Probabilistic Verification of Discrete Event Systems Using Acceptance Sampling

Håkan L. S. Younes and Reid G. Simmons

School of Computer Science, Carnegie Mellon University
Pittsburgh, PA 15213, U.S.A.

Abstract. We propose a model independent procedure for verifying properties of discrete event systems. The dynamics of such systems can be very complex, making them hard to analyze, so we resort to methods based on Monte Carlo simulation and statistical hypothesis testing. The verification is probabilistic in two senses. First, the properties, expressed as CSL formulas, can be probabilistic. Second, the result of the verification is probabilistic, and the probability of error is bounded by two parameters passed to the verification procedure. The verification of properties can be carried out in an anytime manner by starting off with loose error bounds, and gradually tightening these bounds.

1 Introduction

In this paper we consider the problem of verifying properties of discrete event systems. We present a procedure for verifying probabilistic real-time properties of such systems based on Monte Carlo simulation and statistical hypothesis testing. The verification procedure is not tied to any specific model of discrete event systems—we only require that *sample execution paths* for the systems can be generated—but it is mainly intended for verification of systems with complex dynamics such as generalized semi-Markov processes (GSMPs) [14, 8] for which no symbolic methods exist, or semi-Markov processes (SMPs) [11] for which current symbolic and numeric methods do not yield a practical solution.

Since we are using sampling, we cannot guarantee that our verification procedure always produces the correct answer. A key result, however, is that we can bound the probability of error with two parameters α and β, where α is the largest acceptable probability of incorrectly verifying a true property, and β is the largest acceptable probability of incorrectly verifying a false property.

The number of sample execution paths required to verify certain properties can be large, but our procedure can be used in an anytime manner by first verifying a property with loose error bounds, and then successively tighten the error bounds to obtain more accurate results.

We adopt the continuous stochastic logic (CSL) as our formalism for expressing probabilistic real-time properties of discrete event systems. CSL has previously been proposed as a formalism for expressing temporal and probabilistic properties of continuous-time Markov chains (CTMCs) [3, 4, 5] and SMPs [12].

D. Brinksma and K. G. Larsen (Eds.): CAV 2002, LNCS 2404, pp. 223–235, 2002.

The problem of verifying properties of GSMPs has been considered before, but in a qualitative setting where it is checked whether a property holds with probability one or greater than zero [2]. With our approach, we are able to verify whether a property holds with at least (or at most) probability θ, for an arbitrary probability threshold θ. Kwiatkowska et al. [13] present an algorithm for verifying probabilistic timed automata against properties expressed in probabilistic timed CTL, but the complexity of their algorithm makes it seem practically infeasible. Infante Lopéz et al. [12] propose a method for verifying CSL properties of SMPs. They conclude, however, that verifying time-bounded CSL formulas using their algorithm can become numerically very complex, and the negative complexity results carry over to GSMPs.

2 Discrete Event Systems

The verification procedure we present in this paper is model independent, and only requires that we can generate sample execution paths for a discrete event system we want to verify. Because of the model independence, we choose not to introduce any specific model for discrete event systems, but instead focus only on relevant properties of such systems. We will typically use discrete event simulation [15] to generate sample execution paths, but our verification procedure could conceivably be used to verify probabilistic real-time properties of hybrid dynamic systems as well given that we have an appropriate simulator.

At any point in time, a discrete event system occupies some state $s \in S$, where S is a set of states.[1] Let AP be a fixed, finite set of atomic propositions. We then define a labeling function $L : S \to 2^{AP}$ assigning to each state $s \in S$ the set $L(s)$ of atomic propositions that hold in s. The system remains in a state s until the occurrence of an event, at which point the system instantaneously transitions to a state s' (possibly the same state as s). Events can occur at any point along a continuous time-axis.

Execution Paths. An execution path σ of a discrete event system is a sequence

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

with $s_i \in S$ and $t_i > 0$ being the time spent in state s_i before an event triggered a transition to state s_{i+1}. If the lth state of path σ is absorbing, then we set $s_i = s_l$ for all $i > l$, and $t_i = \infty$ for all $i \geq l$.

Let $\sigma[i] = s_i$, for $i \geq 0$, be the ith state along the path σ, let $\delta(\sigma, i) = t_i$ be the time spent in state s_i, let $\tau(\sigma, i) = \sum_{j=0}^{i-1} \delta(\sigma, j)$ be the time elapsed before entering the ith state, and let $\sigma(t) = \sigma[i]$ with i being the smallest index such that $t \leq \tau(\sigma, i+1)$. We denote the set of all paths starting in state s by $Path(s)$. For any given model, we need to define a σ-algebra on the set $Path(s)$ and a

[1] We do not require S to be finite. In fact, for GSMPs it is convenient to think of a state s as some discrete state features s' coupled with a set of real-valued clock settings c for currently enabled events (see [8, 2]).

probability measure on the corresponding measurable space, or else we will not be able to talk about the probability of a set of paths satisfying a property. This is not a serious restriction, however, because it can be done for the models we typically use for discrete event systems. It is done in [5] for CTMCs and in [12] for SMPs, and can be done in a similar way for GSMPs (cf. [13]).

