that decides whether to accept or reject a plan. Our verifier uses Monte Carlo simulation, or more precisely discrete event simulation, to generate sample execution paths given a plan and a world model. Requiring relatively few sample paths, the verifier can guarantee that at most a specified proportion of good plans are rejected and bad plans accepted.

The interaction between planner and verifier is depicted in Figure 1. The planner generates a plan to achieve given objectives in a given dynamic real-time environment. The plan is passed to the verifier that tests if the plan satisfies given safety constraints. The result of the verification is passed to the planner, which based on the given information decides whether the generated plan needs to be revised. If the plan passed the verification, then no revision is needed, and otherwise the verification result is used to guide plan revision. We are not concerned with how to generate and revise plans in this paper, but only how to verify plans.

[*]The work reported in this paper was performed during a summer internship at Honeywell Laboratories.

The World Model

Musliner, Durfee, & Shin (1993) introduced a formalization of the CIRCA world model, and later extended it (Musliner, Durfee, & Shin 1995). We will deviate slightly from their formalization here. The purpose is to enable us to view the CIRCA world model as a *generalized semi-Markov process* (GSMP), which is a formalism for discrete event systems introduced by Matthes (1962) (see also (Glynn 1989; Shedler 1993)). The formal non-probabilistic world model has seven elements $\langle S, S_F, S_0, T, E, min\Delta, max\Delta \rangle$:

1. A finite set of states S, where each state represents a description of relevant features.
2. A set of failure states $S_F \subset S$, which consists of all states in S that violate domain constraints or control-level goals.
3. A set of possible initial states $S_0 \subset S$.
4. A finite set of transitions $T = T_E \cup T_A \cup T_T$, where T_E is a set of event transitions representing world occurrences as instantaneous state changes, T_A is a set of action transitions representing actions performed by the run-time system, and T_T is a set of temporal transitions representing the progression of time.
5. A function $E : S \to 2^T$ mapping a state s to a set of transitions enabled in s.
6. A function $min\Delta : T \to \mathbb{R}$ mapping transitions to minimum trigger times.
7. A function $max\Delta : T \to \mathbb{R}$ mapping transitions to maximum trigger times.

Each transition $\tau \in T$ is a mapping between states; $\tau : S \to S$.

For a given planning problem, the environment is represented by a world model M_{env} without any action transitions. Given M_{env}, the planner generates a plan (or *policy*) π, which is a mapping from states to action transitions. The composition of M_{env} and π is a stochastic process representing the execution of π in the given environment. When verifying that a plan π is safe, we are really verifying that certain properties hold for the stochastic process representing the composition of π and the environment model. Figure 2(a) shows an environment for an unmanned aerial vehicle. In Figure 2(b), actions constituting a plan have been added to the environment.

Model Dynamics

At any particular point in time, the world is considered to occupy a single state in the model. The initial world state can be any state $s \in S_0$. The world state changes when a transition is triggered. If the current state is s and transition τ is triggered, the next state is given by $\tau(s)$. Not all transitions are necessarily enabled in all states. For each state $s \in S$, $E(s)$ is the subset of T denoting the set of transitions that can be triggered in state s. Only one transition can be triggered in each state at any given time, so transitions in $E(s)$ compete to trigger a state change.

We can associate a clock $r_{s,\tau}$ with each enabled transition τ in a state s, showing the time remaining until τ is scheduled to occur in s. The clock value $r_{s,\tau}$ is called the *residual lifetime* of τ in s (Glynn 1989). When a transition τ^* is triggered in state s, causing a transition to state $s' = \tau'(s)$, then the lifetimes of the transitions enabled in s' are initialized as follows:

1. If $\tau \in E(s) \setminus \{\tau^*\}$, then let $r_{s',\tau} = r_{s,\tau} - r_{s,\tau^*}$.
2. If $\tau \notin E(s) \setminus \{\tau^*\}$, then $r_{s',\tau}$ is set to some value in the interval $[min\Delta(\tau), max\Delta(\tau)]$.

The type of a transition determines the general form of the interval $[min\Delta(\tau), max\Delta(\tau)]$. Event transitions can occur at any time, and thus have a lower limit of zero and an upper limit of infinity. Temporal transitions are similar to events, but can have a non-zero lower limit. An action transition represents an action taken by the run-time system, and has a finite upper limit representing the worst-case execution time for that action.

Note that with the formalism given here, the possible trigger time of a transition in a given state at a given time can depend on the history of state transitions, making the world model non-Markovian.

