Statistical Probabilistic Model Checking with a Focus on Time-Bounded Properties [1]

Håkan L. S. Younes [*], Reid G. Simmons

School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 15213, USA

Abstract

Probabilistic verification of continuous-time stochastic processes has received increasing attention in the model-checking community in the past five years, with a clear focus on developing numerical solution methods for model checking of continuous-time Markov chains. Numerical techniques tend to scale poorly with an increase in the size of the model (the "state space explosion problem"), however, and are feasible only for restricted classes of stochastic discrete-event systems. We present a *statistical* approach to probabilistic model checking, employing hypothesis testing and discrete-event simulation. Since we rely on statistical hypothesis testing, we cannot guarantee that the verification result is correct, but we can at least bound the probability of generating an incorrect answer to a verification problem.

Key words: Probabilistic verification, Stochastic processes, Temporal logic, Hypothesis testing, Acceptance sampling, Transient analysis

1 Introduction

Stochastic processes are used to model phenomena in nature that involve an element of chance, such as radioactive decay, or are too complex to fully capture in a deterministic fashion, such as the duration of a call in a telephone system. Given a stochastic process, it is often of interest to know if certain

[*] Corresponding author.
Email addresses: `lorens@cs.cmu.edu` (Håkan L. S. Younes), `reids@cs.cmu.edu` (Reid G. Simmons).

[1] Supported in part by the US Army Research Office (ARO) under contract no. DAAD190110485 and the Royal Swedish Academy of Engineering Sciences (IVA) with grants from the Hans Werthén Fund.

properties hold. For instance, we may ask whether the probability of exhausting bandwidth over a communication link is below 0.01. We can also introduce deadlines, for example that a message arrives at its destination within 15 seconds with probability at least 0.8. Numerous temporal logics (e.g. TCTL [1], PCTL [28], and CSL [6,9]) exist for expressing such properties.

Model checking, pioneered by Clarke and Emerson [15], is a technique for automated verification of hardware and software systems. Practical model-checking algorithms have been developed for verifying probabilistic properties of stochastic systems (e.g. discrete-time Markov chains [28], continuous-time Markov chains [9], and semi-Markov processes [40]). These approaches employ numerical solution techniques to compute probability estimates. The numerical estimates are then compared to the probability thresholds of the properties that are being verified to determine whether the properties hold.

This paper presents a *statistical* approach to model checking based on hypothesis testing and simulation. A key observation is that it is not necessary to obtain an accurate estimate of a probability in order to verify probabilistic properties. For example, to verify whether the probability of exhausting bandwidth is below 0.01, we do not need to know the exact probability that bandwidth will be exhausted. It would be a waste of effort to obtain an accurate estimate of this probability only to realize that it is far from the specified threshold of 0.01. Instead, we can think of a model-checking problem in terms of hypothesis testing. Let p be the actual probability that bandwidth is exhausted. To verify the given property, we need to test the hypothesis $H : p < 0.01$ against the alternative hypothesis $K : p \geq 0.01$. Such hypothesis-testing problems can be solved efficiently using an established statistical technique called *acceptance sampling*, described in further detail in Section 2.

Because we rely on statistical sampling techniques, our solution method is not tied to a specific class of stochastic processes. We can, in principle, handle any system for which we can generate sample execution trajectories. In fact, we could even use trajectories generated from the execution of an actual system rather than through simulation of a model. We focus on the general class of *stochastic discrete-event systems*, described in Section 3, which includes any stochastic process with piecewise constant trajectories and *without nondeterminism*. Our solution method is limited to non-explosive (time-divergent) processes for which only a finite number of events can occur in a finite amount of time. Most finite-state models satisfy this requirement by default.

Existing logics for expressing probabilistic real-time properties of stochastic systems without nondeterminism have semantics that are tied to restricted classes of systems, for instance discrete-time Markov chains in the case of PCTL. To enable a uniform treatment for model checking of all stochastic discrete-event systems, we present the *Unified Temporal Stochastic Logic*

(UTSL) in Section 4. UTSL coincides with PCTL for discrete-time Markov chains and Baier et al.'s [9] CSL (without the steady-state operator) for continuous-time Markov chains. We introduce a version of UTSL, called UTSL$_\delta$, with a relaxed semantics. It is for this version that we develop a model-checking algorithm. The relaxation introduces *indifference regions* around probability thresholds. The rationale behind this relaxation is that when we ask if a probability is above some threshold θ, then we are indifferent with respect to the correctness of the answer if the probability is sufficiently close to θ (less than a δ distance away from θ). We make δ a parameter of our model-checking algorithm. As discussed in Section 8, numerical approaches should also be seen as model-checking algorithms for UTSL$_\delta$ rather than UTSL due to roundoff and truncation errors.

Section 5 introduces a model-checking algorithm for UTSL$_\delta$, based on statistical hypothesis testing. This work originated in an effort to verify plans for complex stochastic temporal domains, with a focus on probabilistic time-bounded reachability properties [70]. Time-bounded CSL properties were later considered [71], although with an unsatisfactory solution for nested probabilistic operators. Younes [66] improved the solution method for nested probabilistic operators, but introduced an error in the analysis of conjunctions, which also has consequences for the verification of path formulae with probabilistic operators. These shortcomings have now been addressed, and a sound *and* practical solution to the verification of properties with nested probabilistic operators is presented for the first time in this paper. The main theoretical results are Theorems 9 and 12, showing how to bound the probability of error for conjunctive and nested probabilistic statements, respectively. With nesting, our solution method is restricted to Markov chains.

Section 6 presents four case studies that are used in Section 7 to evaluate our statistical model-checking algorithm. We show how the performance of the algorithm depends, in practice, on user-supplied parameters and model characteristics. This supplements the theoretical complexity analysis of the algorithm provided in Section 5. We use two different acceptance sampling tests in the evaluation: a test based on fixed-size samples and Wald's [62] sequential probability ratio test. While the relative merits of these tests are well understood in the statistics community, recent papers on probabilistic model checking by Sen et al. [60,61] demonstrate a lack of understanding of these methods in the verification community. We find it warranted to include results for two acceptance sampling tests to show that (i) our solution method is not tied to a specific test, as is falsely claimed by Sen et al., and (ii) the sequential probability ratio test almost always is orders of magnitude faster.

We include a brief discussion on simulation effort in Section 7, but we stress that this is not a paper about simulation techniques. Efficient simulation is an orthogonal topic and statistical model checking can immediately benefit from efforts on faster simulation of Markov chains (e.g., by Hordijk et al. [36]) and discrete-event systems (e.g., by McCormack and Sargent [52]).

Section 8 provides a brief overview of related work on probabilistic model checking, including both statistical and numerical solution techniques. In the common case that we want to verify a property in a single initial state or for a probability distribution over initial states, statistical methods generally scale much better than numerical methods as the state space of the model increases. Other benefits of the statistical approach are that (i) it is easy to implement, (ii) it is model independent, (iii) the error analysis is straightforward, and (iv) it is highly amenable to parallelization (this is true even if sequential hypothesis testing is used, as discussed by Younes [66,67]). Numerical methods, on the other hand, are better suited when a small indifference region is required, probabilistic correctness guarantees are unacceptable, or one wants to know the actual probability of satisfaction with high accuracy. Furthermore, numerical methods have the same asymptotic complexity for verifying a property in a single state as in all state simultaneously. This is not the case for the statistical approach, although it will still have much more modest memory requirements.

2 Acceptance Sampling with Bernoulli Trials

A probabilistic model-checking problem can be phrased as a *hypothesis-testing* problem. We will take advantage of this in Section 5 when presenting a statistical approach to probabilistic model checking. As an example of a hypothesis-testing problem, consider the problem of quality control for a manufacturing process. Each manufactured unit is either functional or defective, and there is some probability p, unknown to us, of the process producing a functional unit. Naturally, we want p to be high. Let θ be the lowest acceptable value of p. By inspecting a limited number of manufactured units, we want to determine if the manufacturing process is acceptable (i.e. $p \geq \theta$). This section discusses how to solve such problems statistically using *acceptance sampling*.

Let X_i be a random variable with a Bernoulli distribution ($\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = 1 - p$). X_i is called a *Bernoulli trial*. For the manufacturing process mentioned above, X_i represents the inspection of a manufactured unit, and an observation of X_i, denoted x_i, represents the outcome of the inspection (x_i is 1 if the ith observed unit is functional and 0 if it is defective). To determine if the manufacturing process is acceptable, we need to test the hypothesis $H : p \geq \theta$ against the alternative hypothesis $K : p < \theta$.

Statistical solution techniques generally cannot guarantee a correct result, but this may be acceptable so long as we can bound the probability of error. The

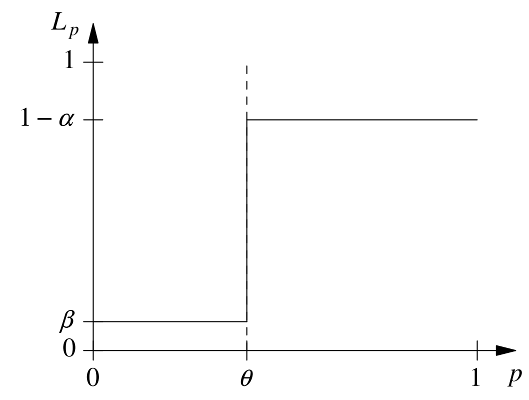


Fig. 1. Probability, L_p, of accepting the hypothesis $H : p \geq \theta$ as a function of p for a hypothetical statistical test.

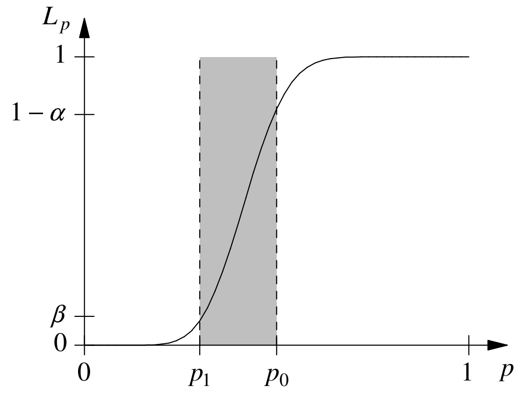


Fig. 2. Probability, L_p, of accepting the hypothesis $H_0 : p \geq p_0$ as a function of p for a test with indifference region.

strength of an acceptance sampling test is determined by two parameters, α and β, where α is a bound on the probability of accepting K when H holds (known as a type I error, or false negative) and β is a bound on the probability of accepting H when K holds (a type II error, or false positive). Fig. 1 plots the probability of accepting H, as a function of p, for a hypothetical acceptance sampling test with ideal performance in the sense that the probability of a type I error is exactly α and the probability of a type II error is exactly β.

The above formulation is problematic and must be relaxed to yield a practical test. For $p = \theta$, the probability of accepting H must be at least $1 - \alpha$, but for p only infinitesimally smaller than θ, the probability of accepting H must not be greater than β. This necessitates exhaustive sampling or using $\beta = 1 - \alpha$. The former is impractical for large sample populations and the latter makes it impossible to ensure a low probability for both types of errors simultaneously. The hypothesis-testing problem is relaxed by introducing two thresholds p_0 and p_1, with $p_0 > p_1$. Instead of testing $H : p \geq \theta$ against $K : p < \theta$, we test $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$. We require 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. Fig. 2 shows the typical performance characteristic for a realistic acceptance sampling test. If the value of p is between p_0 and p_1, we say that we are indifferent with respect to which hypothesis is accepted, and both hypotheses are in fact false in this case. The region (p_1, p_0) is called the *indifference region* and it is shown as a gray area in Fig. 2. By narrowing the indifference region, we can get arbitrarily close to the ideal performance shown in Fig. 1, but this comes at a price since more observations are required to achieve a narrower indifference region.

For probabilistic model checking, we will find it convenient to define the two thresholds, p_0 and p_1, in terms of a single threshold, θ, and the half-width, δ, of the indifference region, i.e. $p_0 = \theta + \delta$ and $p_1 = \theta - \delta$.

A *sample* of size n consists of n observations, $x_1, \ldots, x_n$, of the Bernoulli variates, $X_1, \ldots, X_n$, that represent our experiment. To test $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$, using a single sample of size n, we specify a constant c. If $\sum_{i=1}^{n} x_i$ is greater than c, then H_0 is accepted, otherwise H_1 is accepted. The problem is now to find n and c such that the resulting test has strength $\langle \alpha, \beta \rangle$. The pair $\langle n, c \rangle$, referred to as a *single sampling plan* [25,19], represents an acceptance sampling test that uses a single fixed-size sample.

The sum $Y = \sum_{i=1}^{n} X_i$ of n Bernoulli variates is a random variable with a binomial distribution having cumulative distribution function

$$\Pr[Y \leq c] = F(c; n, p) = \sum_{i=0}^{c} \binom{n}{i} p^i (1-p)^{n-i} \quad . \tag{1}$$

Thus, using a single sampling plan $\langle n, c \rangle$, H_1 is accepted with probability $F(c; n, p)$ and H_0 is accepted with probability $1 - F(c; n, p)$. The sampling plan $\langle n, c \rangle$ has strength $\langle \alpha, \beta \rangle$ if the following conditions are satisfied:

$$F(c; n, p_0) \leq \alpha \tag{2a}$$
$$1 - F(c; n, p_1) \leq \beta \tag{2b}$$

The hypothesis-testing effort is directly proportional to the sample size. For a fixed strength, we should minimize n subject to (2a) and (2b) as this will minimize the effort. The stated optimization problem does not have a closed-form solution except in a few special cases discussed below. In Section 7, we employ the algorithm provided by Younes [66, p. 21], based on binary search, to find an optimal single sampling plan for any choice of p_0, p_1, α, and β.

Example 1 *For probability thresholds $p_0 = 0.5$ and $p_1 = 0.3$, and error bounds $\alpha = 0.2$ and $\beta = 0.1$, the optimal single sampling plan is $\langle 30, 12 \rangle$. This means that we need a sample of size 30, and we accept the hypothesis $p \geq 0.5$ if and only if the sum of the 30 observations exceeds 12. Fig. 2 (p. 5) plots the probability $L_p = 1 - F(12; 30, p)$ of accepting $H_0 : p \geq 0.5$.*

For a few special cases, shown in Table 1, the optimal sample size for a single sampling plan can be expressed precisely as a formula of the test parameters. For the remaining cases, Younes [66, p. 23] derives the following approximation formula for n based on the normal approximation for binomial distributions:

$$\tilde{n} = \frac{\left(\Phi^{-1}(\alpha)\sqrt{p_0(1-p_0)} + \Phi^{-1}(\beta)\sqrt{p_1(1-p_1)}\right)^2}{(p_0 - p_1)^2} \tag{3}$$

Φ^{-1} is the inverse cumulative distribution function for the standard normal

Table 1
Optimal single sampling plans for different choices of p_1 and p_0.

thresholds		optimal single sampling plan	
$p_1 = 0$	$p_0 = 1$	$n = 1$	$c = 0$
$p_1 = 0$	$p_0 < 1$	$n = \left\lceil \dfrac{\log \alpha}{\log(1 - p_0)} \right\rceil$	$c = 0$
$p_1 > 0$	$p_0 = 1$	$n = \left\lceil \dfrac{\log \beta}{\log p_1} \right\rceil$	$c = n - 1$

distribution. According to (3), the sample size for a single sampling plan is approximately inversely proportional to the squared width of the indifference region. The presence of p_0 and p_1 in the numerator indicates that the sample size also depends on the placement of the indifference region. For a fixed width, the sample size is largest if the indifference region is centered around $p = 1/2$, and it decreases if the indifference region is shifted toward $p = 0$ or $p = 1$. From Hasting's [30, p. 191] approximation formula for Φ^{-1}, it follows that n is roughly proportional to the logarithm of α and β. Consequently, decreasing α or β tends to be less costly than narrowing the indifference region.

2.2 Sequential Acceptance Sampling

The sample size for a single sampling plan is fixed and therefore independent of the actual observations made. It is often possible to reduce the *expected* sample size required to achieve a desired test strength by taking the observations into account as they are made. For example, if we use a single sampling plan $\langle n, c \rangle$ and the sum of the first m observations is already greater than c, then we can accept H_0 without making further observations. Conversely, if it is already clear after the first m observations that the sum of all n observations cannot exceed c, then we can safely accept H_1 after making only m observations. The modified test procedure is a simple example of a *sequential* sampling plan: after each observation, we decide whether sufficient information is available to accept either of the two hypotheses or additional observations are required.

Wald's [62] *sequential probability ratio test* is a particularly efficient sequential sampling plan. The reduction in expected sample size, compared to a (sequential) single sampling plan, is often substantial, although there is no fixed upper bound on the sample size so the variance is generally higher.

The sequential probability ratio test is carried out as follows. At the mth stage, i.e. after making m observations $x_1, \ldots, x_m$, we calculate the quantity

$$f_m = \prod_{i=1}^{m} \frac{\Pr[X_i = x_i \mid p = p_1]}{\Pr[X_i = x_i \mid p = p_0]} = \frac{p_1^{d_m}(1-p_1)^{m-d_m}}{p_0^{d_m}(1-p_0)^{m-d_m}} \ , \tag{4}$$

where $d_m = \sum_{i=1}^{m} x_i$. Hypothesis H_0 is accepted if

$$f_m \leq B \ , \tag{5}$$

and hypothesis H_1 is accepted if

$$f_m \geq A \ . \tag{6}$$

Otherwise, additional observations are made until either (5) or (6) is satisfied. A and B, with $A > B$, are chosen so that the probability is at most α of accepting H_1 when H_0 holds, and at most β of accepting H_0 when H_1 holds.

Finding A and B that gives strength $\langle \alpha, \beta \rangle$ is non-trivial. In practice we choose $A = (1-\beta)/\alpha$ and $B = \beta/(1-\alpha)$, which results in a test that very closely matches the prescribed strength. Let the actual strength of this test be $\langle \alpha', \beta' \rangle$. Wald [62, p. 131] shows that $\alpha' \leq \alpha/(1-\beta)$ and $\beta' \leq \beta/(1-\alpha)$. This means that if α and β are small, which typically is the case in practical applications, then α' and β' can only narrowly exceed the target values. Wald [62, p. 132] also shows that $\alpha' + \beta' \leq \alpha + \beta$, so at least one of the inequalities $\alpha' \leq \alpha$ and $\beta' \leq \beta$ must hold, and in practice we often find that both inequalities hold.

Example 2 *Let $p_0 = 0.5$, $p_1 = 0.3$, $\alpha = 0.2$ and $\beta = 0.1$ as in Example 1. If we use $A = (1-\beta)/\alpha$ and $B = \beta/(1-\alpha)$, then we are guaranteed that $\alpha' \leq 0.2/0.9 \approx 0.222$ and $\beta' \leq 0.1/0.8 = 0.125$. Through computer simulation we obtain the estimates $\alpha' \approx 0.175 < \alpha$ and $\beta' \approx 0.082 < \beta$, so the strength of the test is in reality better than $\langle \alpha, \beta \rangle$.*

If $p_0 = 1$ or $p_1 = 0$, then the sequential probability ratio test is equivalent to a sequential single sampling plan, provided that we choose $A = \alpha^{-1}$ and $B = \beta$. Anderson and Friedman [4] call sampling plans of this kind *curtailed single sampling plans* and they prove that such plans are *strongly optimal*. This means that any other sampling plan with at least the same strength *always* requires at least as many observations for all values of p.