3 Continuous Stochastic Logic

Aziz et al. [3] propose the continuous stochastic logic (CSL) as a formalism for expressing properties of CTMCs. CSL—inspired by CTL [7] and its extensions to continuous-time systems [1, 2]—adopts temporal operators and probabilistic path quantification from PCTL [10].

We adopt the version of CSL used by Baier et al. [5], excluding their steady-state probability operator and unrestricted temporal operators[2], and present a semantics for CSL formulas interpreted over discrete event systems. The semantics is model dependent only through the definition of execution paths.

CSL Syntax. A CSL formula is either a state formula or a path formula. The formulas of CSL are inductively defined as follows:

1. tt is a state formula.
2. $a \in AP$ is a state formula.
3. If ϕ is a state formula, then so is $\neg\phi$.
4. If ϕ_1 and ϕ_2 are state formulas, then $\phi_1 \wedge \phi_2$ is a state formula.
5. If ρ is a path formula and $\theta \in [0,1]$, then $\Pr_{\geq\theta}(\rho)$ is a state formula.[3]
6. If ϕ is a state formula, then $X\phi$ (next state) is a path formula.
7. If ϕ_1 and ϕ_2 are state formulas and $0 \leq t < \infty$, then $\phi_1 \,\mathcal{U}^{\leq t}\, \phi_2$ (until) is a path formula.

Other Boolean connectives and path operators are derived in the usual way. For example, $\Pr_{\geq\theta}(\Diamond^{\leq t}\,\phi)$ can be written as $\Pr_{\geq\theta}(\mathrm{tt}\,\mathcal{U}^{\leq t}\,\phi)$.

CSL Semantics. The truth value of a state formula is determined in a specific state. The formula $\Pr_{\geq\theta}(\rho)$ holds in a state s iff the probability of the set of paths starting in s and satisfying ρ is at least θ.

The truth value of a path formula is determined over a specific execution path. The semantics of the next and until operators is standard. The formula $X\phi$ is true over a path σ iff ϕ holds in the state after the first transition. If the initial state along σ is absorbing, there is no next state so the formula is false. The formula $\phi_1 \,\mathcal{U}^{\leq t}\, \phi_2$ is true over a path σ iff ϕ_2 holds in some state along σ at time $x \in [0,t]$, and ϕ_1 holds in all prior states along σ.

We inductively define the satisfaction relation $\models$ as follows:

[2] We need the time-bound on the temporal operators to set a limit on the simulation time for the generation of sample execution paths.

[3] With the sampling based verification procedure we propose, it is not meaningful to distinguish between $\Pr_{\geq\theta}(\rho)$ and $\Pr_{>\theta}(\rho)$. We can therefore write $\Pr_{\leq\theta}(\rho)$ as $\neg\Pr_{\geq\theta}(\rho)$, which means we only need to consider one comparison operator.

1. $s \models \mathrm{tt}$ for all $s \in S$.
2. $s \models a$ iff $a \in L(s)$.
3. $s \models \neg\phi$ iff $s \not\models \phi$.
4. $s \models \phi_1 \wedge \phi_2$ iff $s \models \phi_1$ and $s \models \phi_2$.
5. $s \models \Pr_{\geq\theta}(\rho)$ iff $\Pr\{\sigma \in \mathit{Path}(s) \mid \sigma \models \rho\} \geq \theta$.
6. $\sigma \models X\phi$ iff $\delta(\sigma, 0) < \infty$ and $\sigma[1] \models \phi$.
7. $\sigma \models \phi_1 \, \mathcal{U}^{\leq t} \, \phi_2$ iff $\sigma(x) \models \phi_2$ for some $x \in [0, t]$ and $\sigma(y) \models \phi_1$ for all $y \in [0, x)$.

The probability measure $\Pr\{\ldots\}$ must be well defined, as described in the previous section.

4 Probabilistic Verification

Given a discrete event system M and a state s of M, we want to verify that a property—expressed as a state formula ϕ in CSL—holds in s. In other words, we desire to test if $s \models \phi$.

The complexity of general discrete event systems makes them hard to analyze, and we resort to methods based on Monte Carlo simulation and statistical hypothesis testing. This means that in general we will not be able to answer with certainty whether a given property holds, but we will at least be able to bound the likelihood of error.