Probabilistic Extension

The world model, as presented so far, has limited expressive power. We can say that an event may occur in a state s by representing the event with a transition τ which is enabled in s, and we can bound the time that the world must stay in s before the event may (or must) occur. We can, however, say nothing about the *expected* time that the world must stay in state s before the event occurs. There is no way to distinguish more frequently occurring events, such as rain delaying a tennis match at Wimbledon, from far less frequent events, such as a player being struck by lightning.

A natural extension is to associate a probability distribution function $F(t;\tau)$ with each transition τ, giving the probability that τ will be triggered t time units after it was last enabled. We can easily define the previously used interval limits in terms of F:

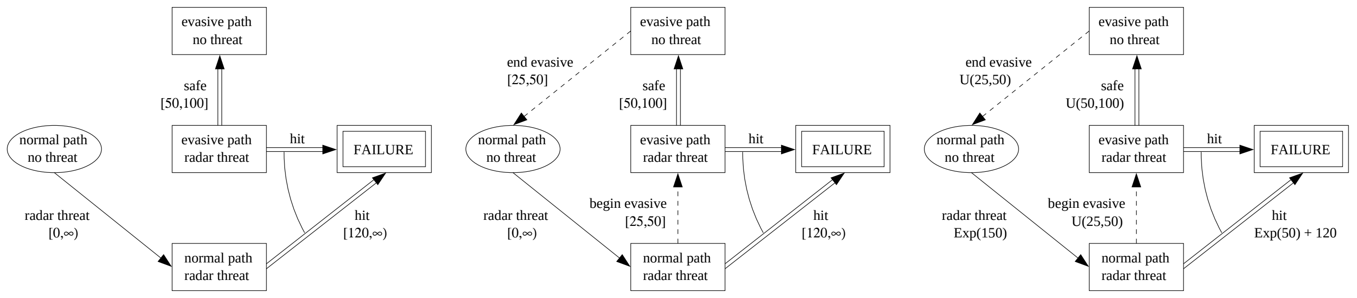
$$min\Delta(\tau) = \sup\{t \mid F(t;\tau) = 0\}$$
$$max\Delta(\tau) = \inf\{t \mid F(t;\tau) = 1\}$$

We require that $F(0;\tau) = 0$ (i.e. the distribution function corresponds to a positive random variable), because no transition can be triggered before it has been enabled. A typical choice of distribution for an event transition would be an exponential distribution, and for a temporal transition a shifted exponential distribution. For an action transition one could, for example, use a uniform distribution or a truncated normal distribution.

In addition we can replace S_0 with a probability distribution p_0 over S, where $p_0(s)$ is the probability that the world starts in state s. The set of possible initial states is then simply

$$S_0 = \{s \mid p_0(s) > 0\}.$$

This way we obtain a probabilistic world model consisting of six elements $\langle S, S_F, p_0, T, E, F \rangle$. To make this a GSMP we need to define transition probabilities $p(s'; s, \tau)$ expressing the probability of the next state being s' given



(a) Environment. (b) Plan and environment. (c) Probabilistic world model.

Figure 2: Three different world models. To the left is a world model representing the environment. The initial state is drawn as an oval, and there is only one failure state. The solid arrow represents an event transition, and the double arrows represent temporal transitions. The arc between the two "hit" transitions indicate that they are in fact the same transition enabled in two different states. In the middle, actions (dashed arrows) constituting a plan have been added to the environment. To the right is the same plan/environment model, but with probability distributions associated with each transition instead of just intervals.

that τ is triggered in state s. This is straightforward, however, because the next state is determined by $\tau(s)$. We thus get the following:

$$p(s'; s, \tau) = \begin{cases} 1 & \text{if } \tau(s) = s' \\ 0 & \text{otherwise} \end{cases}$$

In addition, we set all clock speeds $r(s, \tau)$ to 1. The elements (S, p_0, p, T, E, F, r) constitute a GSMP. In fact, the form of the state transition probabilities and the probability distribution functions makes this a *time-homogeneous* GSMP (Glynn 1989).

Figure 2(c) shows the probabilistic version of the model in Figure 2(b). Instead of an interval, a probability distribution function is given for each transition.