The sample size for a sequential acceptance sampling test is a random variable and the expected sample size depends on the unknown parameter p. Let N_p denote the expected sample size as a function of p. An exact formula for N_p does not exist for the sequential probability ratio test, but Table 2 provides approximation formulae for a few cases of special interest, with

$$p_s = \log \frac{1-p_0}{1-p_1} \Big/ \log \frac{p_1(1-p_0)}{p_0(1-p_1)} \ .$$

Table 2
Approximate expected sample size for the sequential probability ratio test.

p	$\tilde{N}_p$
0	$\log \frac{1-\beta}{\alpha} \Big/ \log \frac{1-p_1}{1-p_0}$
p_1	$\left(\beta \log \frac{\beta}{1-\alpha} + (1-\beta) \log \frac{1-\beta}{\alpha}\right) \Big/ \left(p_1 \log \frac{p_1}{p_0} + (1-p_1) \log \frac{1-p_1}{1-p_0}\right)$
p_s	$\left(-\log \frac{\beta}{1-\alpha} \log \frac{1-\beta}{\alpha}\right) \Big/ \left(\log \frac{p_1}{p_0} \log \frac{1-p_0}{1-p_1}\right)$
p_0	$\left((1-\alpha) \log \frac{\beta}{1-\alpha} + \alpha \log \frac{1-\beta}{\alpha}\right) \Big/ \left(p_0 \log \frac{p_1}{p_0} + (1-p_0) \log \frac{1-p_1}{1-p_0}\right)$
1	$\log \frac{\beta}{1-\alpha} \Big/ \log \frac{p_1}{p_0}$

In general, N_p increases from 0 to p_1 and decreases from p_0 to 1. In the indifference region (p_1, p_0), N_p increases from p_1 to some point p' (generally equal to or very near p_s [63, p. 101]) and decreases from p' to p_0.

A remarkable property of the sequential probability ratio test is that it minimizes the expected sample size at both p_0 and p_1 [64]. It is well known, however, that there exist tests that achieve a lower expected sample size for other values of p, in particular if p is in the indifference region (an example of this is given in Section 7). Alternative approaches have therefore been suggested, most notably so called *Bayesian* approaches where the objective is to minimize the expected cost subject to a cost c per observation and a unit cost for accepting a false hypothesis [58,47]. While such alternative formulations of the hypothesis-testing problem are certainly interesting, we will not explore them further in this paper because they represent a departure from the model where the user specifies the desired strength of the test. We refer the reader to Lai [48] for a more detailed account of the developments in the field of sequential hypothesis testing since the ground-breaking work of Wald.

3 Stochastic Discrete-Event Systems

This section formally defines the class of systems for which we develop a model-checking algorithm in Section 5. We rely heavily on the notion of a *stochastic process*, which is any process that evolves over time and whose evolution we can follow and predict in terms of probability [18]. At any point in time, a stochastic process occupies some state. The outcome of observing the state of a stochastic process at a specific time is governed by some probability law.

Definition 3 (Stochastic Process) *Let S and T be two sets. A* stochastic process *is a family of random variables $\mathcal{X} = \{X_t \mid t \in T\}$, with each random variable X_t having range S.*

The index set T in Definition 3 represents time and is typically the set of non-negative integers, $\mathbb{Z}^*$, for discrete-time stochastic processes and the set of non-negative real numbers, $[0, \infty)$, for continuous-time stochastic processes. The set S represents the states that the stochastic process can occupy.

The definition of a stochastic process as a family of random variables is quite general and includes systems with both continuous and discrete dynamics. We will focus our attention on a limited, but important, class of stochastic processes: *stochastic discrete-event systems*. This class includes any stochastic process that can be thought of as occupying a single state for a duration of time before an *event* causes an instantaneous state transition to occur. The canonical example of such a process is a queuing system, with the state being the number of items currently in the queue. The state changes at the occurrence of an event representing the arrival or departure of an item.

3.1 Trajectories

A random variable $X_t \in \mathcal{X}$ represents the chance experiment of observing the stochastic process $\mathcal{X}$ at time t. By recording observations at consecutive time points for all $t \in T$, we obtain a *trajectory*, or *sample path*, for $\mathcal{X}$. The work presented in this paper is centered around the verification of temporal logic formulae over trajectories for stochastic discrete-event systems. The terminology and notation introduced here is used extensively in later sections.

Definition 4 (Trajectory) *A* trajectory *for a stochastic process $\mathcal{X}$ is any sequence of observations $\{x_t \in S \mid t \in T\}$ of the random variables $X_t \in \mathcal{X}$.*

The trajectory of a stochastic discrete-event system is *piecewise constant* and can be represented as a sequence $\sigma = \{\langle s_0, t_0\rangle, \langle s_1, t_1\rangle, \ldots\}$, with $s_i \in S$ and $t_i \in T \setminus \{0\}$. Zero is excluded so that only a single state can be occupied at any time. Fig. 3 plots part of a trajectory for a simple queuing system. Let

$$T_i = \begin{cases} 0 & \text{if } i = 0 \\ \sum_{j=0}^{i-1} t_j & \text{if } i > 0 \end{cases}, \tag{7}$$

i.e. T_i is the time at which state s_i is entered and t_i is the duration of time for which the process remains in s_i before an event triggers a transition to state s_{i+1}. A trajectory σ is then a sequence of observations of $\mathcal{X}$ with $x_t = s_i$ for $T_i \leq t < T_i + t_i$. According to this definition, trajectories of stochastic

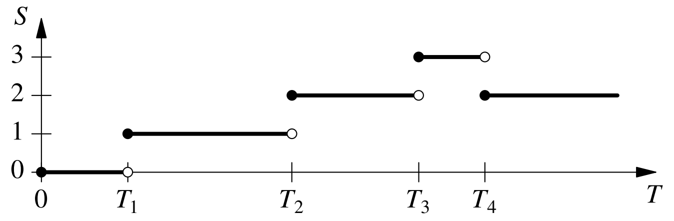


Fig. 3. A trajectory for a simple queuing system with arrival events occurring at T_1, T_2, and T_3, and a departure event occurring at T_4.

discrete-event systems are *right continuous*. A finite trajectory is a sequence $\sigma = \{\langle s_0, t_0\rangle, \ldots, \langle s_n, \infty\rangle\}$ where s_n is an *absorbing* state, meaning that no events can occur in s_n and that $x_t = s_n$ for all $t \geq T_n$.

An infinite trajectory is *convergent* if $\lim_{i\to\infty} T_i < \infty$. For a trajectory to be convergent, however, an infinite sequence of events must occur in a finite amount of time, which is unrealistic for any physical system. Hoel et al. [34] use the term *explosive* to describe processes for which such sequences can occur with non-zero probability. It is common to assume *time divergence* (also called non-Zeno behavior) for infinite trajectories of real-time systems (cf. [3]), i.e. that the systems are non-explosive, and most finite-state systems satisfy this property by default.

3.2 Measurable Stochastic Discrete-Event Systems

Of utmost importance to probabilistic model checking is the definition of a *probability measure* over sets of trajectories for a system. Formally, a *measurable space* is a set Ω with a σ-algebra $\mathcal{F}_\Omega$ of subsets of Ω [26]. A *probability space* is a measurable space $\langle \Omega, \mathcal{F}_\Omega\rangle$ and a probability measure μ.

For stochastic discrete-event systems, the elements of the σ-algebra are sets of trajectories with common *prefix*. A prefix of $\sigma = \{\langle s_0, t_0\rangle, \langle s_1, t_1\rangle, \ldots\}$ is a sequence $\sigma_{\leq\tau} = \{\langle s'_0, t'_0\rangle, \ldots, \langle s'_k, t'_k\rangle\}$, with $s'_i = s_i$ for all $i \leq k$, $\sum_{i=0}^{k} t'_i = \tau$, $t'_i = t_i$ for all $i < k$, and $t'_k < t_k$. Let $\mathit{Path}(\sigma_{\leq\tau})$ denote the set of trajectories with prefix $\sigma_{\leq\tau}$. This set must be measurable so that we can talk about the probability of a system exhibiting certain behavior. This requirement is not a problem in practice—the set of trajectories of a stochastic discrete-event system is measurable if the sets S and T are measurable. Let $\mathcal{F}_S$ be a σ-algebra over S and $\mathcal{F}_T$ a σ-algebra over T. Then the *cylinder set* $C(\sigma_{\leq\tau}, I_k, S_{k+1}, \ldots, I_{n-1}, S_n)$, with $S_i \in \mathcal{F}_S$ and $I_i \in \mathcal{F}_T$, denotes the set of trajectories $\sigma = \{\langle s'_0, t'_0\rangle, \langle s'_1, t'_1\rangle, \ldots\}$ such that $s'_i = s_i$ for $i \leq k$, $s'_i \in S_i$ for $k < i \leq n$, $t'_i = t_i$ for $i < k$, $t'_k > t_k$, and $t'_i \in I_i$ for $k \leq i < n$. The σ-algebra over the set $\mathit{Path}(\sigma_{\leq\tau})$ can be defined using element-wise set operations on cylinder sets. This is analogous to Baier et al.'s [9] definition of a σ-algebra on the set of trajectories for a continuous-time Markov chain (see also, Segala's [59] definition of *trace distributions*).

3.3 Structured Stochastic Discrete-Event Systems

So far, we have defined stochastic discrete-event systems in rather general terms as any stochastic process with piecewise constant trajectories. Most stochastic discrete-event system of interest have more structure than so.

The probability measure on sets of trajectories for a stochastic discrete-event system can be expressed using a holding time distribution with density function $h(\cdot;\sigma_{\leq\tau})$ and a next-state distribution $p(\cdot;\sigma_{\leq\tau},t)$. The probability measure for $C(\sigma_{\leq\tau},I_k,S_{k+1},\ldots,I_{n-1},S_n)$ can then be defined recursively as

$$\mu(C(\sigma_{\leq\tau},I_k,S_{k+1},\ldots,I_{n-1},S_n)) = \\ \int\limits_{I_k} h(t_k+t;\sigma_{\leq\tau}) \int\limits_{S_{k+1}} p(s;\sigma_{\leq\tau},t)\cdot\mu(C(\sigma_{\leq\tau}\oplus\langle t,s\rangle,I_{k+1},S_{k+2},\ldots,I_{n-1},S_n)) \ ,$$

where $\{\langle s_0,t_0\rangle,\ldots,\langle s_k,t_k\rangle\}\oplus\langle t,s\rangle = \{\langle s_0,t_0\rangle,\ldots,\langle s_k,t_k+t\rangle,\langle s,0\rangle\}$. The base case for the recursive definition is $\mu(C(\sigma_{\leq\tau}))=1$.

By making limiting assumptions regarding the shape of the distributions $h(\cdot;\sigma_{\leq\tau})$ and $p(\cdot;\sigma_{\leq\tau},t)$, we enable a succinct representation of μ. This is important for efficient generation of sample trajectories for stochastic discrete-event systems, which is a large component of our statistical model-checking algorithm. A common assumption is that $h(\cdot;\sigma_{\leq\tau})$ and $p(\cdot;\sigma_{\leq\tau},t)$ are independent of history, which gives us *Markov chains*. Discrete-time Markov chains have geometric holding time distributions for each state. For continuous-time Markov chains, the holding time in state s is exponentially distributed: $h(t;s)=\lambda_s\,\mathrm{e}^{-\lambda_s t}$. The parameter λ_s is the *exit rate* for state s. With general positive holding time distributions, we have *semi-Markov processes*. More detailed accounts on Markov chains and semi-Markov processes are provided by, for example, Kolmogoroff [44], Doob [18], Bartlett [11], Howard [37,38], and Çinlar [14]. The *generalized semi-Markov process*, first introduced by Matthes [51], permits even further history dependence. It is an established formalism in queuing theory for modeling stochastic discrete-event systems with focus on the event structure of a system [23].

In addition to using a structured representation of the probability measure on sets of trajectories, it is often natural to describe the state of a system by using multiple state variables. A state variable could represent, for example, the number of elements in a queue or the status of a machine component.

Definition 5 (Factored State Representation) *A* factored *representation of a state space S consists of a set of state variables SV and a value assignment* *function $V(s, x)$ providing the value of $x \in SV$ in state $s \in S$. The domain of x is the set $D_x = \bigcup_{s \in S} V(s, x)$ of possible values that x can take on. A tuple $\langle S, T, \mu, SV, V \rangle$ represents a* factored stochastic discrete-event system.

4 Specifying Properties of Stochastic Discrete-Event Systems

To enable automatic verification of stochastic discrete-event systems, a formalism is needed for expressing interesting properties of such systems. This section introduces the *Unified Temporal Stochastic Logic* (UTSL), which can be used to express properties such as "the probability is at most 0.01 that a call is dropped within a 60-minute period." UTSL has essentially the same syntax as the existing logics PCTL and CSL, but it has a unified semantics for both discrete-time and continuous-time systems, as well as systems with discrete, continuous, and general state spaces. This will allow us, for the most part, to treat all stochastic discrete-event systems uniformly when presenting a statistical approach to probabilistic model checking in Section 5.

4.1 Temporal and Probabilistic Logic

The use of *temporal logic* [57] for specifying properties of deterministic and nondeterministic systems with program verification in mind was pioneered by Pnueli [55] and is now a wide-spread practice in the model-checking community. The propositional branching-time logic CTL [16], a particularly popular formalism, can be used to express properties such as "for all trajectories, Ψ eventually becomes true with Φ holding continuously until then" and "there exists a trajectory such that Φ holds after the next state transition."

For many real-time systems, it is important to ensure that deadlines are met. Extensions of CTL with time as a discrete (RTCTL [20]) or continuous (TCTL [2]) quantity have therefore been proposed. With RTCTL and TCTL, it is possible to express timed properties such as "for all trajectories, Φ becomes true within t time units." The logic TCTL has also been proposed as a formalism for expressing properties of continuous-time stochastic systems, but with "for all trajectories" ($\forall$) and "there exists a trajectory" ($\exists$) reinterpreted as "with probability one" and "with positive probability," respectively [1]. The same interpretation is used in earlier work by Hart and Sharir [29] on the branching-time logic PTL for discrete-time stochastic processes.

It is often not economically or physically feasible to ensure that certain behaviors occur with probability one, but simply guaranteeing that a behavior can be exhibited by the system with positive probability may be too weak. For example, designing a telephone system where no call is ever dropped would be excessively costly, but it is not enough to know that a call can possibly go through. For the telephone system, we would like to ensure that calls go through with high probability, for example 0.9999. Neither TCTL nor PTL permit us to express such a property, but the probabilistic logic PCTL [27,28] does. PCTL has quantitative time bounds just as RTCTL, on which PCTL is based, but the path quantifiers $\forall$ and $\exists$ are replaced by a single probabilistic path quantifier. This lets us express quantitative bounds on the probability of a set of trajectories. For example, PCTL can express the property "with probability at least θ, Φ will be satisfied within t time units."

PCTL formulae are interpreted over discrete-time Markov chains. A similar logic, CSL, with formulae interpreted over continuous-time Markov chains, has been proposed by Aziz et al. [5,6]. Baier et al. [10,9] introduce a variation of CSL, which includes a facility for expressing bounds on steady-state probabilities. This version of CSL has also been used for expressing properties of semi-Markov processes [40]. Yet another logic, with essentially the same syntax as PCTL, has been proposed for expressing properties of probabilistic timed automata [46]. While the difference in syntax is minimal between all mentioned logics for expressing probabilistic real-time properties, the semantics of the various logics are tied to specific classes of stochastic processes.

4.2 UTSL: The Unified Temporal Stochastic Logic

To enable the use of a single logic for different classes of systems, we introduce the logic UTSL, with a unified semantics for all measurable stochastic discrete-event systems. The syntactic structure of UTSL is the same as that of both CSL (without the steady-state operator) and PCTL.

Definition 6 (UTSL Syntax) *For a factored stochastic discrete-event system, $\mathcal{M} = \langle S, T, \mu, SV, V \rangle$ (Definition 5), the syntax for UTSL is defined as follows:*

(1) $x \sim v$ is a UTSL formula for $x \in SV$, $v \in D_x$, and $\sim \in \{\leq, =, \geq\}$.
(2) $\neg\Phi$ is a UTSL formula if Φ is a UTSL formula.
(3) $\Phi \wedge \Psi$ is a UTSL formula if both Φ and Ψ are UTSL formulae.
(4) $\mathcal{P}_{\bowtie\theta}[X^I\ \Phi]$, for $\bowtie \in \{\leq, \geq\}$, $\theta \in [0,1]$, and $I \subset T$, is a UTSL formula if Φ is a UTSL formula.
(5) $\mathcal{P}_{\bowtie\theta}[\Phi\ \mathcal{U}^I\ \Psi]$, for $\bowtie \in \{\leq, \geq\}$, $\theta \in [0,1]$, and $I \subset T$, is a UTSL formula if both Φ and Ψ are UTSL formulae.

The standard logic operators have their usual meaning. $\mathcal{P}_{\bowtie\theta}[\varphi]$ asserts that the probability measure over the set of trajectories satisfying the path formula φ is related to θ according to $\bowtie$. Path formulae are constructed using the temporal path operators X^I ("next") and $\mathcal{U}^I$ ("until"). The path formula $X^I\ \Phi$ asserts that the next state transition occurs $t \in I$ time units into the future and that Φ holds in the next state, while $\Phi\ \mathcal{U}^I\ \Psi$ asserts that Ψ becomes true $t \in I$ time units into the future while Φ holds continuously prior to t.

Definition 6 provides a bare-bones version of UTSL. Additional formulae are derived in the usual way. For example, $\perp \equiv (x = v) \wedge \neg(x = v)$ for some $x \in SV$ and $v \in D_x$, $\top \equiv \neg\perp$, $x < v \equiv \neg(x \geq v)$, $\Phi \vee \Psi \equiv \neg(\neg\Phi \wedge \neg\Psi)$, $\Phi \rightarrow \Psi \equiv \neg\Phi \vee \Psi$, and $\mathcal{P}_{<\theta}[\varphi] \equiv \neg\mathcal{P}_{\geq\theta}[\varphi]$. The path operators $\mathcal{W}$ ("weak until"), $\Diamond$ ("eventually"), and $\Box$ ("continuously") are derived as follows [28]:

$$\begin{aligned}
\mathcal{P}_{\geq\theta}[\Phi\ \mathcal{W}^I\ \Psi] &\equiv \mathcal{P}_{\leq 1-\theta}[\neg\Psi\ \mathcal{U}^I\ \neg(\Phi \vee \Psi)] \\
\mathcal{P}_{\leq\theta}[\Phi\ \mathcal{W}^I\ \Psi] &\equiv \mathcal{P}_{\geq 1-\theta}[\neg\Psi\ \mathcal{U}^I\ \neg(\Phi \vee \Psi)] \\
\mathcal{P}_{\bowtie\theta}[\Diamond^I\ \Phi] &\equiv \mathcal{P}_{\bowtie\theta}[\top\ \mathcal{U}^I\ \Phi] \\
\mathcal{P}_{\bowtie\theta}[\Box^I\ \Phi] &\equiv \mathcal{P}_{\bowtie\theta}[\Phi\ \mathcal{W}^I\ \perp]
\end{aligned}$$

Unbounded versions of all path operators are obtained by letting I equal the time domain T. For example, $\mathcal{P}_{\bowtie\theta}[\Phi\ \mathcal{U}\ \Psi] \equiv \mathcal{P}_{\bowtie\theta}[\Phi\ \mathcal{U}^T\ \Psi]$.