More specifically, given s and ϕ, let H_0 be the hypothesis that ϕ holds in s, and let H_1 be the alternative hypothesis (i.e. that ϕ does not hold in s). The probability of accepting H_1 given that H_0 holds is required to be at most α, and the probability of accepting H_0 if H_1 holds should be no more than β. The error bounds α and β are supplied as parameters to the verification procedure, which is devised so that less effort (on average) is required to verify a property with more relaxed error bounds.

4.1 Verifying Probabilistic Properties

The possibility of error in our verification procedure arises from the way we verify probabilistic properties $\phi = \Pr_{\geq\theta}(\rho)$ given a state s. Let p be the (unknown) probability that ρ holds over paths starting in s. If $p \geq \theta$, then ϕ holds in s.

We use simulation (typically discrete event simulation) to generate sample paths starting in s. Let Y be a binary random variable with parameter p such that $\Pr[Y = 1] = p$. Sample paths over which ρ holds represent samples $y_i = 1$ of Y, and remaining sample paths represent samples $y_i = 0$ of Y. Using these samples, we would like to test the hypothesis $p \geq \theta$ against the alternative hypothesis $p < \theta$, but we are forced to relax the hypotheses in order to freely be able to choose error bounds α and β.

For this purpose we introduce an indifference region of width $2{\cdot}\delta$. Let $p \geq \theta{+}\delta$ be H_0 and let $p \leq \theta{-}\delta$ be H_1. We use acceptance sampling to test H_0 against H_1. The outcome of the acceptance sampling test is that we accept either H_0 or H_1,

so the two events "accept H_0" and "accept H_1" are mutually exclusive and exhaustive. Note, however, that for a non-zero δ the two hypotheses H_0 and H_1 are not exhaustive although they are mutually exclusive. Let H_2 be the hypothesis that neither H_0 nor H_1 holds. H_2 represents indifference, and holds if the probability of ρ being true over paths starting in s is within δ of θ.

We are given the following guarantees by an acceptance sampling test:

$$\Pr[H_0 \text{ holds} \mid \text{accept } H_1] \leq \alpha$$
$$\Pr[H_1 \text{ holds} \vee H_2 \text{ holds} \mid \text{accept } H_1] \geq 1 - \alpha$$
$$\Pr[H_1 \text{ holds} \mid \text{accept } H_0] \leq \beta$$
$$\Pr[H_0 \text{ holds} \vee H_2 \text{ holds} \mid \text{accept } H_0] \geq 1 - \beta$$

The formula ϕ is definitely true if H_0 holds, and definitely false if H_1 holds, but if H_2 holds we have no information about the truth value of ϕ. Recall, however, that H_2 represents indifference—i.e. the true probability of ρ holding over paths starting in s is sufficiently close to θ that we are indifferent to whether it actually is below or above θ. In case H_2 holds we interpret this to mean that ϕ is true if we accepted H_0, and false if we accepted H_1. With this interpretation, we obtain the desired error bounds $\Pr[\phi \mid \text{accept } H_1] \leq \alpha$ and $\Pr[\neg\phi \mid \text{accept } H_0] \leq \beta$.

Nested Probabilistic Operators. The above results for formulas $\phi = \Pr_{\geq\theta}(\rho)$ hold if we can determine the truth value of ρ over sample paths without error. In case ρ contains probabilistic operators, there is some probability at most α' of ρ being true over a sample path σ if it is verified to be false, and some probability at most β' of ρ being false over σ if it is verified to be true. We need to take the possibility of error into account in the acceptance sampling test, and we here present a modification of Wald's sequential probability ratio test [16] that deals with this situation. We choose the sequential probability ratio test because of its strong average performance measured in the number of samples required to reach a decision.

We can model the situation of imprecise samples in general terms as follows. Let Y be a binary random variable with unknown parameter p such that $\Pr[Y = 1] = p$. Our goal is to test the hypothesis H_0 that $p \geq p_0$ (for $p_0 = \theta + \delta$) against the hypothesis H_1 that $p \leq p_1$ (for $p_1 = \theta - \delta$). We want the probability of accepting H_1 given that H_0 holds to be at most α, and the probability of accepting H_0 given that H_1 holds to be at most β. If we could generate samples of Y, then we could accomplish our goal using the unmodified sequential probability ratio test, but instead we can only generate samples from a binary random variable Z related to Y in the following way:

$$\Pr[Z = 1 \mid Y = 1] \geq 1 - \alpha'$$
$$\Pr[Z = 0 \mid Y = 1] \leq \alpha'$$
$$\Pr[Z = 1 \mid Y = 0] \leq \beta'$$
$$\Pr[Z = 0 \mid Y = 0] \geq 1 - \beta'$$

Given these constraints and the total probability formula, we can obtain bounds on the unconditional probability $\Pr[Z = 1]$:

$$p(1 - \alpha') \leq \Pr[Z = 1] \leq 1 - (1 - p)(1 - \beta') \tag{1}$$

We now want to use the sequential probability ratio test to test hypothesis H_0 against H_1 given samples of Z.