Plan Verification

In the non-probabilistic world model, a safe plan is one where no state $s \in S_F$ is reachable from the set of possible initial states S_0. Action transitions are planned to "preempt" temporal transitions to failure states. Transition τ_1 preempts τ_2 in state s if it can be proven that τ_1 will always be triggered before τ_2, independent of the history of state transitions. This immediately rules out plans that have event transitions to failure states. Because event transitions can trigger instantaneously, no other transition can preempt them. Safety of a plan can be verified by applying certain correctness-preserving model transformations, pruning out unreachable states (Musliner, Durfee, & Shin 1995). As was shown above, S_0, $min\Delta$, and $max\Delta$ can be extracted from the probabilistic world model, enabling us to verify probabilistic plans using the same technique. This would, however, be a waste of all the extra information available to us regarding the stochastic behavior of transitions.

In probabilistic terms, the above technique can only distinguish between zero and non-zero probability of reaching a set of states. Plans with non-zero probability of reaching a failure state are considered unsafe. With probability distribution functions available for the transitions, we can set an arbitrary threshold θ representing the highest acceptable failure probability of a plan. Setting $\theta = 0$ we revert to the old model. With $\theta > 0$, though, we can accept plans that would have otherwise been discarded. For example, it now becomes possible to have a plan with event transitions to failure states provided that the events represented by these transitions are sufficiently infrequent, or the probability of entering a states in which such events are enabled is sufficiently low.

As an example, consider the plan in Figure 2(b). The "begin evasive" action preempts the "hit" transition in the bottom state, but the "hit" transition can still be triggered in the center state. If, for example, the lifetime of "begin evasive" is 50 time units, the lifetime of "safe" is 100 time units, and the lifetime of "hit" is 120 time units, then "hit" will trigger before "safe" in the center state, causing a transition to the failure state. The plan in Figure 2(c) is the same as in Figure 2(b), but the intervals have been substituted for probability distribution functions. We will see later that this plan is acceptable with a failure threshold of 0.05, and an upper limit on the execution time set to 200 time units.

We use an acceptance sampling algorithm to probabilistically determine if a plan should be accepted. The samples used by the algorithm are sample execution paths generated through discrete event simulation.

Generating Sample Execution Paths

A plan π, when executed, represents a stochastic process $\{X_\pi(t) \mid t \geq 0\}$, where $X_\pi(t)$ is the state of the world t time units after the plan is set in action. We are interested in determining whether the probability of visiting a failure state $s \in S_F$ within a specified time limit $t_{\max}$ from the start of the execution of π is below a given threshold θ.

For any one state s, the probability of visiting s before time $t_{\max}$ is $\Pr[\inf\{t \mid X_\pi(t) = s\} \leq t_{\max}]$. For a set of absorbing states, such as the set of failure states S_F, at most one state in the set can be visited during execution.

Furthermore, if $X_\pi(t) = s$ for an absorbing state s at time t, then $X_\pi(t') = s$ for all $t' \geq t$. The failure probability of a plan after $t_{\max}$ time units is therefore

$$\Pr[\inf\{t \mid X_\pi(t) \in S_F\} \leq t_{\max}] = \sum_{s \in S_F} \Pr[X_\pi(t_{\max}) = s].$$

Depending on the stochastic characteristics of the process, an analytical calculation of the failure probability may be impossible, and the only feasible approach may be to use sampling techniques that make use of simulation to generate sample paths (Heidelberger 1995).

Let us define the random variable $Y_\pi(s)$ as follows:

$$Y_\pi(s) = \begin{cases} 1 & \text{if } \inf\{t \mid X_\pi(t) = s\} \leq t_{\max} \\ 0 & \text{otherwise} \end{cases}$$

Clearly, $Y_\pi(s)$ is a binomial variate with parameters $1, p_s$; p_s being the probability that state s is visited within $t_{\max}$ time units. Given n samples $y_1(s)$ through $y_n(s)$ of $Y_\pi(s)$, let $x_n(s) = |\{y_i(s) \mid y_i(s) = 1\}|$. This is the number of samples in which state s is visited within $t_{\max}$ time units. A point estimate of p_s is $x_n(s)/n$. This estimate can be used as a heuristic to guide the effort of the planner towards the most likely states (cf. (Atkins, Durfee, & Shin 1996)). Note that $x_n(s)$ can be computed as $\sum_{i=1}^{n} y_i(s)$. Let f_n denote the number of failures observed in n samples. Clearly

$$f_n = \sum_{s \in F} \sum_{i=1}^{n} y_i(s). \tag{1}$$

Each sample $y_i(s)$ is generated using discrete event simulation. Algorithm 1 outlines a procedure for generating $y_i(s)$ for each state $s \in S' \subset S$ simultaneously (cf. Algorithm 4.17 of (Shedler 1993)).