4.3 UTSL Semantics and Model-Checking Problems

The validity of a UTSL formula is determined relative to a trajectory prefix. For simple UTSL formulae of the form $x \sim v$, the validity depends only on the last state of the trajectory prefix, but this is not necessarily the case for UTSL formulae containing one or more probabilistic operators. The formal semantics of UTSL is given by the following inductive definition.

Definition 7 (UTSL Semantics) *Let $\mathcal{M} = \langle S, T, \mu, SV, V\rangle$ be a factored stochastic discrete-event system (Definition 5). With $Path(\sigma_{\leq\tau})$ being the set of trajectories with prefix $\sigma_{\leq\tau}$ and the definition of T_i given by* (7)*, satisfaction relations for UTSL formulae are defined by the following rules:*

$$\begin{aligned}
\mathcal{M}, \{\langle s_0, t_0\rangle, \ldots, \langle s_k, t_k\rangle\} \models x \sim v \quad & \textit{if } V(s_k, x) \sim v \\
\mathcal{M}, \sigma_{\leq\tau} \models \neg\Phi \quad & \textit{if } \mathcal{M}, \sigma_{\leq\tau} \not\models \Phi \\
\mathcal{M}, \sigma_{\leq\tau} \models \Phi \wedge \Psi \quad & \textit{if } (\mathcal{M}, \sigma_{\leq\tau} \models \Phi) \wedge (\mathcal{M}, \sigma_{\leq\tau} \models \Psi) \\
\mathcal{M}, \sigma_{\leq\tau} \models \mathcal{P}_{\bowtie\theta}[\varphi] \quad & \textit{if } \mu(\{\sigma \in Path(\sigma_{\leq\tau}) \mid \mathcal{M}, \sigma, \tau \models \varphi\}) \bowtie \theta
\end{aligned}$$

$$\begin{aligned}
\mathcal{M}, \sigma, \tau \models X^I\ \Phi \quad & \textit{if } \exists k \in \mathbb{Z}^+ . \Big((T_{k-1} \leq \tau) \wedge (\tau < T_k) \wedge (T_k - \tau \in I) \\
& \qquad \wedge (\mathcal{M}, \sigma_{\leq T_k} \models \Phi)\Big) \\
\mathcal{M}, \sigma, \tau \models \Phi\ \mathcal{U}^I\ \Psi \quad & \textit{if } \exists t \in I . \Big((\mathcal{M}, \sigma_{\leq\tau+t} \models \Psi) \\
& \qquad \wedge \forall t' \in T . \big((t' < t) \rightarrow (\mathcal{M}, \sigma_{\leq\tau+t'} \models \Phi)\big)\Big)
\end{aligned}$$

Definition 7 specifies the validity of a UTSL formula at any time during execution of a stochastic discrete-event system. Note that the validity of a path formula is determined relative to an entire trajectory σ and a time point τ along the trajectory, rather than a trajectory prefix $\sigma_{\leq\tau}$.

The semantics of $\Phi\ \mathcal{U}^I\ \Psi$ requires that Φ holds continuously, i.e. at every point in time, along a trajectory until Ψ is satisfied. This is consistent with the semantics of time-bounded until for TCTL [1], but not with Infante López et al.'s [40] semantics of CSL for semi-Markov processes, which requires Φ to hold only at the time of state transitions. As shown in Appendix A, Φ may hold immediately at the entry of a state s and also immediately after a transition from s to s', but still not hold continuously in s; or Ψ may hold at some point in time while the system remains in s, and not hold immediately upon entry to s nor immediately after a transition from s to s'. It is therefore not sufficient, in general, to verify Φ and Ψ at discrete points along a trajectory. It is sufficient to do so, however, if the *Markov property* holds:

$$\mu(\mathit{Path}(\{\langle s_0, t_0\rangle, \ldots, \langle s_k, t_k\rangle\})) = \mu(\mathit{Path}(\{\langle s_k, 0\rangle\})) \tag{8}$$

It follows from (8) that our semantics for UTSL coincides with the semantics for PCTL interpreted over discrete-time Markov chains [28] and CSL interpreted over continuous-time Markov chains [9]. Our solution method is restricted to Markov chains for nested probabilistic statements.

While the semantics of Infante López et al. makes it easier to verify properties with nested probabilistic operators, it is not consistent with the common definition of a trajectory for a continuous-time discrete-event system as a piecewise linear function of time. Furthermore, one could imagine using phase-type distributions to approximate non-memoryless distributions and verify properties for the resulting Markov chain. The introduction of phase transitions would result in nested formulae possibly being verified at different times in the same state, which is incompatible with the semantics of Infante López et al.

We typically want to know whether a property Φ holds for a model $\mathcal{M}$ if execution starts in a specific state s. More generally, we can define the validity of a UTSL formula relative to a probability measure μ_0, where $\mu_0(S')$ is the probability that execution starts in a state $s \in S'$:

$$\begin{array}{ll}
\mathcal{M}, \mu_0 \models x \sim v & \text{if } \forall s \in \operatorname{supp}\mu_0 . \Big(\mathcal{M}, \{\langle s, 0\rangle\} \models x \sim v\Big) \\
\mathcal{M}, \mu_0 \models \neg\Phi & \text{if } \mathcal{M}, \mu_0 \not\models \Phi \\
\mathcal{M}, \mu_0 \models \Phi \wedge \Psi & \text{if } (\mathcal{M}, \mu_0 \models \Phi) \wedge (\mathcal{M}, \mu_0 \models \Psi) \\
\mathcal{M}, \mu_0 \models \mathcal{P}_{\bowtie\theta}[\varphi] & \text{if } \displaystyle\int \mu(\{\sigma \in \mathit{Path}(\{\langle s, 0\rangle\}) \mid \mathcal{M}, \sigma, 0 \models \varphi\})\ \mathrm{d}\mu_0(S) \bowtie \theta
\end{array}$$

A UTSL model-checking problem can now be specified as a triple $\langle \mathcal{M}, \mu_0, \Phi\rangle$.

Consider the model-checking problem $\langle \mathcal{M}, s, \mathcal{P}_{\bowtie\theta}[\varphi]\rangle$ and let p be the probability measure for the set of trajectories that start in s and satisfy φ. If p is "sufficiently close" to θ, then it is likely to make little difference to a user whether or not $\mathcal{P}_{\bowtie\theta}[\varphi]$ is reported to hold by a model-checking algorithm.

To formalize this idea, we introduce UTSL$_\delta$ as a relaxation of UTSL. With each formula of the form $\mathcal{P}_{\bowtie\theta}[\varphi]$, we associate an indifference region centered around θ with half-width $\delta(\theta)$. If $|p-\theta| < \delta(\theta)$, then the truth value of $\mathcal{P}_{\bowtie\theta}[\varphi]$ is undetermined for UTSL$_\delta$; otherwise, it is the same as for UTSL.

Definition 8 (UTSL$_\delta$ Semantics) *Let $\delta(\theta)$ be the half-width of an indifference region centered around θ, and let $\mathcal{M} = \langle S, T, \mu, SV, V\rangle$ be a factored stochastic discrete-event system. Satisfaction relations $\mathrel{|\!\approx}^{\delta}_{\top}$ and unsatisfaction relations $\mathrel{|\!\approx}^{\delta}_{\bot}$ for UTSL$_\delta$ are simultaneously defined by induction as follows:*

$$
\begin{array}{rl}
\mathcal{M}, \{\langle s_0, t_0\rangle, \ldots, \langle s_k, t_k\rangle\} \mathrel{|\!\approx}^{\delta}_{\top} x \sim v & \textit{if } V(s_k, x) \sim v \\
\mathcal{M}, \{\langle s_0, t_0\rangle, \ldots, \langle s_k, t_k\rangle\} \mathrel{|\!\approx}^{\delta}_{\bot} x \sim v & \textit{if } V(s_k, x) \nsim v \\
\mathcal{M}, \sigma_{\le\tau} \mathrel{|\!\approx}^{\delta}_{\top} \neg\Phi & \textit{if } \mathcal{M}, \sigma_{\le\tau} \mathrel{|\!\approx}^{\delta}_{\bot} \Phi \\
\mathcal{M}, \sigma_{\le\tau} \mathrel{|\!\approx}^{\delta}_{\bot} \neg\Phi & \textit{if } \mathcal{M}, \sigma_{\le\tau} \mathrel{|\!\approx}^{\delta}_{\top} \Phi \\
\mathcal{M}, \sigma_{\le\tau} \mathrel{|\!\approx}^{\delta}_{\top} \Phi \wedge \Psi & \textit{if } (\mathcal{M}, \sigma_{\le\tau} \mathrel{|\!\approx}^{\delta}_{\top} \Phi) \wedge (\mathcal{M}, \sigma_{\le\tau} \mathrel{|\!\approx}^{\delta}_{\top} \Psi) \\
\mathcal{M}, \sigma_{\le\tau} \mathrel{|\!\approx}^{\delta}_{\bot} \Phi \wedge \Psi & \textit{if } (\mathcal{M}, \sigma_{\le\tau} \mathrel{|\!\approx}^{\delta}_{\bot} \Phi) \vee (\mathcal{M}, \sigma_{\le\tau} \mathrel{|\!\approx}^{\delta}_{\bot} \Psi) \\
\mathcal{M}, \sigma_{\le\tau} \mathrel{|\!\approx}^{\delta}_{\top} \mathcal{P}_{\ge\theta}[\varphi] & \textit{if } \mu(\{\sigma \in \textit{Path}(\sigma_{\le\tau}) \mid \mathcal{M}, \sigma, \tau \mathrel{|\!\approx}^{\delta}_{\top} \varphi\}) \\
 & \quad \ge \theta + \delta(\theta) \\
\mathcal{M}, \sigma_{\le\tau} \mathrel{|\!\approx}^{\delta}_{\bot} \mathcal{P}_{\ge\theta}[\varphi] & \textit{if } \mu(\{\sigma \in \textit{Path}(\sigma_{\le\tau}) \mid \mathcal{M}, \sigma, \tau \mathrel{|\!\approx}^{\delta}_{\bot} \varphi\}) \\
 & \quad \ge 1 - (\theta - \delta(\theta)) \\
\mathcal{M}, \sigma_{\le\tau} \mathrel{|\!\approx}^{\delta}_{\top} \mathcal{P}_{\le\theta}[\varphi] & \textit{if } \mu(\{\sigma \in \textit{Path}(\sigma_{\le\tau}) \mid \mathcal{M}, \sigma, \tau \mathrel{|\!\approx}^{\delta}_{\top} \varphi\}) \\
 & \quad \le \theta - \delta(\theta) \\
\mathcal{M}, \sigma_{\le\tau} \mathrel{|\!\approx}^{\delta}_{\bot} \mathcal{P}_{\le\theta}[\varphi] & \textit{if } \mu(\{\sigma \in \textit{Path}(\sigma_{\le\tau}) \mid \mathcal{M}, \sigma, \tau \mathrel{|\!\approx}^{\delta}_{\bot} \varphi\}) \\
 & \quad \le 1 - (\theta + \delta(\theta))
\end{array}
$$

$$
\begin{array}{ll}
\mathcal{M}, \sigma, \tau \mathrel{|\!\approx}^{\delta}_{\top} X^I\, \Phi & \textit{if } \exists k \in \mathbb{Z}^+.\big((T_{k-1} \le \tau) \wedge (\tau < T_k) \wedge (T_k - \tau \in I) \\
 & \qquad\qquad \wedge (\mathcal{M}, \sigma_{\le T_k} \mathrel{|\!\approx}^{\delta}_{\top} \Phi)\big) \\
\mathcal{M}, \sigma, \tau \mathrel{|\!\approx}^{\delta}_{\bot} X^I\, \Phi & \textit{if } \forall k \in \mathbb{Z}^+.\big(\big((T_{k-1} \le \tau) \wedge (\tau < T_k) \wedge (T_k - \tau \in I)\big) \\
 & \qquad\qquad \to (\mathcal{M}, \sigma_{\le T_k} \mathrel{|\!\approx}^{\delta}_{\bot} \Phi)\big) \\
\mathcal{M}, \sigma, \tau \mathrel{|\!\approx}^{\delta}_{\top} \Phi\, \mathcal{U}^I\, \Psi & \textit{if } \exists t \in I.\big((\mathcal{M}, \sigma_{\le\tau+t} \mathrel{|\!\approx}^{\delta}_{\top} \Psi) \\
 & \qquad\qquad \wedge \forall t' \in T.\big((t' < t) \to (\mathcal{M}, \sigma_{\le\tau+t'} \mathrel{|\!\approx}^{\delta}_{\top} \Phi)\big)\big) \\
\mathcal{M}, \sigma, \tau \mathrel{|\!\approx}^{\delta}_{\bot} \Phi\, \mathcal{U}^I\, \Psi & \textit{if } \forall t \in I.\big((\mathcal{M}, \sigma_{\le\tau+t} \mathrel{|\!\approx}^{\delta}_{\bot} \Psi) \\
 & \qquad\qquad \vee \exists t' \in T.\big((t' < t) \wedge (\mathcal{M}, \sigma_{\le\tau+t'} \mathrel{|\!\approx}^{\delta}_{\bot} \Phi)\big)\big)
\end{array}
$$

A model-checking problem $\langle \mathcal{M}, s, \Phi \rangle$ may very well belong to neither of the two relations $\approx\!\!\!\models^{\delta}_{\top}$ and $\approx\!\!\!\models^{\delta}_{\bot}$. It is then assumed that the user is indifferent with respect to whether Φ truly holds or not.

5 Probabilistic Model Checking using Acceptance Sampling

This section presents a statistical approach to probabilistic model checking, employing hypothesis testing (as described in Section 2) and discrete-event simulation. The proposed solution method works for any discrete-event system that can be simulated, although the method for verifying properties with nested probabilistic statements is limited to discrete-time systems or systems satisfying the Markov property. We prove two fundamental theorems that establish efficient verification procedures for conjunctive and nested probabilistic statements and we provide complexity results for the solution method.

The algorithm that we present is for UTSL$_\delta$ model checking. Let $\mathcal{M}, \sigma_{\leq\tau} \vdash \Phi$ represent the fact that Φ is accepted as true and $\mathcal{M}, \sigma_{\leq\tau} \nvdash \Phi$ that Φ is rejected as false by our statistical model-checking algorithm. For the remainder of this section, we will often leave out $\mathcal{M}$ from relations for the sake of brevity. We require that our model-checking algorithm satisfies the following conditions:

$$\Pr[\sigma_{\leq\tau} \nvdash \Phi \mid \sigma_{\leq\tau} \approx\!\!\!\models^{\delta}_{\top} \Phi] \leq \alpha \tag{9}$$

$$\Pr[\sigma_{\leq\tau} \vdash \Phi \mid \sigma_{\leq\tau} \approx\!\!\!\models^{\delta}_{\bot} \Phi] \leq \beta \tag{10}$$

The model-checking algorithm is required either to accept a formula as true or reject it as false. It follows, for instance, that $\Pr[\sigma_{\leq\tau} \vdash \Phi \mid \sigma_{\leq\tau} \approx\!\!\!\models^{\delta}_{\top} \Phi]$ must be at least $1-\alpha$, so there should be a high probability of accepting Φ as true when it holds according to the semantics of UTSL$_\delta$.

The parameter α bounds the probability of a type I error (false negative) and β bounds the probability of a type II error (false positive) for UTSL$_\delta$ model checking. By decreasing δ, we can get arbitrarily close to a statistical algorithm for UTSL model checking, although this will most certainly come at a cost.

5.1 Model Checking without Nested Probabilistic Operators

Let us now consider the problem of verifying a formula Φ relative to a trajectory prefix so that conditions (9) and (10) are satisfied. Here, we assume that Φ has no nested probabilistic operators (nesting is considered in Section 5.2). If Φ is of the form $x \sim v$, then it is trivial to satisfy the two conditions for any α and β. Given a trajectory prefix $\{\langle s_0, t_0\rangle, \ldots, \langle s_k, t_k\rangle\}$, we can simply observe the value of x in state s_k and compare it to v without error.

5.1.1 Probabilistic Operator

To verify the formula $\mathcal{P}_{\bowtie\theta}[\varphi]$, we introduce Bernoulli variates X_i with parameter p, where p is the probability measure of the set of trajectories that satisfy φ. An observation of X_i can be obtained by first generating a trajectory for $\mathcal{M}$ using discrete-event simulation and then verifying φ over the sampled trajectory. If φ does not contain any probabilistic operators, as is assumed for now, then we can verify φ without error. If φ is determined to hold over the sampled trajectory, then the observation is 1, otherwise it is 0.

We can now set up a hypothesis-testing problem for verifying $\mathcal{P}_{\geq\theta}[\varphi]$. We should test the hypothesis $H_0 : p \geq \theta + \delta(\theta)$ against the alternative hypothesis $H_1 : p \leq \theta - \delta(\theta)$ (for $\mathcal{P}_{\leq\theta}[\varphi]$, we simply reverse the roles of the two hypotheses). H_0 holds if and only if $\sigma_{\leq\tau} \approx\!\!\!\models^{\delta}_{\top} \mathcal{P}_{\geq\theta}[\varphi]$ holds, and H_1 is similarly related to $\sigma_{\leq\tau} \approx\!\!\!\models^{\delta}_{\bot} \mathcal{P}_{\geq\theta}[\varphi]$. Thus, by using an acceptance sampling test with strength $\langle\alpha, \beta\rangle$ to decide $\sigma_{\leq\tau} \vdash \mathcal{P}_{\geq\theta}[\varphi]$, we can satisfy conditions (9) and (10).

Trajectories for a stochastic discrete-event system may be infinite. For the proposed algorithm to terminate (with probability one), it must suffice to examine a finite prefix of any trajectory σ to determine the truth value of φ over σ. This surely is the case if φ is $X^I\ \Phi$ because we only need to look ahead one state. If φ is $\Phi\ \mathcal{U}^I\ \Psi$, then a sufficient condition for termination is that $\sup I$ is finite and the model is non-explosive. In some cases, termination may be guaranteed even for unbounded until formulae, but this requires a model such that an absorbing state or a state satisfying $\neg\Phi \vee \Psi$ is reachable from the initial state with probability one.

5.1.2 Composite State Formulae

To complete the model-checking algorithm, we need to specify how to verify negated and conjunctive formulae. We take a compositional approach to verification of such formulae. To verify $\neg\Phi$, we verify Φ and reverse the result. To verify a conjunction, we verify each conjunct separately. The following rules formally define the behavior of the model-checking algorithm:

$$\begin{array}{ll} \mathcal{M}, \sigma_{\leq\tau} \vdash \neg\Phi & \text{if } \mathcal{M}, \sigma_{\leq\tau} \nvdash \Phi \\ \mathcal{M}, \sigma_{\leq\tau} \vdash \Phi \wedge \Psi & \text{if } (\mathcal{M}, \sigma_{\leq\tau} \vdash \Phi) \wedge (\mathcal{M}, \sigma_{\leq\tau} \vdash \Psi) \end{array}$$

Next, we show how to bound the probability of error for a composite formula, assuming that we have bounds for the probability of error for subformulae.