The sequential probability ratio test is carried out as follows. At each stage of the test, calculate the quantity

$$\frac{p_{1m}}{p_{0m}} = \frac{\prod_{i=1}^{m} \Pr[Z = z_i \mid p = p_1]}{\prod_{i=1}^{m} \Pr[Z = z_i \mid p = p_0]} \ ,$$

where z_i is the sample of Z generated at stage i. Accept H_0 if

$$\frac{p_{1m}}{p_{0m}} \geq \frac{1 - \beta}{\alpha} \ . \tag{2}$$

Accept H_1 if

$$\frac{p_{1m}}{p_{0m}} \leq \frac{\beta}{1 - \alpha} \ . \tag{3}$$

Otherwise, generate an additional sample and repeat the termination test. This test procedure respects the error bounds α and β.[4]

We cannot compute the fraction p_{1m}/p_{0m} because $\Pr[Z = 1]$ is unknown to us, but we can obtain upper and lower bounds for the fraction, which can then be used to devise a modified test respecting the error bounds α and β.

Let d_m denote the number of samples, of the first m samples, equal to 1. We can then write the fraction p_{1m}/p_{0m} as

$$\frac{p_{1m}}{p_{0m}} = \frac{(\Pr[Z = 1 \mid p = p_1])^{d_m} \, (1 - \Pr[Z = 1 \mid p = p_1])^{m - d_m}}{(\Pr[Z = 1 \mid p = p_0])^{d_m} \, (1 - \Pr[Z = 1 \mid p = p_0])^{m - d_m}} \ .$$

Let I_i be the interval $[p_i(1 - \alpha'), 1 - (1 - p_i)(1 - \beta')]$. We know from (1) that $\Pr[Z = 1 \mid p = p_i] \in I_i$. A lower bound for p_{1m}/p_{0m} can be obtained by finding a $\check{p}_1 \in I_1$ that minimizes p_{1m} and a $\hat{p}_0 \in I_0$ that maximizes p_{0m}. Conversely, an upper bound for the fraction can be obtained by finding a $\hat{p}_1 \in I_1$ that maximizes p_{1m} and a $\check{p}_0 \in I_0$ that minimizes p_{0m}. We then have the bounds

$$\frac{(\check{p}_1)^{d_m} \, (1 - \check{p}_1)^{m - d_m}}{(\hat{p}_0)^{d_m} \, (1 - \hat{p}_0)^{m - d_m}} \leq \frac{p_{1m}}{p_{0m}} \leq \frac{(\hat{p}_1)^{d_m} \, (1 - \hat{p}_1)^{m - d_m}}{(\check{p}_0)^{d_m} \, (1 - \check{p}_0)^{m - d_m}}$$

for the fraction p_{1m}/p_{0m}. Given these bounds, it is safe to accept H_0 if

$$\frac{(\check{p}_1)^{d_m} \, (1 - \check{p}_1)^{m - d_m}}{(\hat{p}_0)^{d_m} \, (1 - \hat{p}_0)^{m - d_m}} \geq \frac{1 - \beta}{\alpha} \tag{4}$$

[4] There is a slight approximation involved in the stopping criteria of the test. See [16] for details.

since then surely condition (2) holds. Likewise, it is safe to accept H_1 if

$$\frac{(\hat{p}_1)^{d_m}\,(1-\hat{p}_1)^{m-d_m}}{(\check{p}_0)^{d_m}\,(1-\check{p}_0)^{m-d_m}} \leq \frac{\beta}{1-\alpha} \tag{5}$$

since then surely condition (3) holds. By replacing the original stopping criteria (2) and (3) with the new stopping criteria (4) and (5) we obtain a sequential acceptance sampling test that can handle imprecise samples. We now need to find the appropriate values for $\check{p}_i$ and $\hat{p}_i$.

Proposition 1. *Let* $f(x) = x^{d_m}(1-x)^{m-d_m}$. *For* $d_m \in I_i$, $\hat{p}_i = d_m/m$ *and*

$$\check{p}_i = \begin{cases} p_i(1-\alpha') & \text{if } f(p_i(1-\alpha')) < f(1-(1-p_i)(1-\beta')) \\ 1-(1-p_i)(1-\beta') & \text{otherwise} \end{cases}.$$

If $d_m/m < p_i(1-\alpha')$, *then* $\hat{p}_i = p_i(1-\alpha')$ *and* $\check{p}_i = 1-(1-p_i)(1-\beta')$. *Otherwise if* $d_m/m > 1-(1-p_i)(1-\beta')$, *then* $\hat{p}_i = 1-(1-p_i)(1-\beta')$ *and* $\check{p}_i = p_i(1-\alpha')$.