The sample generation algorithm does not consider the case of two transitions being triggered simultaneously. If all distribution functions F are continuous, the probability of this happening is in fact zero. Yet when implementing the sampling algorithm on a computer, where real numbers are represented by finite precision floating-point numbers, the occurrence of simultaneous transitions becomes an issue we have to deal with. We can address this by modifying step 4, so that instead of simply letting $s' = \tau^*(s)$, we select s' with uniform probability from the set of transitions with the shortest residual lifetime in s.

Acceptance Sampling

A plan π, when executed, can either fail or succeed. We denote the probability of failure by p_F. As alluded to earlier, we can specify a positive threshold θ representing the maximum acceptable failure probability. If p_F does not exceed θ we are willing to accept π, but would reject it otherwise. Deciding whether to accept or reject a plan can be cast as the problem of testing the hypothesis $p_F \leq \theta$ against the alternative hypothesis $p_F > \theta$. This is an important problem in manufacturing industry and engineering, and has been studied thoroughly in the field of statistical quality control (cf. (Chorafas 1960; Montgomery 1991)). The problem also arises in the area of software certification (Poore, Mills, & Mutchler 1993).

Algorithm 1 Procedure for generating $y_i(s)$ for all $s \in S' \subset S$ simultaneously.

```
1. Let $t = 0$ and $y_i(s) = 0$ for all $s \in S'$. Generate an initial
   state $s$ in accordance with the probability distribution $p_0$.
2. For each transition $\tau \in E(s)$, sample a residual lifetime
   $r_{s,\tau}$ (cf. (Glynn 1989)) according to the distribution function
   $F(\cdot;\tau)$. This is the amount of time remaining until $\tau$
   triggers a transition out of state $s$.
3. Set $y_i(s)$ to 1. Terminate the simulation if $s \in S_F$ or
   $E(s) \cap T_A = \emptyset$.
4. Let $\tau^*$ be the transition with the shortest residual lifetime
   in $s$. Terminate the simulation if $t + r_{s,\tau^*} > t_{\max}$. Otherwise,
   let $s' = \tau^*(s)$.
5. Generate new residual lifetimes for transitions $\tau \in E(s')$
   as follows:
   • If $\tau = \tau^*$ or $\tau \notin E(s)$, then sample $r_{s',\tau}$ according to
     $F(\cdot;\tau)$.
   • Otherwise, let $r_{s',\tau} = r_{s,\tau} - r_{s,\tau^*}$.
6. Set $s$ to $s'$ and $t$ to $t + r_{s,\tau^*}$, and go to step 3.
```

Risk Tolerance and Hypothesis Testing. We would ideally like to accept only those plans with a failure probability no larger than θ and reject all other plans. In general, however, we cannot calculate the failure probability of a plan analytically, nor can we determine it with absolute certainty if falling back on sampling techniques. In the latter case the reason is the potentially infinite sample space. Therefore we must tolerate a certain risk of rejecting a plan with true failure probability at most θ, or accepting a plan with failure probability above θ. In statistical quality control the former kind of error is referred to as a type I error (reject when acceptable), and the latter a type II error (accept when rejectable). We associate a risk level with each type of error. The risk levels are denoted by α and β respectively, and these represent the acceptable probability of making an error of respective type.

Let H_0 be the hypothesis that $p_F \leq \theta$ (*null hypothesis*), and let H_1 be the alternative hypothesis that $p_F > \theta$. We would like to test the hypothesis H_0 against H_1 so that the probability of accepting H_1 when H_0 holds is at most α, and the probability of accepting H_0 when H_1 holds is at most β.

In order to be able to choose α and β freely, however, we need to relax the hypotheses somewhat.[1] For this purpose we introduce an indifference region of non-zero width δ. Let $p_F \leq \theta - \delta$ be H_0 and let $p_F \geq \theta + \delta$ be H_1. We use acceptance sampling to test hypothesis H_0 against H_1. The motivation for the indifference region, other than that it allows us to choose the two risk levels independently, is that if the true failure probability is sufficiently close to the threshold, then we are indifferent to whether the plan is accepted or rejected.

[1]We would have to choose $\alpha = 1 - \beta$ without the suggested relaxation, which means if one of the risk levels was low, then the other would have to be high.

Sequential Sampling. A sequential test is one where the number of observations is not predetermined but is dependent on the outcome of the observations (Wald 1945). Wald (*loc. cit.*) develops the theory of *sequential analysis*, and defines the sequential probability ratio test (see also (Wald 1947)), which is optimal for testing a simple hypothesis against a simple alternative in the sense that it minimizes the expected number of samples needed to reach a decision.