First, consider the verification of $\neg\Phi$, assuming we have already verified Φ so that conditions (9) and (10) are satisfied. Since we negate the verification result for Φ, a type I error for Φ becomes a type II error for $\neg\Phi$, and a type II error for Φ becomes a type I error for $\neg\Phi$. To verify $\neg\Phi$ with error bounds α

and β, we therefore have to verify Φ with error bounds β and α.

Next, consider the verification of $\Phi \wedge \Psi$. A type I error occurs if we believe that at least one of Φ and Ψ does not hold, when in reality both are true. A type II error occurs if we believe that both Φ and Ψ hold, when at least one of the conjuncts actually is false. We will show that in order to verify a conjunction with error bounds α and β, we can use the same type II error bound for each conjunct, but we must use a tighter type I error bound. To prove this, we first derive general bounds on the error probabilities associated with the verification of a conjunction of size n:

Theorem 9 (Conjunction) *Let $\Phi = \bigwedge_{i=1}^{n} \Phi_i$. If Φ_i is verified with type I error bound α_i and type II error bound β_i for all $1 \leq i \leq n$, then Φ can be verified with type I error bound $\sum_{i=1}^{n} \alpha_i$ and type II error bound $\max_{1 \leq i \leq n} \beta_i$.*

PROOF. From elementary probability theory we have that $\Pr[A \wedge B] \leq \Pr[A] + \Pr[B]$ and, by induction, $\Pr[\bigwedge_{i=1}^{n} A_i] \leq \sum_{i=1}^{n} \Pr[A_i]$. It follows immediately that $\sum_{i=1}^{n} \alpha_i$ is a bound on the probability of a type I error for a conjunction $\bigwedge_{i=1}^{n} \Phi_i$ if the type I error bound is α_i for each conjunct Φ_i.

Assume that $\Phi = \bigwedge_{i=1}^{n} \Phi_i$, for some $n \geq 1$, can be verified with type II error probability $\beta = \max_{1 \leq i \leq n} \beta_i$. Furthermore, assume that $\Pr[\sigma_{\leq \tau} \vdash \Phi_{n+1} \mid \sigma_{\leq \tau} \not\approx_{\perp}^{\delta} \Phi_{n+1}] \leq \beta_{n+1}$. A type II error for the verification of $\Phi \wedge \Phi_{n+1}$ occurs if both Φ and Φ_{n+1} are verified as true when either $\sigma_{\leq \tau} \not\approx_{\perp}^{\delta} \Phi$ or $\sigma_{\leq \tau} \not\approx_{\perp}^{\delta} \Phi_{n+1}$ holds:

$$\begin{aligned}&\Pr[(\sigma_{\leq \tau} \vdash \Phi) \wedge (\sigma_{\leq \tau} \vdash \Phi_{n+1}) \mid \sigma_{\leq \tau} \not\approx_{\perp}^{\delta} \Phi] \\ &\qquad \leq \min(\Pr[\sigma_{\leq \tau} \vdash \Phi \mid \sigma_{\leq \tau} \not\approx_{\perp}^{\delta} \Phi], \Pr[\sigma_{\leq \tau} \vdash \Phi_{n+1}]) \leq \min(\beta, 1) = \beta\end{aligned}$$

$$\begin{aligned}&\Pr[(\sigma_{\leq \tau} \vdash \Phi) \wedge (\sigma_{\leq \tau} \vdash \Phi_{n+1}) \mid \sigma_{\leq \tau} \not\approx_{\perp}^{\delta} \Phi_{n+1}] \\ &\leq \min(\Pr[\sigma_{\leq \tau} \vdash \Phi], \Pr[\sigma_{\leq \tau} \vdash \Phi_{n+1} \mid \sigma_{\leq \tau} \not\approx_{\perp}^{\delta} \Phi_{n+1}]) \leq \min(1, \beta_{n+1}) = \beta_{n+1}\end{aligned}$$

We take the maximum over these cases to obtain the bound $\max_{1 \leq i \leq n+1} \beta_i$. □

A verification procedure for conjunction follows immediately from Theorem 9.

Corollary 10 *To verify $\bigwedge_{i=1}^{n} \Phi_i$ with type I error probability α and type II error probability β, it is sufficient to verify each conjunct Φ_i with type I error probability α/n and type II error probability β.*

Intuitively, we need to verify each conjunct with a tighter type I error bound than the type I error bound we desire for the whole conjunction because rejection of any one conjunct as false leads to rejection of the whole conjunction. Of course, it is not necessary to distribute the type I error bound uniformly over the conjuncts, so long as $\alpha = \sum_{i=1}^{n} \alpha_i$. The result of Younes [66], that $\alpha_i = \alpha$ suffices, is incorrect. What is actually shown by Younes is that the probability is bounded by α *for each independent way* of rejecting a conjunction as false when it is true, provided an α bound on the type I error probability for each conjunct. This is not the same as showing that the probability of rejecting a conjunction *in any way* is bounded by α.

We have now shown how to verify any formula without nested probabilistic operators so that conditions (9) and (10) are satisfied. To verify a negation, we verify the negated formula while reversing the role of the error bounds. A conjunction is verified by verifying each conjunct using the type II error bound intended for the conjunction, but a tighter type I error bound. For probabilistic operators, we can use one of the acceptance sampling tests described in Section 2. Section 7 presents empirical results for our algorithm using two different tests: the sequential version of a single sampling plan and Wald's sequential probability ratio test.

5.2 Model Checking with Nested Probabilistic Operators

This section considers formulae with nested probabilistic operators. If a path formula contains probabilistic operators, we can no longer assume that it can be verified without error. To deal with such verification errors, we need to modify the verification procedure for probabilistic statements. This part of the algorithm applies only to Markov chains.

5.2.1 Probabilistic Operator

We want to use acceptance sampling, as before, to verify a probabilistic statement $\mathcal{P}_{\bowtie\theta}[\varphi]$ according to the semantics for UTSL$_\delta$. With nested probabilistic operators, there is some probability that the verification result for φ is incorrect. We assume the following bounds on the probability of error:

$$\Pr[\sigma, \tau \nvdash \varphi \mid \sigma, \tau \approx_{\top}^{\delta} \varphi] \leq \alpha' \tag{11}$$
$$\Pr[\sigma, \tau \vdash \varphi \mid \sigma, \tau \approx_{\bot}^{\delta} \varphi] \leq \beta' \tag{12}$$

We will show, next, that we can verify $\mathcal{P}_{\bowtie\theta}[\varphi]$ with error bounds α and β by using an acceptance sampling test with strength $\langle \alpha, \beta \rangle$ and probability thresholds $(\theta + \delta(\theta))(1 - \alpha')$ and $1 - (1 - (\theta - \delta(\theta)))(1 - \beta')$.

Let X and Y be two random variables such that, for any sample trajectory σ,

$$Y = 1 \iff \mathcal{M}, \sigma, \tau \vdash \varphi \qquad X = 1 \iff \mathcal{M}, \sigma, \tau \approx_{\top}^{\delta} \varphi$$
$$Y = 0 \iff \mathcal{M}, \sigma, \tau \nvdash \varphi \qquad X = 0 \iff \mathcal{M}, \sigma, \tau \approx_{\bot}^{\delta} \varphi$$

Note that Y has exactly two outcomes and is thus a Bernoulli variate, but X can have more than two outcomes. Let $\Pr[X = 1] = p$, $\Pr[X = 0] = q$, $p_0 = \theta + \delta(\theta)$ and $p_1 = \theta - \delta(\theta)$. It follows from Definition 8 that $\mathcal{M}, \sigma_{\leq\tau} \approx^{\delta}_{\top} \mathcal{P}_{\geq\theta}[\varphi]$ if and only if $p \geq p_0$ and $\mathcal{M}, \sigma_{\leq\tau} \approx^{\delta}_{\perp} \mathcal{P}_{\geq\theta}[\varphi]$ if and only if $q \geq 1 - p_1$. Hence, to verify $\mathcal{P}_{\geq\theta}[\varphi]$, we should test $H_0 : p \geq p_0$ against $H_1 : q \geq 1 - p_1$ (for $\mathcal{P}_{\leq\theta}[\varphi]$, the roles of the hypotheses are simply reversed).

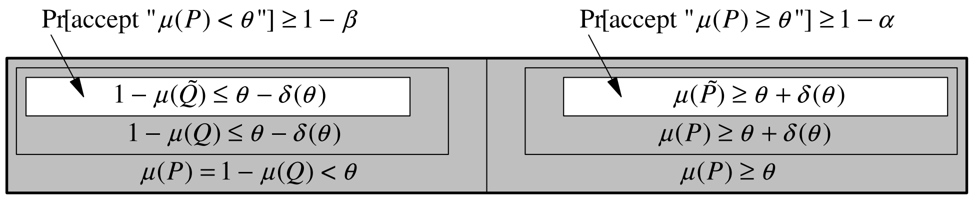
Lemma 11 *Let X and Y be two random variables such that* $\Pr[Y = 0 \mid X = 1] \leq \alpha'$ *and* $\Pr[Y = 1 \mid X = 0] \leq \beta'$. *If* $\Pr[X = 1] = p$ *and* $\Pr[X = 0] = q$, *then* $p(1 - \alpha') \leq \Pr[Y = 1] \leq 1 - q(1 - \beta')$.

Theorem 12 (Acceptance Sampling with Observation Errors) *Let Y be a Bernoulli variate whose observations are related to the observations of a random variable X as follows:* $\Pr[Y = 0 \mid X = 1] \leq \alpha'$ *and* $\Pr[Y = 1 \mid X = 0] \leq \beta'$. *Furthermore, let* $\Pr[X = 1] = p$, $\Pr[X = 0] = q$, *and* $\Pr[Y = 1] = p'$. *An acceptance sampling test with strength $\langle \alpha, \beta \rangle$ for testing $H'_0 : p' \geq p_0(1-\alpha')$ against $H'_1 : p' \leq 1 - (1 - p_1)(1 - \beta')$ has strength at least $\langle \alpha, \beta \rangle$ when used as a test of $H_0 : p \geq p_0$ against $H_1 : q \geq 1 - p_1$, assuming that H_0 is accepted if and only if the test dictates acceptance of H'_0.*

PROOF. From (1), assuming a single sampling plan $\langle n, c \rangle$ is used, we get $F(c; n, p')$ as the probability of accepting H'_1. We know from Lemma 11 that $p' \geq p(1 - \alpha')$. Since $F(c; n, p)$ is a non-increasing function of p in the interval $[0, 1]$, we have $F(c; n, p') \leq F(c; n, p(1 - \alpha'))$, which if $H_0 : p \geq p_0$ holds is at most $F(c; n, p_0(1 - \alpha'))$. By choosing n and c so that $F(c; n, p_0(1 - \alpha')) \leq \alpha$, we ensure that the probability of accepting H'_1, and therefore also H_1, is at most α when H_0 holds.

The probability of accepting H'_0 is $1 - F(c; n, p')$ when using the single sampling plan $\langle n, c \rangle$. It follows from Lemma 11 that $p' \leq 1 - q(1 - \beta')$. Thus, $1 - F(c; n, p') \leq 1 - F(c; n, 1 - q(1 - \beta'))$, which in turn is at most $1 - F(c; n, 1 - (1 - p_1)(1 - \beta'))$ if $H_1 : q \geq 1 - p_1$ holds. Consequently, if we choose n and c so that $1 - F(c; n, 1 - (1 - p_1)(1 - \beta')) \leq \beta$, we are guaranteed that the probability of accepting H'_0, and therefore also H_0, is at most β when H_1 holds. $\square$

The above proof establishes Theorem 12 specifically for single sampling plans. The result is more general, however, because we only need to modify the probability thresholds in order to cope with observation error while leaving the rest of the test intact. We can use the same modification for other acceptance sampling tests, for example Wald's sequential probability ratio test. Note that the probability thresholds equal p_0 and p_1 if the observation error is zero, as should be expected. A procedure for verifying $\mathcal{P}_{\geq\theta}[\varphi]$ with probabilistic operators in φ follows immediately from Theorem 12.



Pr[accept "$\mu(P) < \theta$"] $\geq 1 - \beta$
Pr[accept "$\mu(P) \geq \theta$"] $\geq 1 - \alpha$
$1 - \mu(\tilde{Q}) \leq \theta - \delta(\theta)$
$1 - \mu(Q) \leq \theta - \delta(\theta)$
$\mu(P) = 1 - \mu(Q) < \theta$
$\mu(\tilde{P}) \geq \theta + \delta(\theta)$
$\mu(P) \geq \theta + \delta(\theta)$
$\mu(P) \geq \theta$

Fig. 4. Probabilistic guarantees for model-checking problems with formulae of the form $\mathcal{P}_{\geq\theta}[\varphi]$ and probabilistic operators in φ. The thick box represents all such model-checking problems. In the right half are problems with an affirmative answer. A subset of these problems have an affirmative answer even with an indifference region at the top level of half-width $\delta(\theta)$. For some of the latter set of problems, the formula holds with indifference regions at all levels. It is for this last set of problems that we can guarantee an affirmative answer with probability at least $1 - \alpha$. There is a similar hierarchy for the problems with a negative answer, in the left half of the thick box. The gray area represents the set of problems for which we give no correctness guarantees. The white boxes correspond to satisfaction and unsatisfaction according to the semantics of UTSL$_\delta$.

Corollary 13 *An acceptance sampling test with strength $\langle\alpha, \beta\rangle$ and probability thresholds $(\theta + \delta(\theta))(1 - \alpha')$ and $1 - (1 - (\theta - \delta(\theta)))(1 - \beta')$ can be used to verify $\mathcal{P}_{\geq\theta}[\varphi]$ with type I error probability α and type II error probability β, provided that φ can be verified over trajectories with type I error probability α' and type II error probability β'.*

To better understand the verification procedure for $\mathcal{P}_{\geq\theta}[\varphi]$ with nested probabilistic operators, consider the following sets of trajectories:

$$\begin{array}{ll} P = \{\sigma \in \mathit{Path}(\sigma_{\leq\tau}) \mid \mathcal{M}, \sigma, \tau \models \varphi\} & Q = \{\sigma \in \mathit{Path}(\sigma_{\leq\tau}) \mid \mathcal{M}, \sigma, \tau \not\models \varphi\} \\ \tilde{P} = \{\sigma \in \mathit{Path}(\sigma_{\leq\tau}) \mid \mathcal{M}, \sigma, \tau \mathrel{|\!\!\approx^{\delta}_{\top}} \varphi\} & \tilde{Q} = \{\sigma \in \mathit{Path}(\sigma_{\leq\tau}) \mid \mathcal{M}, \sigma, \tau \mathrel{|\!\!\approx^{\delta}_{\perp}} \varphi\} \end{array}$$

We cannot determine membership in P or Q for a sampled trajectory $\sigma \in \mathit{Path}(\sigma_{\leq\tau})$ if φ contains probabilistic operators. We assume, however, that we have a probabilistic procedure for determining membership in $\tilde{P}$ or $\tilde{Q}$. We require a probability of at most α' that σ is determined to be in $\tilde{Q}$ if it is in $\tilde{P}$, and a probability of at most β' that σ is determined to be in $\tilde{P}$ when it is in $\tilde{Q}$. Given such a procedure, Theorem 12 provides us with a way to test $H_0 : \mu(\tilde{P}) \geq \theta + \delta(\theta)$ against $H_1 : \mu(\tilde{Q}) \geq 1 - (\theta - \delta(\theta))$. Acceptance of H_0 leads to acceptance of $\mathcal{P}_{\geq\theta}[\varphi]$ as true, and acceptance of H_1 leads to rejection of $\mathcal{P}_{\geq\theta}[\varphi]$ as false. We are guaranteed that H_0 is accepted with probability at least $1 - \alpha$ if H_0 holds. Since $\tilde{P} \subset P$, we know that $\mu(P) \geq \theta$ when H_0 holds, so there is a high probability of accepting $\mathcal{P}_{\geq\theta}[\varphi]$ when it holds with some margin. We also know that H_1 is accepted with probability at least $1 - \beta$ if H_1 holds, and $\mu(P) < \theta$ in that case, so there is a high probability of rejecting $\mathcal{P}_{\geq\theta}[\varphi]$ when it is false with some margin. Fig. 4 gives a graphical representation of the correctness guarantees provided by our algorithm.

5.2.2 Path Formulae with Probabilistic Operators

We have established a procedure for verifying probabilistic statements when path formulae are verified with some probability of error. It remains for us to show how to verify path formulae so that conditions (11) and (12) are satisfied. This is straightforward for $X^I\ \Phi$. We simulate a single state transition and verify Φ in the resulting state with error bounds α' and β'.

Path formulae of the form $\Phi\ \mathcal{U}^I\ \Psi$ require more thought. We need to find a $t \in I$ such that Ψ is satisfied at time t and Φ is satisfied at all time points t' prior to t. We assume that the model is a Markov chain so that it suffices to consider the time points at which state transitions occur (cf. Section 4.3). This is guaranteed to be a finite number of time points if $\sup I$ is finite and the model is non-explosive. We can then treat the verification of $\Phi\ \mathcal{U}^I\ \Psi$ as a large disjunction of conjunctions. Let $t_0 = 0$ and let $\{t_1, \ldots, t_n\}$ be the set of time points at which state transitions occur, with $t_i \leq \sup I$. Furthermore, let t_{n+1} be some time point later than $\sup I$. We can verify $\Phi\ \mathcal{U}^I\ \Psi$ as follows:

$$\sigma, \tau \vdash \Phi\ \mathcal{U}^I\ \Psi \qquad \text{if } \bigvee_{i=0}^{n} \Big((t_i \geq \tau) \wedge \big([t_i, t_{i+1}) \cap I \neq \emptyset \big) \wedge (s_i \vdash \Psi) \tag{13}$$

$$\wedge \big((t_i \in I) \vee (s_i \vdash \Phi) \big) \wedge \bigwedge_{j=0}^{i-1} (s_j \vdash \Phi) \Big) \tag{14}$$

Since disjunction can be expressed using conjunction and negation, and we know how to verify negations and conjunctions using statistical techniques, this gives us a way to verify $\Phi\ \mathcal{U}^I\ \Psi$ so that (11) and (12) are satisfied. In general, (13) has $n + 1$ disjuncts, the ith disjunct being a conjunction consisting of $i + 1$ conjuncts. Hence, if Φ and Ψ for disjunct i (corresponding to a trajectory prefix up to time t_i) are verified with type I error probability $\alpha'/(i + 1)$ and type II error probability $\beta'/(n + 1)$, then (11) and (12) are satisfied. If states are repeated along a sample trajectory, then (13) can be made smaller by eliminating the verification of Φ or Ψ in repeated states.

The dependence of the nested error bounds on path length may seem prohibitive, but the sample size of acceptance sampling tests is typically logarithmic in the error bounds. Hence, the sample size for nested probabilistic operators will be logarithmic in the path length. Note that if Φ is probabilistic, but not Ψ, and the time bound is exceeded before Ψ is satisfied, then we do not need to verify Φ in any state along σ. We already know that the path formula does not hold in that case. This can reduce verification effort substantially in practice, as demonstrated in Section 7.