Proof. For $d_m = 0$, $f(x) = (1-x)^m$ is monotonously decreasing in the interval $[0,1]$. For $d_m = m$, $f(x) = x^m$ is monotonously increasing in the interval $[0,1]$. Otherwise for $0 < d_m < m$, $f(0) = f(1) = 0$. The first derivative of $f(x)$ is $f'(x) = d_m x^{d_m-1}(1-x)^{m-d_m} - (m-d_m)x_m^d(1-x)^{m-d_m-1}$. We can find a local maximum of $f(x)$ in the open interval $(0,1)$ by setting $f'(x) = 0$:

$$d_m x^{d_m-1}(1-x)^{m-d_m} = (m-d_m)x_m^d(1-x)^{m-d_m-1} \implies$$
$$d_m(1-x) = (m-d_m)x \implies x = \frac{d_m}{m}$$

Thus, $f(x)$ has a local maximum at d_m/m. □

The stopping criteria (4) and (5) reduce to the regular stopping criteria for the test if $\alpha' = \beta' = 0$, as expected. With imprecise samples, the average number of samples required before a decision can be reached will increase, but it is worth noting that the choice of α' and β' can be made independent of the values for α and β.

4.2 Verifying Compound State Formulas

When verifying a compound state formula ϕ such as $\neg\phi_1$ or $\phi_1 \wedge \phi_2$ in a state s, we first test parts that do not involve any probabilistic operators. The truth value of those parts can be determined with certainty, and results in a reduced formula ϕ'. If ϕ reduces to either true or false we are done. Otherwise all parts of ϕ' contain probabilistic operators, and we need to propagate appropriate error bounds to the test of the parts in order to obtain the desired error bounds for the compound formula.

Negation. For a negation $\neg\phi_1$, assume inductively that we can obtain the error bounds α_1 and β_1 for ϕ_1. By setting $\beta_1 = \alpha$ and $\alpha_1 = \beta$, we obtain the required error bounds for $\neg\phi_1$.

Conjunction. For a conjunction $\phi_1 \wedge \cdots \wedge \phi_n$, the situation is slightly more complicated. We want to accept the conjunction as true if all conjuncts are true, and reject the conjunct as false if some conjunct is false.

First assume that we can verify each conjunct ϕ_i as true with error bounds α_i and β_i. This means that the probability of ϕ_i being false is at most β_i, which implies that the probability of the whole conjunction being false is at most $\sum_{i=1}^{n} \beta_i$. We can thus achieve a verification error of at most β in this case if we choose the β_i's so that $\sum_{i=1}^{n} \beta_i = \beta$. Without any further information about the complexity of each ϕ_i, the natural choice is $\beta_i = \beta/n$.

Now assume that we can verify some conjuncts ϕ_i as false with error bounds α_i and β_i. This means that the probability of ϕ_i being true is at most α_i, which implies that the probability of the whole conjunction being true is at most $\max_{i=1}^{n} \alpha_i$. By setting $\alpha_i = \alpha$, we achieve the desired error bound α on the verification of the whole conjunction.

We combine these two results into a complete verification procedure for conjunctions. In order to minimize the expected verification effort, we use a two-step procedure. The first step is a "fast reject" step, in which we verify each conjunct ϕ_i with error bounds α and β', where β' can be chosen arbitrarily. If we can verify any ϕ_i as false using these bounds, we can conclude with sufficient confidence that the whole conjunction is false. We will want to choose β' high so that the number of samples required to verify each ϕ_i in the first step is low, but not too high because it would lower the chance of verifying any ϕ_i as false.

If we verify each conjunct as true in the first step, we perform a second step corresponding to a "rigorous accept". Again we verify each conjunct ϕ_i with $\alpha_i = \alpha$, but this time with $\beta_i = \beta/n$. If we verify any conjunct ϕ_i as false using these bounds, we can conclude with sufficient confidence that the whole conjunction is false, but we can also conclude with sufficient confidence that the conjunction is true if we verify each conjunct as true.

4.3 Verifying Path Formulas

When verifying $\Pr_{\geq\theta}(\rho)$ in a state s, we need to determine the truth value of the path formula ρ over sample execution paths starting in the given state. A sample path σ is generated by simulation. We only generate as much of a path that is needed to determine the truth value of ρ with sufficient confidence.

The Next Operator. To verify a path formula $\rho = X\phi$ with error bounds α and β over a path starting in the state s, we sample a next state s'. We then verify ϕ in s' with α and β as error bounds. If s is a terminal state, we can conclude without error that ρ is false.