Let X be a binary random variable with unknown parameter p such that $\Pr[X = 1] = p$. The sequential probability ratio test is carried out as follows to test the hypothesis H_0 that $p \leq \theta - \delta$ against the hypothesis H_1 that $p \geq \theta + \delta$. At each stage of the test, calculate the ratio

$$\frac{p_{1n}}{p_{0n}} = \frac{\prod_{i=1}^{n} \Pr[X = x_i \mid p = \theta + \delta]}{\prod_{i=1}^{n} \Pr[X = x_i \mid p = \theta - \delta]},$$

where x_i is the sample of X generated at stage i. Accept H_1 if

$$\frac{p_{1n}}{p_{0n}} \geq \frac{1-\beta}{\alpha}.$$

Accept H_0 if

$$\frac{p_{1n}}{p_{0n}} \leq \frac{\beta}{1-\alpha}.$$

Otherwise, generate an additional sample and repeat the termination test. This test procedure respects the risk levels α and β.[2]

Let $\theta_0 = \theta - \delta$ and $\theta_1 = \theta + \delta$. Applied to the problem of validating a plan, if at stage n we have observed f_n failures, the ratio to compute is

$$\frac{p_{1n}}{p_{0n}} = \frac{\theta_1^{f_n}(1-\theta_1)^{n-f_n}}{\theta_0^{f_n}(1-\theta_0)^{n-f_n}}.$$

For purposes of practical computation we work with logarithms, and carry out the test as follows. At the inspection of the nth sample, compute

$$\log \frac{p_{1n}}{p_{0n}} = f_n \log \frac{\theta_1}{\theta_0} + (n - f_n) \log \frac{1-\theta_1}{1-\theta_0}.$$

Continue sampling if

$$\log \frac{\beta}{1-\alpha} < \log \frac{p_{1n}}{p_{0n}} < \log \frac{1-\beta}{\alpha}.$$

Terminate by accepting hypothesis H_1 if

$$\log \frac{p_{1n}}{p_{0n}} \geq \log \frac{1-\beta}{\alpha}.$$

Terminate by accepting H_0 if

$$\log \frac{p_{1n}}{p_{0n}} \leq \log \frac{\beta}{1-\alpha}.$$

Alternatively, we can compute an acceptance number a_n and a rejection number r_n. We accept H_0 if $f_n \leq a_n$, reject H_0 (accept H_1) if $f_n \geq r_n$, and continue sampling otherwise. Let

$$u = \log \frac{\theta_1}{\theta_0} \quad \text{and} \quad v = \log \frac{1-\theta_0}{1-\theta_1}.$$

The acceptance number at stage n is

$$a_n = \frac{\log \frac{\beta}{1-\alpha} + nv}{u+v}, \qquad (2)$$

and the rejection number is

$$r_n = \frac{\log \frac{1-\beta}{\alpha} + nv}{u+v}. \qquad (3)$$

Verification Algorithm

We now have all the pieces needed to specify a plan verification algorithm. Algorithm 2 describes the steps of the procedure. The algorithm uses Wald's sequential probability ratio test, and so has input parameters θ, δ, α, and β. In addition, the parameter $t_{\max}$ needs to be specified for the sample generation algorithm used by the verification procedure.

Algorithm 2 Procedure for verifying plan π.

```
1. Let $n = 0$.
2. Increment $n$ by one and generate samples $y_n(s)$ of $Y_\pi(s)$
   for all $s \in S_F$ (Algorithm 1).
3. Compute $f_n$ (equation (1)).
4. Compute $a_n$ (equation (2)) and $r_n$ (equation (3)).
5. Accept $\pi$ if $f_n \leq a_n$, and reject $\pi$ if $f_n \geq r_n$. Otherwise
   go to step 2.
```

Figure 3 graphically represents the execution of the verification algorithm on the plan in Figure 2(c) using parameters $\theta = 0.05$, $\delta = 0.01$, $\alpha = \beta = 0.05$, and with $t_{\max}$ set to 200 time units. The acceptance and rejection lines correspond to equations 2 and 3 respectively. The curve starting out between the two lines represents the number of observed failures. After generating 201 sample execution paths (of which 3 ended in a failure state), the curve crosses the acceptance line, which means we accept the plan. Had the curve crossed the rejection line instead, we would have rejected the plan. With the given parameters, we are 95% confident that the true failure probability of the plan is less than 0.06.