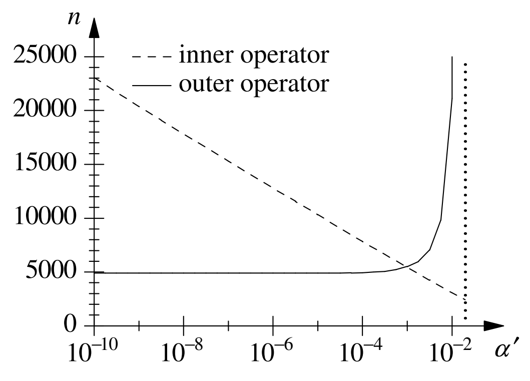


Fig. 5. Verification effort, as a function of the symmetric observation error α', for the probabilistic operators of $\mathcal{P}_{\geq 0.9}\big[X\ \mathcal{P}_{\geq 0.85}[X\ x{=}1]\big]$.

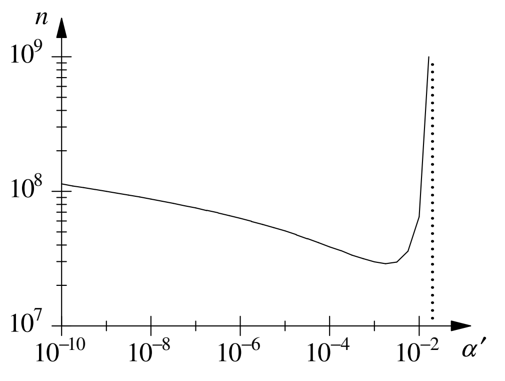


Fig. 6. Total verification effort, as a function of the symmetric observation error α', for $\mathcal{P}_{\geq 0.9}\big[X\ \mathcal{P}_{\geq 0.85}[X\ x{=}1]\big]$.

5.2.3 Efficiency Considerations

A noteworthy consequence of Theorem 12 is that the bounds on the observation error, α' and β', can be chosen independently of the bounds on the probability of a verification error occurring, α and β. We can decrease α' and β' to increase the indifference region of the outer probabilistic statement and therefore lower the sample size required to verify this part of the formula. This will increase the effort required per observation, however, since we have to verify the nested probabilistic statements with higher accuracy. If we increase α' and β' to lower the effort per observation, then we need to make more observations. Clearly, there is a tradeoff here, and the choice for the bounds on the observation error can have a great impact on performance.

Example 14 *Consider the formula* $\Phi = \mathcal{P}_{\geq 0.9}\big[X\ \mathcal{P}_{\geq 0.85}[X\ x{=}1]\big]$, *with* $\delta(\theta)$ *such that* $p_0 = 0.91$ *and* $p_1 = 0.89$ *for the outer probabilistic operator, and* $p'_0 = 0.865$ *and* $p'_1 = 0.835$ *for the inner operator. Furthermore, assume that we want to verify* Φ *with error bounds* $\alpha = \beta = 0.01$. *Assuming symmetric observation error* ($\alpha' = \beta'$) *and using single sampling plans, the verification effort for* Φ *is the product of the samples sizes needed to verify the outer and inner probabilistic operators. Fig. 5 plots the two factors of the total effort separately. The dotted line indicates an upper bound on the symmetric observation error corresponding to a choice of* α' *that makes the width of the inner indifference region zero. The total effort is plotted in Fig. 6. The effort is minimal at* $\alpha' = \beta' \approx 0.00153$ *in this case.*

Ideally, we should use an observation error that minimizes the expected verification effort, but this quantity is non-trivial to compute in general. To find a reasonable observation error, we can use a *heuristic* estimate of the verification effort and numerical function minimization to find an observation error with low estimated effort. Such a heuristic is defined by Younes [66].

In addition to choosing a good value for the observation error, we can use *memoization* [53] to further improve the performance of the statistical model-checking algorithm. This means that when we verify a path formula $\Phi\, \mathcal{U}^I\, \Psi$, with Φ or Ψ being probabilistic statements, then we record the tightest error bounds that have been achieved for Φ and Ψ in each visited state. If the same state occurs multiple times along a sample trajectory, the memoized verification result is used. If tighter error bounds are required for subsequent verification results, then the verification effort is limited to reducing the error bounds. Memoization does not affect the validity of the verification result, because it is based on the logical equivalence $\Phi \wedge \Phi \equiv \Phi$. It is also safe to reuse memoized results across observations. If we ensure that each trajectory is generated independently, each observation will be independent as well.

5.3 Complexity of Statistical Probabilistic Model Checking

The time complexity of statistical probabilistic model checking depends on the number of observations (sample size) required to reach a decision, as well as the time required to generate each observation. An observation involves the verification of a path formula over a sample trajectory. Both the sample size and the time per observation are generally random variables, so we talk about the *expected* complexity of statistical probabilistic model checking.

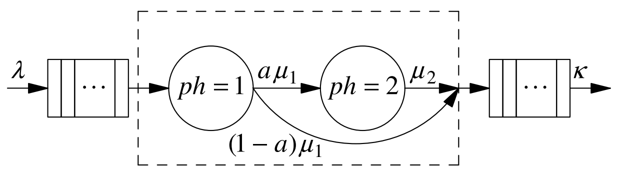
First, consider the time complexity for verifying $\mathcal{P}_{\bowtie\theta}[\varphi]$ without nested probabilistic operators. The first component of the complexity is the time per observation. A sample trajectory σ_i may be infinite, but to verify the path formula $X^I\, \Phi$, we only need to consider a finite prefix of σ_i. The same is true for path formulae of the form $\Phi\, \mathcal{U}^I\, \Psi$ under circumstances discussed above. Without nested probabilistic operators, nested formulae will be classical logic expressions, which we assume can be verified in constant time. Let m be the expected effort to simulate a state transition. The time per observation is proportional to m for $X^I\, \Phi$ and proportional to m times the number of state transitions that occur in a time interval of length $\sup I$ for $\Phi\, \mathcal{U}^I\, \Psi$. Let q denote the expected number of state transitions that occur in a unit-length interval of time. For continuous-time Markov chains, an upper bound for q is the maximum exit rate of any state. The expected time per observation is then $O(m \cdot q \cdot \sup I)$ for $\Phi\, \mathcal{U}^I\, \Psi$. This is an estimate for the worst-case scenario that $\neg\Phi \vee \Psi$ is not satisfied prior to time $\sup I$. If we reach a state satisfying $\neg\Phi \vee \Psi$ long before $\sup I$ time units, then we can determine the truth value of $\Phi\, \mathcal{U}^I\, \Psi$ without considering further states.

The second component of the time complexity for verifying $\mathcal{P}_{\bowtie\theta}[\varphi]$ is the expected sample size, which is a function of α, β, θ, and δ. If we use a sequential test, then the expected sample size also depends on the unknown probability measure p of the set of trajectories that satisfy φ. The expected sample size for various acceptance sampling tests was discussed in Section 2. For example, the sample size for a single sampling plan is approximately proportional to the logarithm of α and β, and inversely proportional to δ^2.

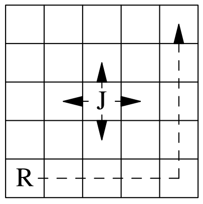
Let N_p denote the expected sample size of the test used to verify probabilistic statements. The verification time for $\mathcal{P}_{\bowtie\theta}[X^I\ \Phi]$ is then $O(N_p \cdot m)$ and for $\mathcal{P}_{\bowtie\theta}[\Phi\ \mathcal{U}^I\ \Psi]$ it is $O(N_p \cdot m \cdot q \cdot \sup I)$. The time complexity of statistical probabilistic model checking, for a single initial state or an initial-state distribution, is independent of the size of the state space for a model if N_p, m, and q are independent of state-space size. We can make N_p completely model independent by using a single sampling plan, in which case N_p depends only on α, β, θ, and δ. The factor m is generally both model and implementation dependent and therefore hard to capture. For generalized semi-Markov processes, for example, m could very well be proportional to the number of events in the model. It can also be state-space dependent, but models often have structure that can be exploited by the simulator to limit such dependence. Finally, q is clearly model dependent, but may be independent of state-space size, as is the case for the symmetric polling system described in Section 6.2.

With nested probabilistic operators, the verification time per state along a sample trajectory is no longer constant. The complexity depends on the level of nesting and the path operators involved. Consider $\mathcal{P}_{\bowtie\theta}\Big[\mathcal{P}_{\bowtie\theta'}[\Phi'\ \mathcal{U}^{I'}\ \Psi']\ \mathcal{U}^I\ \Psi\Big]$ with one level of nesting as an example. On average we need to verify $\mathcal{P}_{\bowtie\theta'}[\Phi'\ \mathcal{U}^{I'}\ \Psi']$ in $q \cdot \sup I$ states for each of the N_p observations required to verify the outer probabilistic operator. The time complexity for verifying $\mathcal{P}_{\bowtie\theta'}[\Phi'\ \mathcal{U}^{I'}\ \Psi']$ is $O(N'_p \cdot m \cdot q \cdot \sup I')$, so the total time complexity is $O(N_p \cdot N'_p \cdot m^2 \cdot q^2 \cdot \sup I \cdot \sup I')$. With memoization, however, the expected time complexity is $O(m \cdot q \cdot (N_p \cdot \sup I + k \cdot N'_p \cdot \sup I'))$, where k is the expected number of unique states visited within $\sup I + \sup I'$ time units from some initial state. The value of k is in the worst case $|S|$, the size of the state space, but can be significantly smaller depending on the dynamics of the model.

The space complexity of statistical probabilistic model checking is generally modest. We need to store the current state of a sample trajectory when generating an observation for the verification of a probabilistic statement, and this typically requires $O(\log|S|)$ space. For systems that do not satisfy the Markov property, we may also need to store additional information to capture the execution history during simulation. In the presence of nesting, we may need to store up to d states simultaneously at any point in time during verification, where d is the maximum depth of a nested probabilistic operator. The nesting depth for Φ is at most $|\Phi|$, so the space requirements are still modest. If we use memoization to speed up the verification of formulae with nested probabilistic operators, the space complexity can be as high as $O(|\Phi| \cdot |S|)$. Memoization,



(a) Tandem queuing network.



(b) Robot grid world.

Fig. 7. (a) A tandem queuing network with a two-phase Coxian distribution governing the routing time between the queues; (b) A grid world with a robot (R) moving along the dashed line and a janitor (J) moving randomly around the grid.

as usual, is a way of trading space efficiency for time efficiency.

6 Case Studies

We present four case studies, taken from the literature on performance evaluation and probabilistic model checking, and selected to stress specific performance characteristics of solution methods for probabilistic model checking.

6.1 Tandem Queuing Network

The first case study is based on a model of a tandem queuing network presented by Hermanns et al. [33]. The network consists of two serially connected queues, each with capacity n. Messages arrive at the first queue, get routed to the second queue, and eventually leave the system from the second queue. The interarrival time for messages at the first queue is exponentially distributed with rate $\lambda = 4n$. The processing time at the second queue is exponentially distributed with rate $\kappa = 4$. Fig. 7(a) shows a schematic view of the model with a Coxian routing-time distribution ($\mu_1 = \mu_2 = 2$ and $a = 0.9$). The size of the state space for this model is $O(n^2)$. We will also use a non-Markovian variation of the model with a lognormal routing-time distribution.

We will verify whether the probability is less than 0.5 that a system starting out with both queues empty becomes full within τ time units. Let $s_i \in \{0, \ldots, n\}$, for $i \in \{1, 2\}$, be the number of messages currently in the ith queue. The UTSL formula $\mathcal{P}_{<0.5}[\Diamond^{[0,\tau]}\ s_1{=}n \wedge s_2{=}n]$ represents the property of interest, and we will verify this formula in the state $s_1 = 0 \wedge s_2 = 0$.

6.2 Symmetric Polling System

The second case study uses the model of an n-station symmetric polling system described by Ibe and Trivedi [39]. Each station has a single-message buffer and the stations are attended by a single server in cyclic order. The server begins by polling station 1. If there is a message in the buffer of station 1, the server starts serving that station. Once station i has been served, or if there is no message in the buffer of station i when it is polled, the server starts polling station $i+1$ (or 1 if $i = n$). The polling and service times are exponentially distributed with rates $\gamma = 200$ and $\mu = 1$, respectively. There is a separate arrival event for each station and the inter-arrival time per station is exponentially distributed with rate $\lambda = 1/n$. The size of the state space for a system with n stations is $O(n \cdot 2^n)$.

We will verify the property that, if station 1 is full, then it is polled within τ time units with probability at least 0.5. We do so for different values of n and τ in the state where station 1 has just been polled and the buffers of all stations are full. Let $s \in \{1, \ldots, n\}$ be the station currently receiving the server's attention, let $a \in \{0, 1\}$ represent the activity of the server (0 for polling and 1 for serving), and let $m_i \in \{0, 1\}$ be the number of messages in the buffer of station i. The property of interest is represented in UTSL as $m_1{=}1 \rightarrow \mathcal{P}_{\geq 0.5}[\Diamond^{[0,\tau]}\ \mathit{poll}_1]$, where $\mathit{poll}_1 \equiv s{=}1 \wedge a{=}0$, and the state in which we verify the formula is given by $s{=}1 \wedge a{=}1 \wedge m_1{=}1 \wedge \cdots \wedge m_n{=}1$.

6.3 Robot Grid World

The third case study involves a robot navigating in a grid world, and was introduced by Younes et al. [69] to illustrate the verification of formulae with nested probabilistic operators. A robot is moving in an $n \times n$ grid world from the bottom left corner to the top right corner, while a janitor moves randomly around the grid. The robot first moves along the bottom edge and then along the right edge. Fig. 7(b) provides a schematic view of a grid world with $n = 5$.

The objective is for the robot to reach the top right corner within τ_1 time units with probability at least 0.9, while maintaining at least a 0.5 probability of periodically communicating with a base station. The robot moves at rate $\lambda_R = 1$, unless the janitor occupies the destination square, in which case the robot remains stationary. The janitor moves around randomly in the grid at rate $\lambda_J = 2$, selecting the destination from the set of neighboring squares with equal probability. The robot initiates communication with the base station at rate $\mu = 1/10$, and the duration of each communication session is exponentially distributed with rate $\kappa = 1/2$. Let c be a Boolean state variable that is true when the robot is communicating, and let x and y represent the current location of the robot. The UTSL formula $\mathcal{P}_{\geq 0.9}\left[\mathcal{P}_{\geq 0.5}[\Diamond^{[0,\tau_2]}\ c]\ \mathcal{U}^{[0,\tau_1]}\ x{=}n \wedge y{=}n\right]$ expresses the desired objective for this case study. The robot moves along a line only, so the size of the state space for the robot grid world is $O(n^3)$.

6.4 Dependable Workstation Cluster

The final case study is a dependable cluster of workstations due to Haverkort et al. [31]. The system consists of two sub-clusters, each containing n workstations. Communication between the two sub-clusters is performed over a backbone connection. The workstations of each sub-cluster are connected in a star topology, with a single switch providing connectivity to the backbone. Each of the components can fail at any time, and the time to failure is exponentially distributed with different rates for different components. There is a single repair unit that can restore failed units. The repair time is assumed to be exponentially distributed. The size of the state space is $O(n^2)$.

We will also use a Weibull distribution as the failure-time distribution for workstations to get a non-Markovian model. Note that if the failure time for a workstation is exponentially distributed with rate λ, then the time to a single failure in a sub-cluster with k operational workstations is exponentially distributed with rate $k \cdot \lambda$. We can hence represent failure of any workstation by a single event with a state-dependent rate. If the failure-time distribution is non-exponential, however, we need a separate event for each workstation.

The minimum quality of service (QoS) for a cluster is defined as having at least three interconnected operational workstations. Let w_{l} (w_{r}) denote the number of operational workstations in the left (right) sub-cluster. Furthermore, let b represent the atomic proposition that the backbone is working, and s_{l} (s_{r}) that the left (right) switch is up. Minimum QoS can then be defined as $\mathit{minimum} \equiv (w_{\mathrm{l}}{\geq}3 \wedge s_{\mathrm{l}}) \vee (w_{\mathrm{r}}{\geq}3 \wedge s_{\mathrm{r}}) \vee (w_{\mathrm{l}}{+}w_{\mathrm{r}}{\geq}3 \wedge b \wedge s_{\mathrm{l}} \wedge s_{\mathrm{r}})$. The property we will verify is $\mathcal{P}_{<0.1}[\Diamond^{\tau}\ \neg \mathit{minimum}]$ and we do so in the state where all units are functional.

7 Empirical Evaluation of Probabilistic Model Checking

This section explores empirical performance characteristics of statistical probabilistic model checking using the case studies introduced in Section 6. The results have been generated on a 3-GHz Pentium 4 PC running Linux.

7.1 Sample Size and Trajectory Length

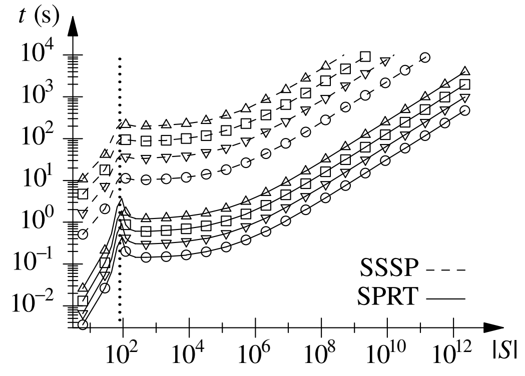
As discussed in Section 5.3, two main factors influencing the verification time for the statistical approach are sample size and trajectory length.

The sample size depends on the sampling plan that we choose to use, the error bounds α and β that we want to guarantee, the threshold θ, and the choice of $\delta(\theta)$ determining the half-width of an indifference region centered around θ. We consider two different sampling plans described in Section 2: the sequential version of a single sampling plan (SSSP) and Wald's sequential probability ratio test (SPRT). For these sampling plans, the sample size is a random variable whose expectation also varies with p, which in our case is the probability measure of a set of trajectories satisfying a path formula.

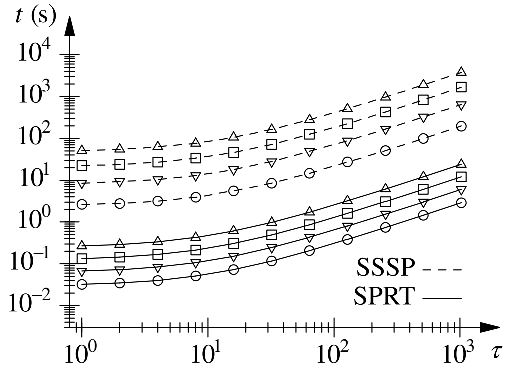
Figs. 8 and 9 present data for the tandem queuing network and symmetric polling system case studies, respectively. In each case, we show verification time for the SSSP and the SPRT using four different test strengths (subfigures (a) and (b)). We also give details of both sample size (subfigures (c) and (d)) and trajectory length (subfigures (e) and (f)). For all data, we plot the results against model size (subfigures (a), (c), and (e)) and the time bound of the path formula (subfigures (b), (d), and (f)). Each data point is an average over 20 runs. We used $\delta(\theta) = 5 \cdot 10^{-3}$ as the half-width of the indifference region. Furthermore, we used a symmetric test strength ($\alpha = \beta$) across the board.