The Until Operator. A path formula $\rho = \phi_1 \, \mathcal{U}^{\leq t} \, \phi_2$ holds over a path σ if ϕ_2 holds in $\sigma[0]$, or if ϕ_2 holds in $\sigma[i]$ and ϕ_1 holds in all states $\sigma[j]$ for $j < i$. Let $\phi_i^{(j)}$ represent the proposition that ϕ_i holds in the state $\sigma[j]$, and let n be the

smallest index such that $\tau(\sigma, n) > t$. Then the path formula ρ holds over σ iff

$$\bigvee_{i=0}^{n-1} \left(\phi_2^{(i)} \wedge \bigwedge_{j=0}^{i-1} \phi_1^{(j)} \right) \quad , \tag{6}$$

which can be verified in the same way as a compound state formula.

Equation (6) is a disjunction of size n, with the ith disjunct being a conjunction of size $i + 1$. In the worst case, we will have to verify each disjunct with error bounds α/n and β in order to verify the whole formula with error bounds α and β. In that case we may have to verify each component of the ith disjunct with error bounds α/n and $\beta/(i+1)$, which can require quite a few samples if n is large. This will happen if both ϕ_1 and ϕ_2 contain probabilistic operators and we verify the path formula ρ as true.

The problem of verifying an until formula simplifies significantly if either ϕ_1 or ϕ_2, or both, can be verified without error. In the simples case, without any nested probabilistic operators, we only need to expand the path σ until either ϕ_2 becomes true, or ϕ_1 becomes false or the time limit is exceeded. In the former case the until formula holds with certainty, while in the latter case we can conclude with certainty that the until formula is false.

5 Performance Evaluation

The performance of our procedure for verifying a formula on the form $\phi = \Pr_{\geq\theta}(\rho)$ in a state s depends primarily on the number of samples n needed by the acceptance sampling test used. If we are using a sequential test, such as Wald's sequential probability ratio test [16], then n is a random variable. Let $\mathrm{E}_p[n]$ denote the expected number of samples required by the test given that p is the true probability of ρ holding over paths starting in s. We can expect to need more samples the closer p is to the probability threshold θ.

The expected number of samples depends not only on p and θ, but also on the parameters δ, α, and β. In addition, if there are nested probabilistic operators so that we need to use the modified test as described earlier in this paper, then the average number of samples also depends on the parameters α' and β' corresponding to the maximum error in the verification of a nested formula.

Figure 1 shows the average number of samples as a function of p for three different values of θ, and with the remaining parameters fixed. The data is based on 5,000 tests for each of 201 equidistant values of p. Similar data is shown in Fig. 2 but with θ fixed and δ varying, and in Fig. 3 the error bounds α and β are varying. Finally, Fig. 4 shows how the average number of samples increases with an increase in α' and β'. The dotted curve is the same in all four figures.

As can be seen, the number of samples is typically very low, suggesting that the proposed verification procedure can be quite efficient. Note however that if we are verifying a formula with nested probabilistic operators, then for each sample generated for the outer probabilistic operator, we need to generate several samples to verify the inner probabilistic operator. Given one level of nesting, if

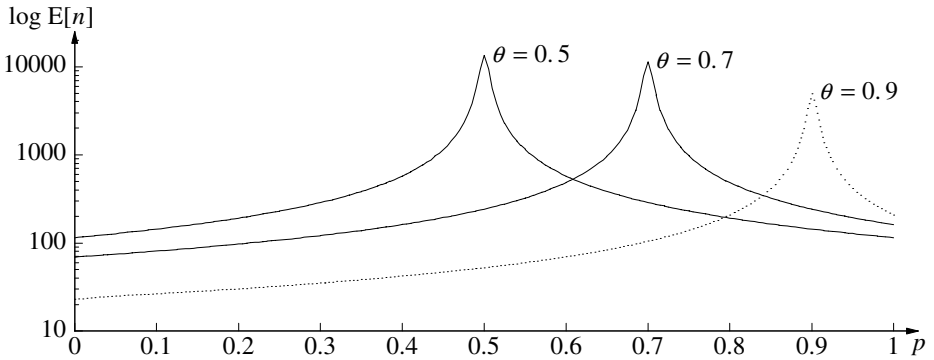

Fig. 1. Expected number of samples for different values of θ, with $\delta = 0.01$, $\alpha = \beta = 0.01$, and $\alpha' = \beta' = 0$