The number of samples n that the algorithm needs to inspect before a decision is reached does not have a definite upper bound. Wald (1947) proves that the sequential probability ratio test terminates with probability 1. Although the probability is small that the required sample size will exceed twice or three times the expected number of required samples, it may be desirable to set an upper bound $n_{\max}$ in some cases. If an upper bound is provided and the test does not terminate for $n \leq n_{\max}$, Wald (*loc. cit.*) suggests that the null hypothesis be rejected if $f_{n_{\max}} \geq (a_{n_{\max}} + r_{n_{\max}})/2$ and accepted otherwise. If the upper bound is set sufficiently high (e.g. three times the expected value of n), then truncating the process has negligible effect on the strength of the test.

[2] There is a slight approximation involved in the stopping criteria of the test. See (Wald 1945) for details.

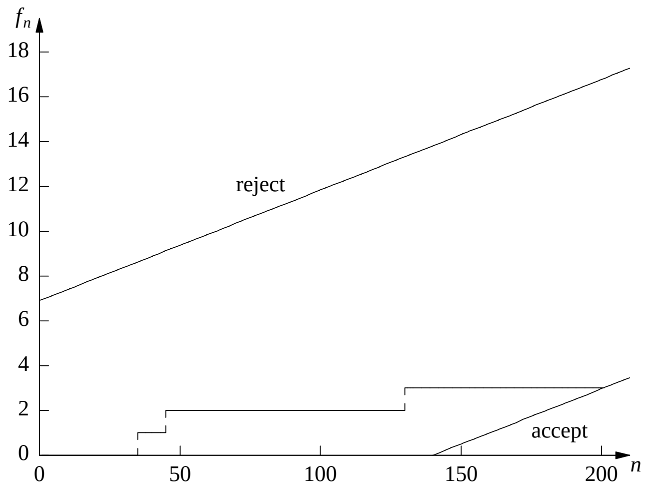


Figure 3: Result of executing the verification algorithm on the plan in Figure 2(c).

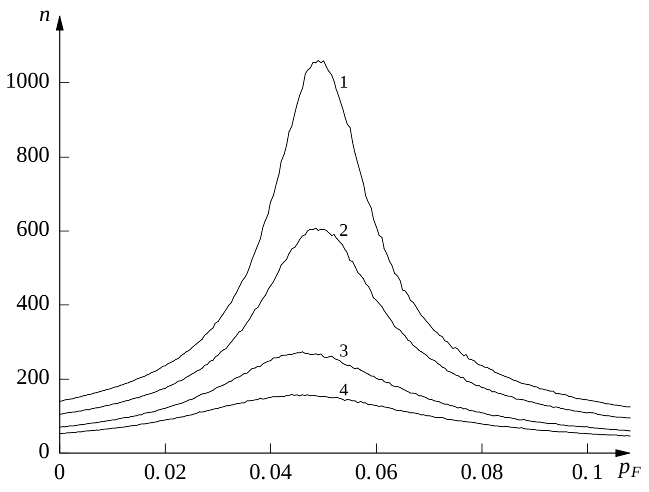


Figure 4: Average number of samples for $\delta = 0.01$ and (1) $\alpha = \beta = 0.05$ and (2) $\alpha = \beta = 0.10$, and for $\delta = 0.02$ with (3) $\alpha = \beta = 0.05$ and (4) $\alpha = \beta = 0.10$ ($\theta = 0.05$ in all cases).

Performance

The performance of our verification algorithm depends on several factors. We can separate these factors into two groups—domain dependent and domain independent.

The domain dependent factors affect the performance of Algorithm 1, used for generating the samples $y_i(s)$. The main factors of this kind are the time period considered ($t_{\max}$) and the mean values of the distribution functions F. If the mean values are small relative to $t_{\max}$, the number of transitions triggering before the simulation terminates will be high, hence increasing the time needed to generate each set of samples. Note, however, that the size of the state space plays a minimal role. Only once, in the initialization step, do we need to perform work at most linear in $|S|$. This work amounts to clearing all the $y_i(s)$'s, which can be done quite efficiently using a bit-vector to represent each set of samples.

As a domain independent factor we view the number of samples, n, needed to be generated before the verification algorithm terminates. We will show below how this number depends on the true failure probability p_F. Although arguably dependent on the domain and the current plan π, because this ultimately determines p_F, we can estimate the sample size needed independently of any particular domain or plan, hence motivating the label "domain independent".

Sample Size

We can expect to need fewer samples the further the true failure probability is from the indifference region. If p_F is significantly less than θ_0, we can expect to satisfy the acceptance criterion at an early stage. Conversely, if p_F is much above θ_1, the number of failures observed will tend to quickly exceed the rejection number. The number of samples, n, required by the verification algorithm is a random variable since it depends on the outcome of the observations. The expected value of n, often called the *average sample number*, depends on p_F.