Our data shows that the SPRT outperforms the SSSP almost exclusively by a wide margin. A clear exception is seen in Fig. 9(d). The SPRT has a larger expected sample size than the SSSP for $\alpha = \beta$ equal to 10^{-4} and 10^{-8} close to where the truth value of the UTSL formula changes (indicated by the dotted line). Fig. 10 zooms in on the relevant region to show this more clearly. The gray area indicates the range of τ for which the probability measure, p, of the set of trajectories satisfying the path formula $\Diamond^{[0,\tau]}\, poll_1$ is in the indifference region $(\theta - \delta, \theta + \delta)$. There is a sharp increase in the expected sample size for the SPRT in and near the indifference region, while the expected sample size for the SSSP remains largely unchanged. Note, however, that neither test gives any valuable accuracy guarantees in the indifference region. If we have reason to believe that p is very close to θ, and we really want to know on which side p is of the threshold, then we may want to resort to numerical solution techniques. The alternative is to narrow the indifference region. Fig. 11 shows how the expected sample size for the two sampling plans depends on the half-width of the indifference region. The plot is for the symmetric polling system with $\theta = 0.5$, $n = 10$, and $\tau = 10$. It is generally more costly to narrow the indifference region when using the SSSP rather than the SPRT.

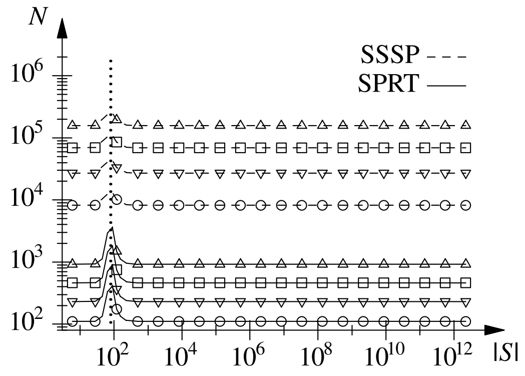
The expected length of trajectories varies with the model and the path for-



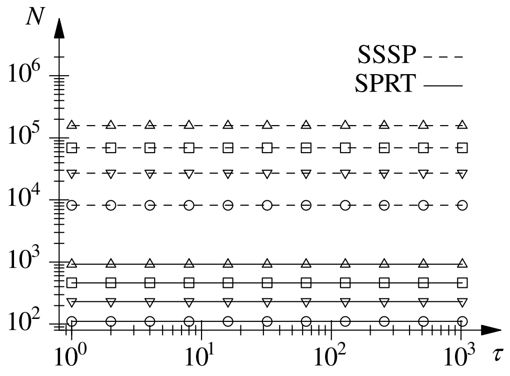
(a) Verification time as a function of state space size.



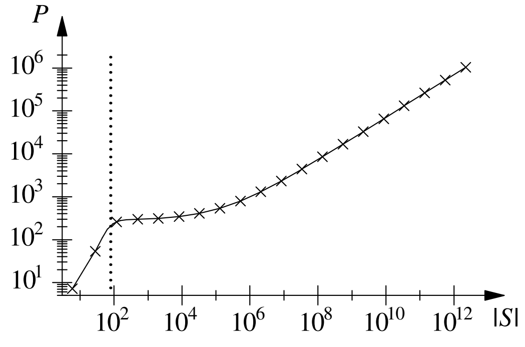
(b) Verification time as a function of time bound.



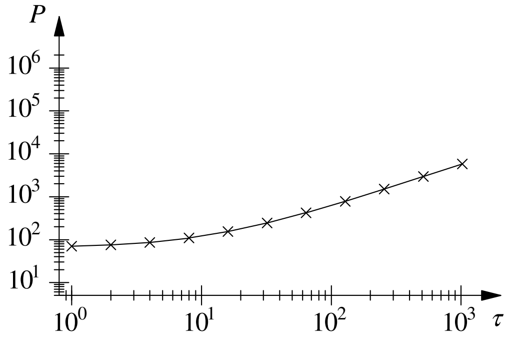
(c) Sample size as a function of state space size.



(d) Sample size as a function of time bound.

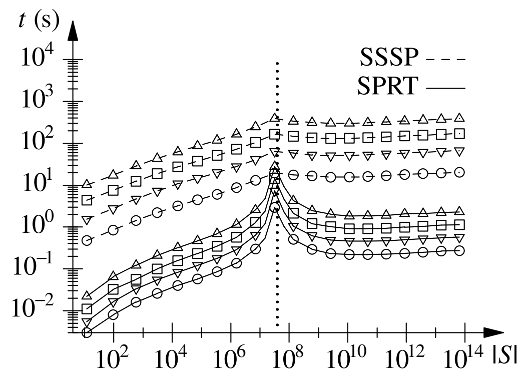


(e) Trajectory length as a function of state space size.

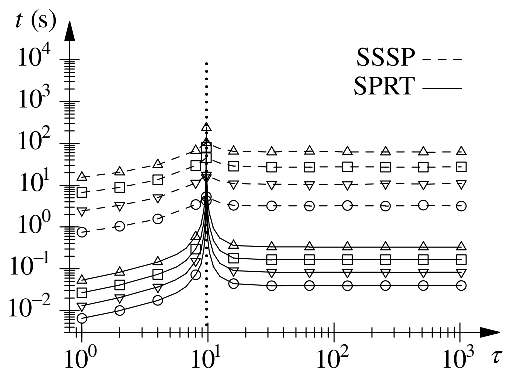


(f) Trajectory length as a function of time bound.

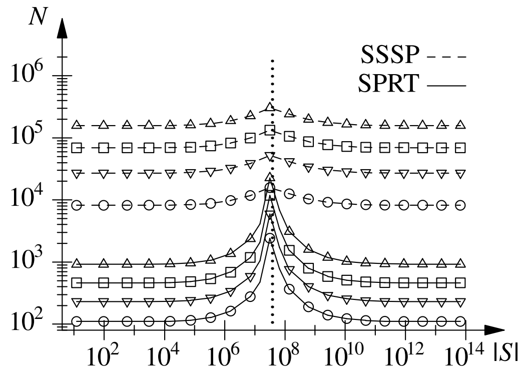
Fig. 8. Empirical results for the tandem queuing network ($\theta = 0.5$), with $\tau = 50$ (left) and $n = 63$ (right), using acceptance sampling with $2\delta = 10^{-2}$ and symmetric error bounds $\alpha = \beta$ equal to 10^{-8} ($\triangle$), 10^{-4} ($\square$), 10^{-2} ($\triangledown$), and 10^{-1} ($\circ$). The average trajectory length is the same for all values of α and β. The dotted lines mark a change in the truth value of the formula being verified.



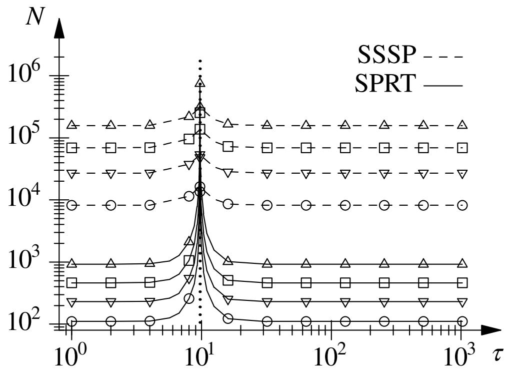
(a) Verification time as a function of state space size.



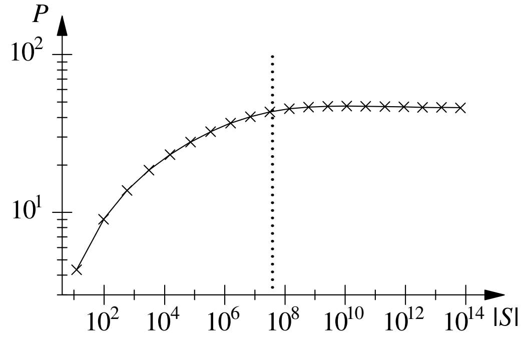
(b) Verification time as a function of time bound.



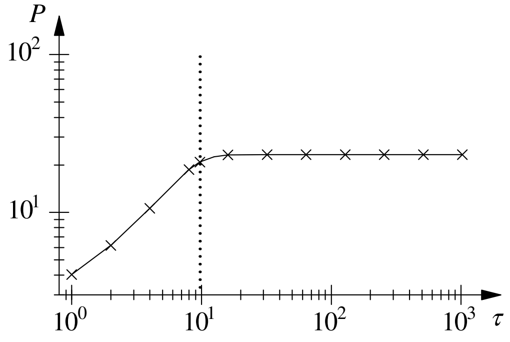
(c) Sample size as a function of state space size.



(d) Sample size as a function of time bound.



(e) Trajectory length as a function of state space size.



(f) Trajectory length as a function of time bound.

Fig. 9. Empirical results for the symmetric polling system ($\theta = 0.5$), with $\tau = 20$ (left) and $n = 10$ (right), using acceptance sampling with $2\delta = 10^{-2}$ and symmetric error bounds $\alpha = \beta$ equal to 10^{-8} ($\triangle$), 10^{-4} ($\square$), 10^{-2} ($\triangledown$), and 10^{-1} ($\circ$). The average trajectory length is the same for all values of α and β. The dotted lines mark a change in the truth value of the formula being verified.

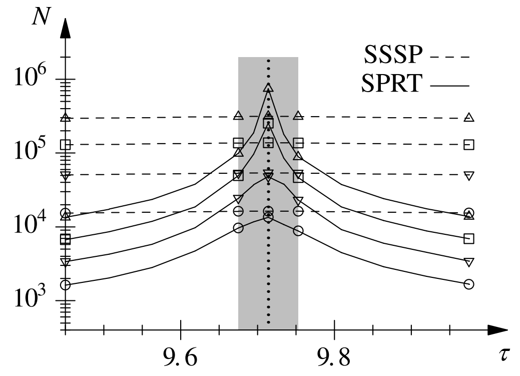


Fig. 10. Sample size as a function of the formula time bound for the symmetric polling system ($\theta = 0.5$, $n = 10$) near the indifference region (shaded area), with $2\delta = 10^{-2}$ and $\alpha = \beta$ equal to 10^{-8} ($\triangle$), 10^{-4} ($\square$), 10^{-2} ($\triangledown$), and 10^{-1} ($\circ$).

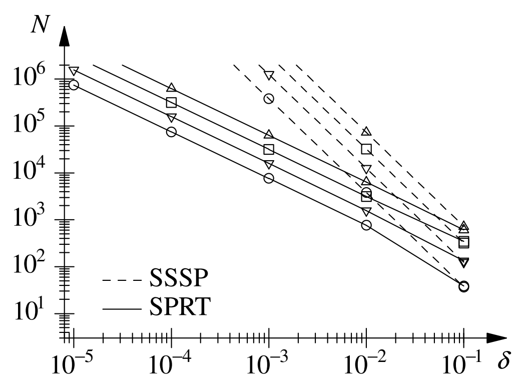


Fig. 11. Sample size as a function of the half-width of the indifference region for the symmetric polling system ($\theta = 0.5$, $n = 10$, $\tau = 10$), with $\alpha = \beta$ equal to 10^{-8} ($\triangle$), 10^{-4} ($\square$), 10^{-2} ($\triangledown$), and 10^{-1} ($\circ$).

mula. If we are lucky, we can verify a time-bounded path formula over a sample trajectory by considering only a short prefix that ends long before the time bound is exceeded. We can see this phenomenon in Fig. 9(f) for the path formula $\Diamond^{[0,\tau]}$ $poll_1$. As τ increases so does the probability of achieving $poll_1$ in the interval $[0, \tau]$. The average trajectory length approaches a constant as τ increases because the average number of state transitions required to achieve $poll_1$ is independent of τ. When the truth value of a path formula cannot be determined before the time bound is reach, then the average trajectory length grows linearly with τ, as is seen, for example, in Fig. 8(f).

The expected trajectory length over a fixed time interval depends on the distributions that govern the trigger time of events. If the distribution parameters, in particular the mean, depend on the model size, then so will the expected trajectory length. The average trajectory length for the tandem queuing network increases linearly with the capacity, n, of the queues because the arrival rate for messages is $4n$. Note, however, that the size of the state space is $O(n^2)$ for the tandem queuing network, so the average trajectory length is proportional to the square root of $|S|$ (Fig. 8(e)). In contrast, the rates for the symmetric polling system are independent of the size of the state space. Initially, the average trajectory length increases with the size of the state space (Fig. 9(e)) because it takes longer time to achieve $poll_1$ with more polling stations. As the state space increases further, the probability of achieving $poll_1$ in the interval $[0, \tau]$ goes to zero, and all sample trajectories end with the time bound τ being exceeded. The expected number of state transitions occurring in the interval $[0, \tau]$ is the same for all state space sizes, since the exit rates are constant, so the verification time does not increase for larger state spaces.

7.2 "Five Nines"

For safety critical systems, we want to ensure that the probability of failure is very close to zero. While guaranteeing a zero probability of failure is usually unrealistic, it is not uncommon to require the failure probability of a safety critical system to be at most 10^{-4} or 10^{-5}. A failure probability of at most 10^{-5} means a success probability of at least $1 - 10^{-5} = 0.99999$, commonly referred to as "five nines." For such high accuracy requirements, it is typically best to use numerical solution techniques, but if the model is non-Markovian or has a large state space, this may not be an option.

To use statistical hypothesis testing with a probability threshold $1 - 10^{-5}$, we need an indifference region with half-width at most 10^{-5}. An indifference region that narrow requires a large average sample size if the success probability is close to one, as we would expect it to be for a good system design. A possible alternative is to set the indifference region to $(1 - 10^{-5}, 1)$ and use a curtailed single sampling plan. The advantage of a curtailed single sampling plan is that it has a fixed upper bound on the sample size: $n = \lceil \log \beta / \log(1 - 10^{-5}) \rceil$, where β is the maximum probability that we accept the system as safe if the success probability is at most $1 - 10^{-5}$. We accept the system as safe if all n observations are positive, but reject the system as unsafe at the first negative observation. This means that if the success probability for the system is far below acceptable, we will quickly reject the system, while acceptance always requires n observations. Note, however, that we have no control over the probability of rejecting an acceptable system design, except that we will always accept a system that has success probability one. If we require finer control over the risk of rejecting a system with success probability greater than $1 - 10^{-5}$, then a curtailed single sampling plan is not a viable option.

Fig. 12 plots the average verification time, as a function of the formula time bound, for the symmetric polling system ($n = 10$) with indifference regions $(0.99999, 1)$ and $(0.999985, 0.999995)$, of which the former leads us to use a curtailed single sampling plan. In the latter case (solid curves), the SPRT was used. We can see that for low values of τ (the time bound of the property being verified), the average verification time is negligible for both choices of indifference region. As τ increases and the success probability approaches $1 - 10^{-5}$, the average sample size increases. For the curtailed single sampling plan, as we pass the point at which the success probability exceeds $1 - 10^{-5}$ (roughly at $\tau = 29.57$), the sample size settles at around $2 \cdot 10^6$ for $\beta = 10^{-8}$. The verification time at this point is just under 11 minutes on our test machine (the average trajectory length is just over 23). For the SPRT, we can see clear peaks in the verification time where the probability is close to $1 - 10^{-5}$. The price for moving the upper bound of the indifference region away from 1 is that verification can take over an hour on average instead of a few minutes.

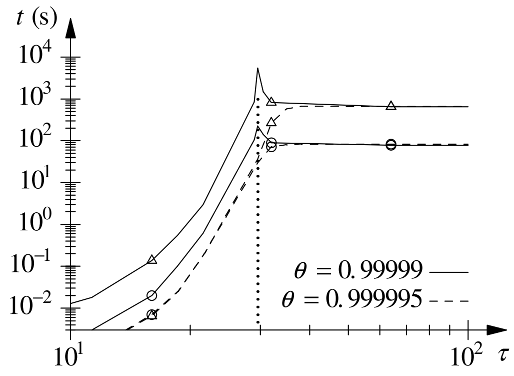


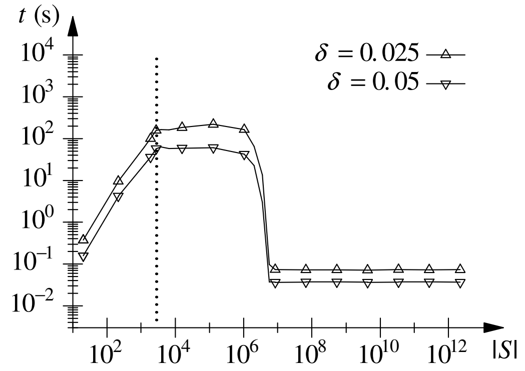
Fig. 12. Verification time as a function of the formula time bound for the symmetric polling system ($n = 10$), with $2\delta = 10^{-5}$ and $\alpha = \beta$ equal to 10^{-8} ($\triangle$) and 10^{-1} ($\circ$).

One of the 20 experiments for $\alpha = \beta = 10^{-8}$ required a sample size of over 35 million, which can be compared to a maximum sample size of just over 1.8 million for the curtailed single sampling plan with $\beta = 10^{-8}$.

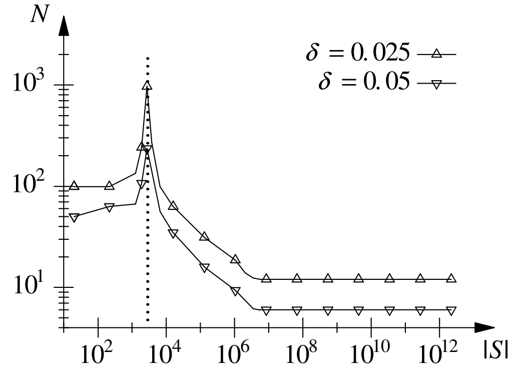
7.3 Nested Probabilistic Operators

We use the robot grid world case study to show results for verification with nested probabilistic operators. We have proven that a statistical approach is possible even in the presence of nested probabilistic operators, with Theorem 12 being the key theoretical result. A practical concern, however, is that such verification could be costly, since each observation for the outer probabilistic operator involves an acceptance sampling test for the inner probabilistic operators. Nevertheless, our empirical results suggest that a statistical approach is, in fact, tractable, provided that memoization is used.

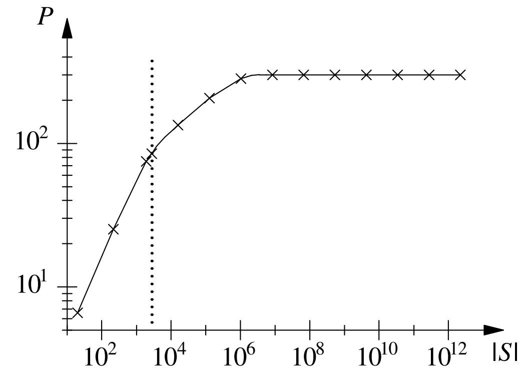
Fig. 13 shows empirical data for the robot grid world case study for verifying the UTSL formula $\mathcal{P}_{\geq 0.9}\Big[\mathcal{P}_{\geq 0.5}[\Diamond^{[0,\tau_2]}\ c]\ \mathcal{U}^{[0,\tau_1]}\ x{=}n \wedge y{=}n\Big]$. This formula asserts that the probability is high that the robot reaches the goal position while periodically communicating with a base station. The time bounds τ_1 and τ_2 were set to 100 and 9, respectively. We used the SPRT, exclusively, and the heuristic proposed by Younes [66] to select the nested error bounds. With $\tau_2 = 9$, the probability measure of the set of trajectories satisfying $\Diamond^{[0,\tau_2]}\ c$ is $1-\mathrm{e}^{-0.9} \approx 0.593$, independent of the start state. We used an indifference region with half-width δ independent of θ. For both values of δ that we used, $\delta = 0.05$ and $\delta = 0.025$, 0.593 is more than a δ-distance from the threshold 0.5 for the inner probabilistic operator, so we will have a low probability of erroneously verifying the path formula $(\mathcal{P}_{\geq 0.5}[\Diamond^{[0,\tau_2]}\ c]\ \mathcal{U}^{[0,\tau_1]}\ x{=}n \wedge y{=}n)$ over sample trajectories. For the outer probabilistic operator, we used the symmetric error bounds $\alpha = \beta = 10^{-2}$. The heuristic gave us the symmetric nested error bounds 0.0153 and 0.00762 for $\delta = 0.05$ and $\delta = 0.025$, respectively.