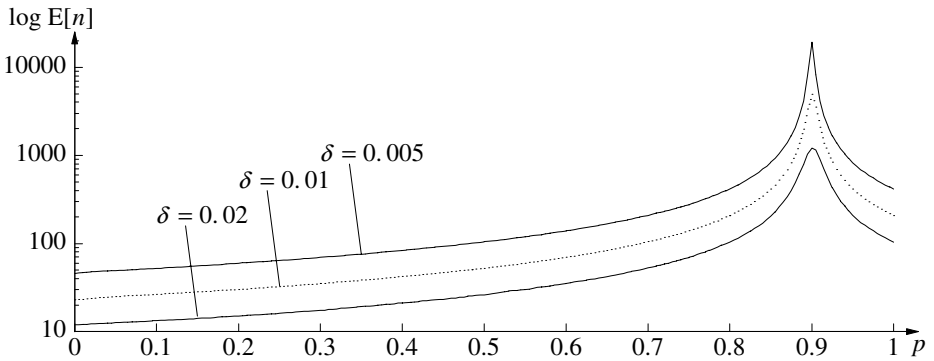

Fig. 2. Expected number of samples for different values of δ, with $\theta = 0.9$, $\alpha = \beta = 0.01$, and $\alpha' = \beta' = 0$

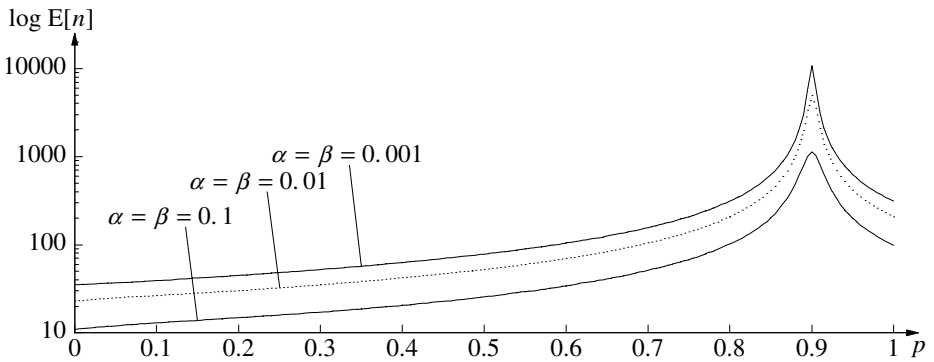

Fig. 3. Expected number of samples for different values of α and β, with $\theta = 0.9$, $\delta = 0.01$, and $\alpha' = \beta' = 0$

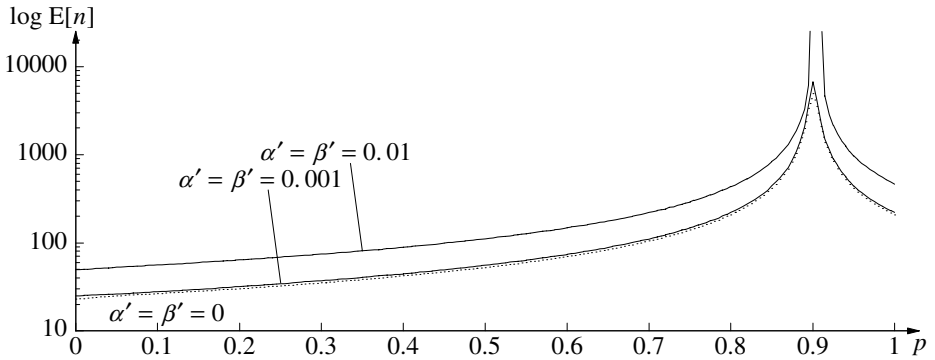

Fig. 4. Expected number of samples for different values of α' and β', with $\theta = 0.9$, $\delta = 0.01$, and $\alpha = \beta = 0.01$

the expected number of samples for the outer probabilistic operator is n_o, and we need n_i samples on average for the inner operator, then $n_o \cdot n_i$ is the expected number of samples needed to verify the whole formula. The total number of required samples grows rapidly with the level of nesting, but this does not seem to be a problem in practice since CSL formulas typically have at most one level of nesting—if any at all.

There is no definite upper bound on the number of samples required by the sequential probability ratio test. If this is a problem, as it can be when verifying nested probabilistic formulas, then a truncated test can be used. Wald [16] suggests a method for choosing an upper bound on the number of samples so that the given error bounds, for all practical purposes, are still respected.

6 Discussion

We have presented a model independent procedure for verifying properties of discrete event systems. The properties are expressed as CSL formulas, and we have shown how to interpret these formulas given a definition of sample execution paths of a discrete event system. The definition of sample execution paths, as well as the probability measure on sets of paths, is the only model dependent component of the framework we have discussed.

Because of the complex nature of many discrete event systems, we depend on Monte Carlo simulation and statistical hypothesis testing in order to verify CSL formulas. Our verification procedure takes two parameters, α and β, where α is the highest acceptable probability of incorrectly verifying a true formula, and β is the highest acceptable probability of incorrectly verifying a false formula.