Wald (1945) provides an approximation formula for the expected sample size. Let $\mathrm{E}_p[n]$ denote the expected number of samples required given $p_F = p$. An approximate value for this expectation is

$$\tilde{\mathrm{E}}_p[n] = \frac{L(p)\log\dfrac{\beta}{1-\alpha} + (1-L(p))\log\dfrac{1-\beta}{\alpha}}{p\log\dfrac{\theta_1}{\theta_0} + (1-p)\log\dfrac{1-\theta_1}{1-\theta_0}}, \qquad (4)$$

where $L(p)$ is the probability that the sequential test terminates with acceptance if $p_F = p$.

Figure 4 plots the average sample number in a region close to θ (for $\theta = 0.05$) and with different choices of δ, α, and β. We can see that by raising the risk levels or widening the indifference region, we will need fewer runs on average to reach a decision. This gives us an opportunity to trade quality for performance.

Truncated Test. As mentioned earlier, we may want to set an upper limit $n_{\max}$ on the number of samples generated. Equation (4) can help us choose this upper bound. The average sample number is at a maximum at, or close to, the common slope s of the acceptance and rejection lines:

$$s = \frac{\log\dfrac{1-\theta_0}{1-\theta_1}}{\log\dfrac{\theta_1}{\theta_0} - \log\dfrac{1-\theta_1}{1-\theta_0}}$$

The average sample number at this point is approximately

$$\tilde{\mathrm{E}}_s(n) = \frac{\log\dfrac{\beta}{1-\alpha}\log\dfrac{1-\beta}{\alpha}}{\log\dfrac{\theta_1}{\theta_0}\log\dfrac{1-\theta_1}{1-\theta_0}}.$$

With $n_{\max} = 3\tilde{\mathrm{E}}_s(n)$, the probability that the sequential test has terminated before n reaches $n_{\max}$ is nearly 1, and the truncation has a negligible effect on the strength of the test.

Related Work

BURIDAN uses a notion of plan failure similar to ours, where a threshold is given representing the maximum acceptable failure probability (Kushmerick, Hanks, & Weld 1995). BURIDAN implements several methods for plan assessment, computing a guaranteed upper bound on the failure probability. The cost of obtaining a guaranteed bound is that the efficiency of these methods vary significantly between domains. In our approach, we can trade efficiency for accuracy by adjusting the risk levels. A further difference is that our world model allows for external events, while in BURIDAN only actions can be represented, and there is only a limited notion of time, where each performed action represents a discrete time step. The same holds for planners adopting a model based on Markov decision processes.

The work by Dean & Kanazawa (1989) is more closely related to ours. They use what they call probabilistic projection to reason about persistence of propositions, but their model is Markovian. They construct a belief network in order to compute probabilistic predictions. Blythe (1994) uses a similar approach for computing the failure probability of a plan subject to external events. Probabilistic inference in belief networks is known to be NP-hard (Cooper 1990), however, and current exact algorithms have worst-case exponential behavior. Both Blythe and Dean & Kanazawa consider approximate algorithms, but they do not provide any guaranteed error bounds, and the convergence is often slow.

Atkins, Durfee, & Shin (1996) consider a probabilistic extension of CIRCA similar to ours. Their approach is analytical, and they present an iterative algorithm for state probability estimation. Their state probability calculations are based on heuristic approximations of transition times, but no quantitative error bounds are provided by the algorithm. Furthermore, they do not propagate probabilities around cycles in the state space, which can lead to serious underestimation (although this problem is addressed in later work (Li *et al.* 2000)).

Alur, Courcoubetis, & Dill (1991) describe an algorithm for verifying formulas specified in a language called TCTL, with an underlying GSMP world model. TCTL is a branching-time temporal logic for expressing real-time properties, but lacks support for expressing quantitative bounds on probabilities. Aziz *et al.* (1996) present CSL (continuous stochastic logic), which is a formalism in which quantitative probability bounds can be expressed, and they show that the problem of verifying CSL formulas is decidable. Baier, Katoen, & Hermanns (1999) describe an implementation of a model checker using an analogous formalism. The underlying model for this work is continuous-time Markov chains, however, and not GSMPs. The difference is that in the former model only exponential probability distribution functions are permitted.