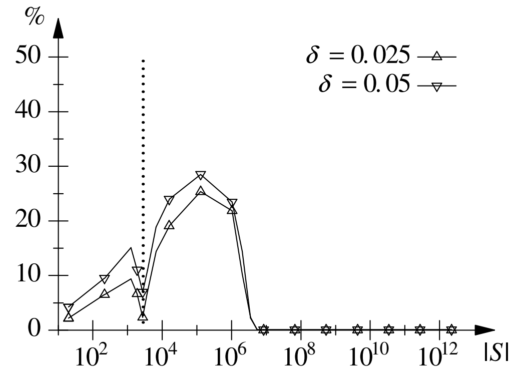
(a) Verification time as a function of state space size.



(b) Sample size as a function of state space size.



(c) Trajectory length as a function of state space size.



(d) Fraction of visited states with nested verification.

Fig. 13. Empirical results for the robot grid world ($\tau_1 = 100$ and $\tau_2 = 9$), using acceptance sampling with symmetric error bounds $\alpha = \beta = 10^{-2}$. The average trajectory length is the same for all values of δ. The dotted lines mark a change in the truth value of the formula being verified.

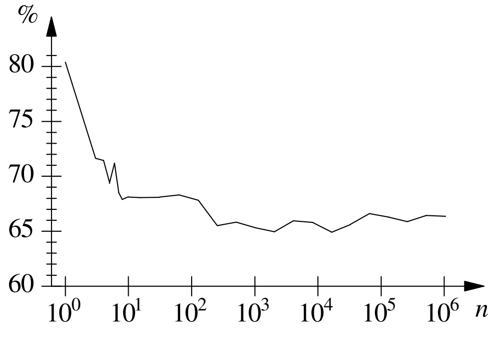
We can see in Fig. 13(b) the familiar peaks in the average sample size where the value of the UTSL formula goes from true to false. Note, however, that the peaks are not present in Fig. 13(a), where the verification time is plotted as a function of the state space size. This is due to memoization. Fig. 13(d) shows the fraction of states in which the inner probabilistic statement is verified, among all states visited along sample trajectories for the outer probabilistic operator. This graph is almost the mirror image of that for the average sample size. As we generate more sample trajectories, we visit the same states more often. With memoization, we verify nested probabilistic statements at most once per unique visited state, so the cost per observation drops over time. The net effect is that total verification time is notably reduced. The price we pay for improved speed is increased memory usage, but the number of unique visited states is only a tiny fraction of the total number of states for the robot grid world. For large state spaces, in this particular case, the time bound is reached before $x{=}n \wedge y{=}n$ is satisfied. When this happens, the inner probabilistic statement does not have to be verified in any state, which dramatically reduces the verification time.

7.4 Simulation Effort and non-Markovian Models

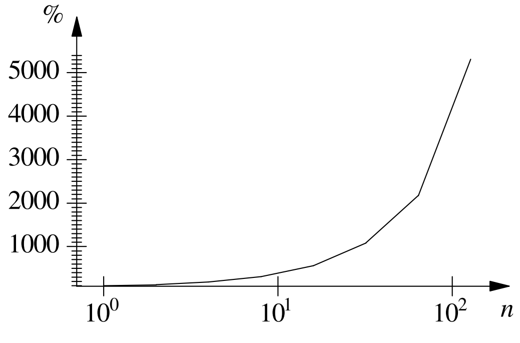
Simulation effort is a major performance factor that can vary greatly from one model to another. For our experiments, we have used a general-purpose discrete-event simulator. This simulator works with an event-based representation of a discrete-event system based on the PRISM input language [54] (with extensions for non-exponential distributions described by Younes [66]). The simulation effort per state transition for this simulator is $O(|E|)$, where E is the set of events specifying the model dynamics. For the tandem queuing network and robot grid world models, the number of events is constant for all model sizes, so simulation effort is a constant factor. For the symmetric polling system, simulation effort is $O(n)$, where n is the number of polling stations. Note, however, that the size of the state space is $O(n \cdot 2^n)$ in this case, so simulation effort grows very slowly as a function of state space size.

The results presented so far have been for Markov chains, but the statistical solution method works equally well for non-Markovian models (with the exception of nested probabilistic properties). Any difference in performance between Markov chains and non-Markovian models will mainly be due to simulation effort. For the simulator we use, non-Markovian models are no harder to simulate, *per se*, than Markov chain. The simulation effort depends on E. If we replace the Coxian routing-time distribution for the tandem queuing network with a lognormal distribution having matching first two moments, for example, then we reduce the number of events by two. Fig. 14(a) shows that this leads to reduced simulation effort. In contrast, if we want to have Weibull-distributed lifetimes for workstations in the workstation cluster model, then we need to have one failure event per workstation instead of a single failure event per cluster. The result is a substantial increase in the simulation effort for the non-Markovian model as a function of cluster size (Fig. 14(b)).

Significant research effort has been devoted to the subject of efficient simulation of Markov chains (e.g., by Hordijk et al. [36]) and discrete-event systems (e.g., by McCormack and Sargent [52]; see, also, [12]). The results in this section are specific to the simulator we have used and are only meant to illustrate how the time complexity of statistical probabilistic model checking depends on simulation effort. The simulation effort could, most certainly, be reduced for some models by using a more efficient simulator, but finding the best simulator for a specific model is well beyond the scope of this paper.



(a) Tandem queuing network.



(b) Dependable workstation cluster.

Fig. 14. Simulation effort per state transition for non-Markovian model relative to simulation effort per state transition for Markov chain model.

8 Related Work

This section discusses some previous research on probabilistic model checking and relates other solution methods to the approach presented in this paper.

8.1 Statistical Solution Methods

The solution method presented in this paper is not the only statistical approach to probabilistic model checking. Lassaigne and Peyronnet [49] propose a statistical approach for model checking a fragment of LTL. They do not formulate it as a hypothesis-testing problem, but instead rely on less efficient techniques for statistical estimation. Grosu and Smolka [24] present an algorithm for classical (non-probabilistic) LTL model checking based on statistical hypothesis testing. They draw samples from the space of trajectories that end in a cycle—called lassos—and need to determine whether the probability measure of lassos involving accepting states is non-zero. Since the model is not assumed to be probabilistic, the probability measure over sets of trajectories can be chosen arbitrarily, and Grosu and Smolka discuss the merits of different choices. To test for non-zero probability, a curtailed single sampling plan is used for optimal performance. This is the same solution method that we would use to verify $\mathcal{P}_{>0}[\varphi]$, but we would have a different sample space and a probability measure over sets of trajectories that is fixed by the model.

Younes [68] modifies the statistical solution method we have presented in this paper so that the probability of false negatives and false positives is bounded by the parameters α and β even *inside* of the indifference region. This feat is accomplished by permitting *undecided* results. The probability of an undecided result is bounded by a third parameter, γ, outside of the indifference region, but is unbounded inside of the indifference region.

Sen et al. [60] explore the idea of probabilistic verification for "black-box" systems. A system is considered to be a black box if we lack a model of the system dynamics. This precludes generation of sample trajectories through discrete-event simulation and instead the analysis needs to be based on trajectories observed during actual execution of the system. Sen et al. present an algorithm for analyzing execution trajectories that amounts to statistical hypothesis testing with fixed-size samples. Their approach does not permit the user to bound the probability of type I and type II errors. Instead, a measure of confidence—the p-value [35, pp. 255–256]—is computed for a verification result. This is reasonable if sample trajectories cannot be generated on demand. There is a hidden assumption in their own evaluation of their algorithm, however, that the black box has a "reset button." This permits the generation of trajectories at will from an initial state. They guide their choice of sample size by a desire to achieve a certain p-value, which is a function of *a specific observation* rather than a general property of an acceptance sampling test as is the case with the type I and type II error bounds. All their empirical evaluation really proves is that a smaller sample size results in shorter verification time—which should surprise no one—but the casual reader may be misled into believing that Sen et al. have devised a more efficient statistical solution method than the one originally proposed by Younes and Simmons [71] and further developed here. The fact is that if the black-box system has a reset button, then our solution method is still applicable and it has the advantage of allowing a user to control the probability of an erroneous result. Sen et al.'s algorithm permits no control over the probability of error and, perhaps worse, provides no reliable procedure for finding the appropriate sample size to achieve a certain p-value—the sample sizes reported in the paper were selected *manually* by the authors (K. Sen, personal communication, May 20, 2004). Younes [65] offers a more thorough analysis of Sen et al.'s algorithm for black-box verification and provides several important improvements when it is assumed that a fixed set of execution trajectories are provided and additional trajectories cannot be generated on demand from an initial state.

The focus of this paper has been on time-bounded properties. We have briefly noted that our approach could handle unbounded properties, such as $\mathcal{P}_{\bowtie\theta}[\Phi\ \mathcal{U}\ \Psi]$. To be ensured of termination, however, we would need to know that the system is such that every trajectory eventually reaches an absorbing state or a state satisfying $\neg\Phi \vee \Psi$. This knowledge could be hard to obtain without costly reachability analysis. Sen et al. [61] attempt to devise a purely statistical approach for verifying unbounded properties that does not rely on any specific knowledge about the system other than the assumption that the state space, S, is finite. Their idea is to generate trajectories using biased sampling. They introduce a stopping probability, p_s, which is the probability of terminating the generation of a trajectory after each state transition. Let p be the probability measure of the set of trajectories that satisfy $\Phi \,\mathcal{U}\, \Psi$ and let p' be the corresponding probability measure when there is a p_s stopping probability in each state. The validity of their algorithm relies on the condition $p' \geq p(1-p_s)^{|S|}$, but this condition holds only for cycle-free models. In general, there is no lower bound for the fraction p'/p that can be expressed only in terms of p_s and $|S|$. Even if the required condition could be shown to hold in general, the accuracy of the verification result would depend on $|S|$, making the approach impractical for anything but models with small state spaces.

8.2 Numerical Solution Methods

To verify the formula $\mathcal{P}_{\bowtie\theta}[\Phi \,\mathcal{U}^{[0,\tau]}\, \Psi]$ for some initial-state distribution μ_0 and model $\mathcal{M}$ with state space S, we can compute the probability

$$p = \int \mu(\{\sigma \in \mathit{Path}(\{\langle s, 0\rangle\}) \mid \mathcal{M}, \sigma, 0 \models \Phi \,\mathcal{U}^{[0,\tau]}\, \Psi\}) \,\mathrm{d}\mu_0(S)$$

numerically and test if $p \bowtie \theta$ holds. Such numerical computation is primarily feasible when $\mathcal{M}$ is a finite-state Markov chain. First, as initially proposed by Baier et al. [8], the problem is reduced to the computation of transient probabilities on a modified Markov chain $\mathcal{M}'$, where all states in $\mathcal{M}$ satisfying $\neg\Phi \vee \Psi$ have been made absorbing. The probability p is equal to the probability that we are in a state satisfying Ψ at time τ in model $\mathcal{M}'$. This probability can be computed using a technique called *uniformization* (also know as *randomization*), originally proposed by Jensen [41]. Let $\mathbf{Q}$ be the generator matrix of $\mathcal{M}'$, $q = \max_i -q_{ii}$, and $\mathbf{P} = \mathbf{I} + \mathbf{Q}/q$. Then p can be expressed as follows:

$$p = \vec{\mu}_0 \cdot \sum_{k=0}^{\infty} \mathrm{e}^{-q\cdot\tau} \frac{(q\cdot\tau)^k}{k!} \mathbf{P}^k \cdot \vec{\chi}_\Psi \tag{15}$$

Here, $\vec{\mu}_0 = [\mu_0(s)]$ and $\vec{\chi}_\Psi$ is a 0-1 column vector, with a 1 in each row corresponding to a state that satisfies Ψ. In practice, the infinite summation is truncated by using the techniques of Fox and Glynn [21], so that the truncation error is bounded by an *a priori* error tolerance ϵ. The number of iterations is $R_\epsilon = q\cdot\tau + c\sqrt{2q\cdot\tau} + 3/2$, where c is $o\Big(\sqrt{\log(1/\epsilon)}\Big)$. This means that the number of iterations grows very slowly as ϵ decreases. For large values of $q\cdot\tau$, the number of iterations is essentially $O(q\cdot\tau)$. The kth term of p is computed iteratively as $p_k = \vec{\mu}_k \cdot \vec{\chi}_\Psi \cdot \mathrm{e}^{-q\cdot\tau}(q\cdot\tau)^k/k!$, where $\vec{\mu}_k = \vec{\mu}_{k-1}\mathbf{P}$ for $k > 0$.

Each iteration involves a matrix-vector multiplication. Let a be the maximum number of non-zero elements in any row of $\mathbf{P}$ and let b_k be the number of non-zero elements of $\vec{\mu}_k$. Then the kth term requires $O(a\cdot b_k)$ operations. In the worst case, both a and b_k are $O(|S|)$, making the total complexity of the numerical solution method $O(q\cdot\tau\cdot|S|^2)$. Often, however, $\mathbf{P}$ is sparse and a is a constant, making the complexity $O(q \cdot \tau \cdot |S|)$. If b_k is constant for all $k \leq R_\epsilon$, then the complexity can be as low as $O(q \cdot \tau)$, i.e. independent of the size of the state space. Typically, though, b_k becomes $O(|S|)$ after only a few iterations, even if $b_1 = 1$. This is, for example, the case for all four models described in Section 6. Hence, the time complexity of the numerical solution method is typically $O(q \cdot \tau \cdot M)$, where M is proportional to $|S|$, *even if we want to verify a formula in only a single state*. This is in comparison to the theoretical time complexity $O(q \cdot \tau \cdot m \cdot N_p)$ for our statistical solution method for verifying a formula in a single state, where m is the simulation effort per state transition and N_p is the expected sample size as a function of p. The product $m \cdot N_p$, is often significantly smaller than $|S|$, and is in some cases independent of $|S|$ even when the numerical solution method has time complexity $O(q \cdot \tau \cdot |S|)$.

The number of iterations required by the numerical solution method can, in some cases, be reduced significantly through the use of steady-state detection [56,50,69,43]. This can give the numerical solution method a clear edge over the statistical approach for large values of τ. It should be noted, however, that true steady-state detection is generally not possible. In practice, this optimization is based on a comparison of successive iteration vectors, which could give unreliable results if convergence is slow, although recent work by Katoen and Zapreev [43] seems to address this problem. Further reduction is possible with the sequential stopping rule described by Younes et al. [69], but this does not reduce the asymptotic time complexity of the numerical method.

The presence of a truncation error, ϵ, means that no definite answer can be given if p is within an ϵ distance of θ. Let $\tilde{p}$ be the computed probability. By accepting a probabilistic formula as true if and only if $\tilde{p} + \epsilon/2 \bowtie \theta$, the numerical approach can be interpreted as solving a UTSL$_\delta$ model-checking problem with $\delta(\theta) = \epsilon/2$ and $\alpha = \beta = 0$. Hence, neither the numerical nor the statistical solution method can solve true UTSL model-checking problems, but it is costlier for the statistical method to narrow the indifference region. Also, the statistical method gives only probabilistic correctness guarantees. The numerical approach is a deterministic algorithm for UTSL$_\delta$ model checking and the statistical approach is a randomized algorithm.

A limiting factor for numerical solution methods is memory. The space complexity for verifying $\mathcal{P}_{\bowtie\theta}\left[\Phi \, \mathcal{U}^{[0,\tau]} \; \Psi\right]$ is $O(|S|)$ in most cases. Various different data structures can be used for numerical computation. Katoen et al. [42] suggest the use of MTBDDs [17,7,22] for CSL model checking, while Baier et al. [8] express a preference for sparse matrices. Parker [54] has developed a hybrid approach, which uses flat representations of vectors and MTBDDs for matrices. With steady-state detection enabled, this hybrid approach requires storage of three double precision floating point vectors of size $|S|$, which for a memory limit of 800 MB means that systems with at most 35 million states can be analyzed. The asymptotic space complexity is the same for the different representations, and sparse matrices nearly always provide faster numerical computation, but symbolic representations of rate and probability matrices can exploit structure in the model and therefore use less memory in practice [45]. Another interesting approach is presented by Buchholz et al. [13], who use Kronecker products to exploit structure.

By removing $\vec{\mu}_0$ from (15) and using $\mathbf{P}^k \cdot \vec{\chi}_\Psi$ as the iteration vector, it is possible to verify a formula in all states simultaneously with the same asymptotic time complexity as for verifying the formula in a single state [42]. Clearly, the same cannot be said of the statistical approach. This gives the numerical solution method a great advantage when dealing with nested probabilistic operators. Consider the formula $\mathcal{P}_{\geq 0.9}\big[\mathcal{P}_{\geq 0.5}[\Diamond^{[0,\tau_2]}\ c]\ \mathcal{U}^{[0,\tau_1]}\ x{=}n \wedge y{=}n\big]$ for the robot grid world. The time complexity for the numerical solution method is essentially $O(q \cdot \tau_1 \cdot M)$ for $\tau_2 < \tau_1$. The statistical solution method, on the other hand, suffers more from the presence of nested probabilistic operators. Younes et al. [69] have suggested a mixed solution method, which uses the numerical approach for nested probabilistic operators and the statistical approach for top-most probabilistic operators. This mixed approach shares performance characteristics of both solution methods, but is limited by memory in the same way as the pure numerical solution method. A brief comparison of the three methods is provided by Younes [66,67].

9 Conclusion and Future Work

This paper establishes the foundations of statistical probabilistic model checking. A key observation is that probabilistic model checking can be modeled as a hypothesis-testing problem. We can therefore use well-established and efficient statistical hypothesis-testing techniques, in particular sequential acceptance sampling, for probabilistic model checking. Our model-checking approach is not tied to any specific statistical test. The only requirement is that we can bound the probability of an incorrect answer (either a false positive or a false negative). Given this, we have shown how to derive error bounds for compound and nested probabilistic statements. The result is a randomized algorithm for probabilistic model checking with indifference regions.

We have considered only transient properties of stochastic systems. The logic CSL, as described by Baier et al. [9], can also express steady-state properties. Statistical techniques for steady-state analysis exist, including *batch means analysis* and *regenerative simulation* [12]. Although these techniques have been used for statistical *estimation*, we are confident that they could be adapted for hypothesis testing, as well. Extending our work on statistical probabilistic model checking to steady-state properties is therefore a prime candidate for future work. We are also looking at ways to ensure termination of our algorithm for properties that involve unbounded until formulae, primarily by using efficient techniques for reachability analysis from symbolic model checking.