Using sequential acceptance sampling the number of samples required for verifying a CSL formula is typically low, but can be high for verifying certain formulas—in particular formulas of the form $\Pr_{\geq\theta}(\phi_1 \, \mathcal{U}^{\leq t} \, \phi_2)$, where both ϕ_1 and ϕ_2 contain probabilistic operators. Our verification procedure can, however,

be applied in an anytime manner. To do this, we would start by verifying a formula ϕ with loose error bounds α and β, which should produce a result quickly. We could then successively tighten the error bounds, and obtain more accurate results the more resources we give the verifier.

A direction for future research would be to obtain a better understanding of the number of samples required for verifying properties of varying complexity, and how to best choose parameter values when there is a free choice (e.g. α' and β' in case of nested probabilistic formulas). It may also be possible to increase performance when verifying conjunctions (and therefore also until formulas with nested probabilistic operators) by considering heuristics for ordering conjuncts (cf. variable ordering heuristics for constraint satisfaction problems [6]).

Another problem to consider is that of verifying CSL formulas with unrestricted temporal operators and the steady-state operator, which requires developing techniques for evaluating the long-run behavior of a discrete event system. Work on output analysis for simulation of transient and steady-state quantities in operations research (see, e.g., [9]) may be applicable. Also, the algorithm proposed by Infante Lopéz et al. [12] is reported to scale well to SMPs in the case of unrestricted temporal operators and the steady-state operator, and a similar approach may be fruitful even for more general models of discrete event systems.

References

[1] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking for real-time systems. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 414–425, Philadelphia, PA, June 1990. IEEE Computer Society Press. 225

[2] Rajeev Alur, Costas Courcoubetis, and David Dill. Model-checking for probabilistic real-time systems. In J. Leach Albert, B. Monien, and M. Rodríguez Artalejo, editors, *Proceedings of the 18th International Colloquium on Automata, Languages and Programming*, volume 510 of *Lecture Notes in Computer Science*, pages 115–126, Madrid, Spain, July 1991. Springer. 224, 225

[3] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert Brayton. Verifying continuous time Markov chains. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th International Conference on Computer Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 269–276, New Brunswick, NJ, July/August 1996. Springer. 223, 225

[4] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert Brayton. Model-checking continuous-time Markov chains. *ACM Transactions on Computational Logic*, 1(1):162–170, July 2000. 223

[5] Christel Baier, Joost-Pieter Katoen, and Holger Hermanns. Approximate symbolic model checking of continuous-time Markov chains. In Jos C. M. Baeten and Sjouke Mauw, editors, *Proceedings of the 10th International Conference on Concurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 146–161, Eindhoven, the Netherlands, August 1999. Springer. 223, 225

[6] James R. Bitner and Edward M. Reingold. Backtrack programming techniques. *Communications of the ACM*, 18(11):651–656, November 1975. 234

[7] E. M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, April 1986. 225

[8] Peter W. Glynn. A GSMP formalism for discrete event systems. *Proceedings of the IEEE*, 77(1):14–23, January 1989. 223, 224

[9] Peter W. Glynn and Donald L. Iglehart. Simulation methods for queues: An overview. *Queueing Systems*, 3:221–255, 1988. 234

[10] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994. 225

[11] Ronald A. Howard. *Dynamic Probabilistic Systems*, volume II. John Wiley & Sons, New York, NY, 1971. 223

[12] Gabriel G. Infante López, Holger Hermanns, and Joost-Pieter Katoen. Beyond memoryless distributions: Model checking semi-Markov chains. In Luca de Alfaro and Stephen Gilmore, editors, *Proceedings of the 1st Joint International PAPM-PROBMIV Workshop*, volume 2165 of *Lecture Notes in Computer Science*, pages 57–70, Aachen, Germany, September 2001. Springer. 223, 224, 225, 234

[13] Marta Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Verifying quantitative properties of continuous probabilistic timed automata. In Catuscia Palamidessi, editor, *Proceedings of the 11th International Conference on Concurrency Theory*, volume 1877 of *Lecture Notes in Computer Science*, pages 123–137, State College, PA, August 2000. Springer. 224, 225

[14] Klaus Matthes. Zur Theorie der Bedienungsprozesse. In Jaroslav Kožešník, editor, *Transactions of the Third Prague Conference on Information Theory, Statistical Decision Functions, Random Processes*, pages 513–528, Liblice, Czechoslovakia, June 1962. Publishing House of the Czechoslovak Academy of Sciences. 223

[15] Gerald S. Shedler. *Regenerative Stochastic Simulation*. Academic Press, Boston, MA, 1993. 224

[16] Abraham Wald. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, June 1945. 227, 228, 231, 233