Conclusions

We have presented a probabilistic extension to CIRCA. The extended world model can be viewed as a generalized semi-Markov process. We use discrete event simulation to generate sample paths in a world model, and use acceptance sampling theory to minimize the expected number of samples needed to determine if the failure probability is sufficiently low. Our work differs from other probabilistic planners in that our world model is more expressive. Yet, we are able to bound the fraction of erroneous classifications that our plan verifier makes. Using sequential acceptance sampling, we often need very few samples to reach a decision with sufficient confidence, but the user (or a higher-level deliberation scheduling module) can easily trade efficiency for accuracy by varying the parameters δ, α, and β.

For future work, we would like to combine importance sampling (Heidelberger 1995) with acceptance sampling. Doing so could improve performance of our verification algorithm when the failure threshold is close to zero, or when the indifference region is narrow. In addition, more work on how to use probabilistic information in guiding plan generation is needed. We have mentioned how to obtain point estimates of state probabilities from sample execution paths, and that these could be used to guide the planning effort towards more likely states, effectively pruning the least likely states from the search space if limited resources are given to the planner. We would like to investigate the effectiveness of such pruning techniques in the future.

Acknowledgments. This material is based upon work supported by DARPA/ATO and the Space and Naval Warfare Systems Center, San Diego, under contract no. N66001–00–C–8039. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA, the U.S. Government, or the Space and Naval Warfare Systems Center, San Diego. Not yet approved for public release.

References

Alur, R.; Courcoubetis, C.; and Dill, D. 1991. Model-checking for probabilistic real-time systems. In Albert, J. L.; Monien, B.; and Artalejo, M. R., eds., *Proceedings of the 18th International Colloquium on Automata, Languages and Programming*, volume 510 of *Lecture Notes in Computer Science*, 115–126. Madrid, Spain: Springer.

Atkins, E. M.; Durfee, E. H.; and Shin, K. G. 1996. Plan development using local probabilistic models. In Horvitz, E., and Jensen, F. V., eds., *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, 49–56. Portland, OR: Morgan Kaufmann Publishers.

Aziz, A.; Sanwal, K.; Singhal, V.; and Brayton, R. 1996. Verifying continuous time markov chains. In Alur, R., and Henzinger, T. A., eds., *Proceedings of the 8th International Conference on Computer Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, 269–276. New Brunswick, NJ: Springer.

Baier, C.; Katoen, J.-P.; and Hermanns, H. 1999. Approximate symbolic model checking of continuous-time Markov chains. In Baeten, J. C. M., and Mauw, S., eds., *Proceedings of the 10th International Conference on Concurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, 146–161. Eindhoven, the Netherlands: Springer.

Blythe, J. 1994. Planning with external events. In de Mantaras, R. L., and Poole, D., eds., *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 94–101. Seattle, WA: Morgan Kaufmann Publishers.

Chorafas, D. N. 1960. *Statistical Processes and Reliability Engineering*. Princeton, NJ: D. Van Nostrand Company.

Cooper, G. F. 1990. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence* 42(2–3):393–405.

Dean, T., and Kanazawa, K. 1989. A model for reasoning about persistence and causation. *Computational Intelligence* 5(3):142–150.

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

Heidelberger, P. 1995. Fast simulation of rare events in queueing and reliability models. *ACM Transactions on Modeling and Computer Simulation* 5(1):43–85.

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

Li, H.; Atkins, E. M.; Durfee, E. H.; and Shin, K. G. 2000. Resource allocation for a limited real-time agent using a temporal probabilistic world model. In *Working Notes of the AAAI Spring Symposium on Real-Time Autonomous Systems*.

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

Montgomery, D. C. 1991. *Introduction to Statistical Quality Control*. New York, NY: John Wiley & Sons, second edition.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1993. CIRCA: A cooperative intelligent real-time control architecture. *IEEE Transactions on Systems, Man, and Cybernetics* 23(6):1561–1574.

Musliner, D. J.; Durfee, E. H.; and Shin, K. G. 1995. World modeling for the dynamic construction of real-time control plans. *Artificial Intelligence* 74(1):83–127.

Poore, J. H.; Mills, H. D.; and Mutchler, D. 1993. Planning and certifying software system reliability. *IEEE Software* 10(1):88–99.

Schoppers, M. J. 1987. Universal plans for reactive robots in unpredictable environments. In McDermott, J., ed., *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 1039–1046. Milan, Italy: Morgan Kaufmann Publishers.

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

Wald, A. 1945. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics* 16(2):117–186.

Wald, A. 1947. *Sequential Analysis*. New York, NY: John Wiley & Sons.