To more efficiently handle probability thresholds close to zero and one, the use of *importance sampling* [32] may also be possible. It would moreover be worthwhile exploring Bayesian techniques for acceptance sampling, in particular the test developed by Lai [47]. It is well known that the sequential probability ratio test, while generally very efficient, tends to require a large sample size if the true probability lies in the indifference region of the test. Consequently, we spend the most effort where we are indifferent of the outcome. This shortcoming is addressed by Bayesian hypothesis testing. The challenge would be to devise a Bayesian test for conjunctive and nested probabilistic operators.

A final topic for future work, which we have only briefly touched in this paper, is to improve the efficiency of discrete-event simulation for our representation of stochastic discrete-event systems (a variation of the PRISM language [54]). A bottleneck in our current implementation is the determination of enabled events in a state. Our solution is to scan through the list of all events and evaluate the enabling condition for each event. This is not efficient for models with many events. We think that the use of symbolic data structures, such as MTBDDs, could speed up the generation of sample trajectories.

Acknowledgments

The authors are grateful to the anonymous reviewers, whose detailed comments and insightful remarks helped tremendously in improving this paper.

A Nested Probabilistic Operators and Non-Markovian Models

Our semantics for $\Phi\,\mathcal{U}^I\,\Psi$ requires that Φ holds continuously along a trajectory until Ψ is satisfied. In contrast, Infante López et al.'s [40] semantics of CSL for semi-Markov processes requires Φ to hold only at the time of state transitions. We demonstrate with two examples that the two semantics are incompatible.

Example 15 *Consider the semi-Markov process with two states depicted in Fig. A.1. Assume that G is a standard Weibull distribution with shape parameter 0.5, denoted $W(1, 0.5)$, and that we want to verify the UTSL formula $\Phi = \mathcal{P}_{\geq 0.5}[\varphi]$ in s_0, where φ is the path formula $\mathcal{P}_{\geq 0.5}[x{=}0\,\mathcal{U}^{[0,1]}\,x{=}1]\,\mathcal{U}^{[0,1]}\,x{=}1$.*

Let P denote the set of trajectories that start in s_0 at time 0 and satisfy the path formula φ. Members of P are of the form $\{\langle s_0, t\rangle, \langle s_1, \infty\rangle\}$ with $t \in [0, t']$

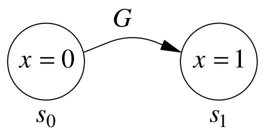


Fig. A.1. A two-state semi-Markov process with holding time distribution G in s_0.

for some $t' \leq 1$. The probability measure of P is therefore at most $F(1) \approx 0.632$, where $F(\cdot)$ is the cumulative distribution function for $W(1, 0.5)$. Of the trajectories with $t \in [0, 1]$, only the ones where $\Psi = \mathcal{P}_{\geq 0.5}[x{=}0 \; \mathcal{U}^{[0,1]} \; x{=}1]$ holds until s_1 is reached satisfy the path formula φ.

If we require Ψ to hold continuously along a trajectory until s_1 is reached, then we have to rule out trajectories with $t \geq t'$ such that Ψ does not hold if verified relative to the trajectory prefix $\{\langle s_0, t' \rangle\}$. The probability of reaching s_1 within 1 time unit, given that we have already spent t' time units in s_0, is

$$q(t') = \frac{1}{1 - F(t')} \int_{t'}^{t'+1} f(x) \; \mathrm{d}x \ ,$$

where $f(\cdot)$ is the probability density function for $W(1, 0.5)$. The value of q is greater than 0.5 for $t' = 0.1$, but less than 0.5 for $t' = 0.2$. Since q is a decreasing function of t', it means that Ψ does not hold continuously over trajectories starting in s_0 if $t \geq 0.2$. It follows that the probability measure of the set P is less than $F(0.2) \approx 0.361$, so Φ does not hold. We would reach the opposite conclusion if we simply verified the nested formulae at the entry of each state, since Ψ holds initially in s_0.

Example 16 *Consider the same semi-Markov process as in the previous example, but this time with G equal to $W(1, 1.5)$. We want to verify $\Phi = \mathcal{P}_{\geq 0.5}[\varphi]$ in s_0, where φ is $x{=}0 \; \mathcal{U}^{[0,1]} \; \mathcal{P}_{\geq 0.7}[x{=}0 \; \mathcal{U}^{(0,1]} \; x{=}1]$. Note that the time interval is open to the left in the formula $\Psi = \mathcal{P}_{\geq 0.7}[x{=}0 \; \mathcal{U}^{(0,1]} \; x{=}1]$, so Ψ cannot hold in s_1 because $x{=}0$ must hold at the entry of a state for Ψ to hold in that state. Ψ does not hold immediately in s_0 either: the probability of reaching s_1 within 1 time unit is $F(1) \approx 0.632 < 0.7$ at time 0 in s_0. The formula Ψ does become true, however, along trajectories that remain in s_0 for 0.2 time units or more before a transition to s_1 occurs. Since $F(1) - F(0.2) \approx 0.547 \geq 0.5$, it follows that Φ holds with the semantics given by Definition 7.*

References

[1] Alur, R., Courcoubetis, C., and Dill, D. L. (1991), Model-checking for probabilistic real-time systems, *in* "Proc. 18th International Colloquium on Automata, Languages and Programming" (J. L. Albert, B. Monien, and M. R. Artalejo, Eds.), pp. 115–126, Springer.

[2] Alur, R., Courcoubetis, C., and Dill, D. L. (1993), Model-checking in dense real-time, *Inform. and Comput.* **104**, 2–34.

[3] Alur, R., and Dill, D. L. (1994), A theory of timed automata, *Theoret. Comput. Sci.* **126**, 183–235.

[4] Anderson, T. W., and Friedman, M. (1960), A limitation of the optimum property of the sequential probability ratio test, *in* "Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling" (I. Olkin, S. G. Ghurye, W. Hoeffding, W. G. Madow, and H. B. Mann, Eds.), pp. 57–69, Stanford University Press, Stanford, California.

[5] Aziz, A., Sanwal, K., Singhal, V., and Brayton, R. K. (1996), Verifying continuous time Markov chains, *in* "Proc. 8th International Conference on Computer Aided Verification" (R. Alur and T. A. Henzinger, Eds.), pp. 269–276, Springer.

[6] Aziz, A., Sanwal, K., Singhal, V., and Brayton, R. K. (2000), Model-checking continuous-time Markov chains, *ACM Trans. Comput. Log.* **1**, 162–170.

[7] Bahar, R. I., Frohm, E. A., Gaona, C. M., Hachtel, G. D., Macii, E., Pardo, A., and Somenzi, F. (1993), Algebraic decision diagrams and their applications, *in* "Proc. 1993 IEEE/ACM International Conference on Computer-Aided Design", pp. 188–191, IEEE Computer Society Press.

[8] Baier, C., Haverkort, B. R., Hermanns, H., and Katoen, J.-P. (2000), Model checking continuous-time Markov chains by transient analysis, *in* "Proc. 12th International Conference on Computer Aided Verification" (E. A. Emerson and A. P. Sistla, Eds.), pp. 358–372, Springer.

[9] Baier, C., Haverkort, B. R., Hermanns, H., and Katoen, J.-P. (2003), Model-checking algorithms for continuous-time Markov chains, *IEEE Trans. Softw. Eng.* **29**, 524–541.

[10] Baier, C., Katoen, J.-P., and Hermanns, H. (1999), Approximate symbolic model checking of continuous-time Markov chains, *in* "Proc. 10th International Conference on Concurrency Theory" (J. C. M. Baeten and S. Mauw, Eds.), pp. 146–161, Springer.

[11] Bartlett, M. S. (1966), *An Introduction to Stochastic Processes with Special Reference to Methods and Applications*, 2nd Edition, Cambridge University Press, London.

[12] Bratley, P., Fox, B. L., and Schrage, L. E. (1987), *A Guide to Simulation*, 2nd Edition, Springer, Berlin.

[13] Buchholz, P., Katoen, J.-P., Kemper, P., and Tepper, C. (2003), Model-checking large structured Markov chains, *J. Log. Algebr. Program.* **56**, 69–97.

[14] Çinlar, E. (1975), *Introduction to Stochastic Processes*, Prentice-Hall, Englewood Cliffs, New Jersey.

[15] Clarke, E. M., and Emerson, E. A. (1982), Design and synthesis of synchronization skeletons using branching time temporal logic, *in* “Proc. 1981 Workshop on Logics of Programs” (D. Kozen, Ed.), pp. 52–71, Springer, Berlin.

[16] Clarke, E. M., Emerson, E. A., and Sistla, A. P. (1986), Automatic verification of finite-state concurrent systems using temporal logic specifications, *ACM Trans. Program. Lang. Syst.* **8**, 244–263.

[17] Clarke, E. M., McMillan, K. L., Zhao, X., and Fujita, M. (1993), Spectral transforms for large Boolean functions with applications to technology mapping, *in* “Proc. 30th International Conference on Design Automation”, pp. 54–60, ACM Press.

[18] Doob, J. L. (1953), *Stochastic Processes*, John Wiley & Sons, New York.

[19] Duncan, A. J. (1974), *Quality Control and Industrial Statistics*, 4th Edition, Richard D. Irwin, Homewood, Illinois.

[20] Emerson, E. A., Mok, A. K., Sistla, A. P., and Srinivasan, J. (1992), Quantitative temporal reasoning, *Real-Time Syst.* **4**, 331–352.

[21] Fox, B. L., and Glynn, P. W. (1988), Computing Poisson probabilities, *Comm. ACM* **31**, 440–445.

[22] Fujita, M., McGeer, P. C., and Yang, J. C.-Y. (1997), Multi-terminal binary decision diagrams: An efficient data structure for matrix representation, *Formal Methods Syst. Des.* **10**, 149–169.

[23] Glynn, P. W. (1989), A GSMP formalism for discrete event systems, *Proc. IEEE* **77**, 14–23.

[24] Grosu, R., and Smolka, S. A. (2005), Monte Carlo model checking, *in* “Proc. 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems” (N. Halbwachs and L. D. Zuck, Eds.), pp. 271–286, Springer.

[25] Grubbs, F. E. (1949), On designing single sampling inspection plans, *Ann. Math. Statist.* **20**, 242–256.

[26] Halmos, P. R. (1950), *Measure Theory*, Van Nostrand Reinhold Company, New York.

[27] Hansson, H., and Jonsson, B. (1989), A framework for reasoning about time and reliability, *in* “Proc. Real-Time Systems Symposium”, pp. 102–111, IEEE Computer Society Press.

[28] Hansson, H., and Jonsson, B. (1994), A logic for reasoning about time and reliability, *Formal Aspects Comput.* **6**, 512–535.

[29] Hart, S., and Sharir, M. (1984), Probabilistic temporal logics for finite and bounded models, *in* “Proc. Sixteenth ACM Symposium on Theory of Computing”, pp. 1–13, ACM SIGACT.

[30] Hastings, Jr., C. (1955), *Approximations for Digital Computers*, Princeton University Press, Princeton, New Jersey.

[31] Haverkort, B. R., Hermanns, H., and Katoen, J.-P. (2000), On the use of model checking techniques for dependability evaluation, *in* "Proc. 19th IEEE Symposium on Reliable Distributed Systems", pp. 228–237, IEEE Computer Society.

[32] Heidelberger, P. (1995), Fast simulation of rare events in queueing and reliability models, *ACM Trans. Model. Comput. Simul.* **5**, 43–85.

[33] Hermanns, H., Meyer-Kayser, J., and Siegle, M. (1999), Multi terminal binary decision diagrams to represent and analyse continuous time Markov chains, *in* "Proc. 3rd International Workshop on the Numerical Solution of Markov Chains" (B. Plateau, W. J. Stewart, and M. Silva, Eds.), pp. 188–207, Prensas Universitarias de Zaragoza.

[34] Hoel, P. G., Port, S. C., and Stone, C. J. (1972), *Introduction to Stochastic Processes*, Houghton Mifflin Company, Boston.

[35] Hogg, R. V., and Craig, A. T. (1978), *Introduction to Mathematical Statistics*, 4th Edition, Macmillan Publishing Co., New York.

[36] Hordijk, A., Iglehart, D. L., and Schassberger, R. (1976), Discrete time methods for simulating continuous time Markov chains, *Adv. in Appl. Probab.* **8**, 772–788.

[37] Howard, R. A. (1971), *Dynamic Probabilistic Systems*, Vol. I: Markov Models, John Wiley & Sons, New York.

[38] Howard, R. A. (1971), *Dynamic Probabilistic Systems*, Vol. II: Semi-Markov and Decision Processes, John Wiley & Sons, New York.

[39] Ibe, O. C., and Trivedi, K. S. (1990), Stochastic Petri net models of polling systems, *IEEE J. Sel. Areas Commun.* **8**, 1649–1657.

[40] Infante López, G. G., Hermanns, H., and Katoen, J.-P. (2001), Beyond memoryless distributions: Model checking semi-Markov chains, *in* "Proc. 1st Joint International PAPM-PROBMIV Workshop" (L. de Alfaro and S. Gilmore, Eds.), pp. 57–70, Springer.

[41] Jensen, A. (1953), Markoff chains as an aid in the study of Markoff processes, *Scand. Actuar. J.* **36**, 87–91.

[42] Katoen, J.-P., Kwiatkowska, M., Norman, G., and Parker, D. (2001), Faster and symbolic CTMC model checking, *in* "Proc. 1st Joint International PAPM-PROBMIV Workshop" (L. de Alfaro and S. Gilmore, Eds.), pp. 23–38, Springer.

[43] Katoen, J.-P., and Zapreev, I. S. (2006), Safe on-the-fly steady-state detection for time-bounded reachability, *in* "Proc. Third International Conference on the Quantitative Evaluation of Systems", IEEE Computer Society, in press.

[44] Kolmogoroff, A. (1931), Über die analytischen Methoden in der Wahrscheinlichkeitsrechnung, *Math. Ann.* **104**, 415–458.

[45] Kwiatkowska, M., Norman, G., and Parker, D. (2004), Probabilistic symbolic model checking with PRISM: A hybrid approach, *Int. J. Softw. Tools Techn. Transfer* **6**, 128–142.

[46] Kwiatkowska, M., Norman, G., Segala, R., and Sproston, J. (2000), Verifying quantitative properties of continuous probabilistic timed automata, *in* "Proc. 11th International Conference on Concurrency Theory" (C. Palamidessi, Ed.), pp. 123–137, Springer.

[47] Lai, T. L. (1988), Nearly optimal squential tests of composite hypotheses, *Ann. Statist.* **16**, 856–886.

[48] Lai, T. L. (2001), Sequential analysis: Some classical problems and new challenges, *Statist. Sinica* **11**, 303–408.

[49] Lassaigne, R., and Peyronnet, S. (2002), Approximate verification of probabilistic systems, *in* "Proc. 2nd Joint International PAPM-PROBMIV Workshop" (H. Hermanns and R. Segala, Eds.), pp. 213–214, Springer.

[50] Malhotra, M., Muppala, J. K., and Trivedi, K. S. (1994), Stiffness-tolerant methods for transient analysis of stiff Markov chains, *Microelectron. Reliab.* **34**, 1825–1841.

[51] Matthes, K. (1962), Zur Theorie der Bedienungsprozesse, *in* "Trans. Third Prague Conference on Information Theory, Statistical Decision Functions, Random Processes" (J. Kožešník, Ed.), pp. 513–528, Publishing House of the Czechoslovak Academy of Sciences.

[52] McCormack, W. M., and Sargent, R. G. (1981), Analysis of future event set algorithms for discrete event simulation, *Comm. ACM* **24**, 801–812.

[53] Michie, D. (1968), "Memo" functions and machine learning, *Nature* **218**, 19–22.

[54] Parker, D. (2002), Implementation of symbolic model checking for probabilistic systems, Ph.D. thesis, School of Computer Science, University of Birmingham, Birmingham, United Kingdom.

[55] Pnueli, A. (1977), The temporal logic of programs, *in* "Proc. 18th Annual Sumposium on Foundations of Computer Science", pp. 46–57, IEEE Computer Society.

[56] Reibman, A., and Trivedi, K. S. (1988), Numerical transient analysis of Markov models, *Comput. Oper. Res.* **15**, 19–36.

[57] Rescher, N., and Urquhart, A. (1971), *Temporal Logic*, Springer, New York.

[58] Schwarz, G. (1962), Asymptotic shapes of Bayes sequential testing regions, *Ann. Math. Statist.* **33**, 224–236.

[59] Segala, R. (1995), Modeling and verification of randomized distributed real-time systems, Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, MIT-LCS-TR-676.

[60] Sen, K., Viswanathan, M., and Agha, G. (2004), Statistical model checking of black-box probabilistic systems, *in* "Proc. 16th International Conference on Computer Aided Verification" (R. Alur and D. A. Peled, Eds.), pp. 202–215, Springer.

[61] Sen, K., Viswanathan, M., and Agha, G. (2005), On statistical model checking of stochastic systems, *in* "Proc. 17th International Conference on Computer Aided Verification" (K. Etessami and S. K. Rajamani, Eds.), pp. 266–280, Springer.

[62] Wald, A. (1945), Sequential tests of statistical hypotheses, *Ann. Math. Statist.* **16**, 117–186.

[63] Wald, A. (1947), *Sequential Analysis*, John Wiley & Sons, New York.

[64] Wald, A., and Wolfowitz, J. (1948), Optimum character of the sequential probability ratio test, *Ann. Math. Statist.* **19**, 326–339.

[65] Younes, H. L. S. (2005), Probabilistic verification for "black-box" systems, *in* "Proc. 17th International Conference on Computer Aided Verification" (K. Etessami and S. K. Rajamani, Eds.), pp. 253–265, Springer.

[66] Younes, H. L. S. (2005), Verification and planning for stochastic processes with asynchronous events, Ph.D. thesis, Computer Science Department, Carnegie Mellon University, Pittsburgh, Pennsylvania, CMU-CS-05-105.

[67] Younes, H. L. S. (2005), Ymer: A statistical model checker, *in* "Proc. 17th International Conference on Computer Aided Verification" (K. Etessami and S. K. Rajamani, Eds.), pp. 429–433, Springer.

[68] Younes, H. L. S. (2006), Error control for probabilistic model checking, *in* "Proc. 7th International Conference on Verification, Model Checking, and Abstract Interpretation" (E. A. Emerson and K. S. Namjoshi, Eds.), pp. 142–156, Springer.

[69] Younes, H. L. S., Kwiatkowska, M., Norman, G., and Parker, D. (2006), Numerical vs. statistical probabilistic model checking, *Int. J. Softw. Tools Techn. Transfer*, in press.

[70] Younes, H. L. S., and Musliner, D. J. (2002), Probabilistic plan verification through acceptance sampling, *in* "Proc. AIPS-02 Workshop on Planning via Model Checking" (F. Kabanza and S. Thiébaux, Eds.), pp. 81–88.

[71] Younes, H. L. S., and Simmons, R. G. (2002), Probabilistic verification of discrete event systems using acceptance sampling, *in* "Proc. 14th International Conference on Computer Aided Verification" (E. Brinksma and K. G. Larsen, Eds.), pp. 223–235, Springer.