Software Tools for Technology Transfer manuscript No.
(will be inserted by the editor)

Numerical vs. Statistical Probabilistic Model Checking*

Håkan L. S. Younes[1], **Marta Kwiatkowska**[2], **Gethin Norman**[2], **David Parker**[2]

[1] Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[2] School of Computer Science, University of Birmingham, Birmingham B15 2TT, United Kingdom

The date of receipt and acceptance will be inserted by the editor

Abstract. Numerical analysis based on uniformisation and statistical techniques based on sampling and simulation are two distinct approaches for transient analysis of stochastic systems. We compare the two solution techniques when applied to the verification of time-bounded until formulae in the temporal stochastic logic CSL, both theoretically and through empirical evaluation on a set of case studies. Our study differs from most previous comparisons of numerical and statistical approaches in that CSL model checking is a hypothesis testing problem rather than a parameter estimation problem. We can therefore rely on highly efficient sequential acceptance sampling tests, which enables statistical solution techniques to quickly return a result with some uncertainty. We also propose a novel combination of the two solution techniques for verifying CSL queries with nested probabilistic operators.

Key words: Model checking – Probabilistic verification – Markov chains – Temporal logic – Transient analysis – Uniformisation – Hypothesis testing – Sequential analysis – Discrete event simulation

1 Introduction

Continuous-time Markov chains (CTMCs) are an important class of stochastic models, widely used in performance and dependability evaluation. The temporal logic CSL (Continuous Stochastic Logic) introduced by Aziz et al. [2,3] and since extended by Baier et al. [6] provides a powerful means to specify both path-based and traditional state-based performance measures on CTMCs in a concise and flexible manner. CSL contains a time-bounded until operator, the focus of this study, that allows one to express properties such as "the probability of n servers becoming faulty within 15.07 seconds is at most 0.01".

Analysis of stochastic systems is typically carried out using either *numerical* or *statistical* solution techniques. Numerical methods can often provide a higher accuracy than statistical methods, whose results are probabilistic in nature. However, numerical methods are far more memory intensive, which often leaves statistical solution techniques as a last resort [31,8].

The verification of time-bounded CSL formulae can be reduced to transient analysis [5,6]. Efficient numerical solution techniques, such as uniformisation [16,26,23,8], for transient analysis of CTMCs have existed for decades and are well-understood. Younes and Simmons [38] have proposed a statistical approach for verifying time-bounded CSL formulae, based on acceptance sampling and discrete event simulation. The use of acceptance sampling is possible because CSL formulae only ask if a probability is above or below some threshold. Previous comparisons of numerical and statistical solution techniques have typically been based on estimation problems. This study is concerned with hypothesis testing problems, for which there exist highly efficient *sequential* acceptance sampling tests that make statistical solution techniques look more favourable than in a comparison with numerical techniques on estimation problems. For probabilistic model checking, it would generally be a waste of effort to obtain a good estimate of a probability, only to realise that it is far from the threshold.

We have implemented the statistical model checking algorithm in YMER [35], which also includes numerical solution engines for time-bounded CSL formulae taken from the PRISM tool [25,18]. PRISM, a probabilistic model checker developed at the University of Birmingham, makes use of symbolic data representation in order to reduce memory requirements for numerical techniques.

Probabilistic model checking in general, and the two approaches implemented in PRISM and YMER, are described in

* Supported in part by DARPA and ARO under contract no. DAAD19–01–1–0485, a grant from the Royal Swedish Academy of Engineering Sciences (IVA), FORWARD and EPSRC grants GR/N22960, GR/S11107 and GR/S46727.

Sect. 2, which includes a theoretical discussion of the computational complexity for the two competing approaches. In this section we also propose a combination of numerical and statistical solution techniques to handle CSL formulae with nested probabilistic operators. The idea of combining the two techniques has been explored before [29,8], but not in the context of nested CSL queries. The mixed solution technique has also been implemented in YMER.

In Sect. 4, we present empirical results obtained using YMER on a number of case studies, described in Sect. 3. This serves as a practical comparison of the two approaches. The empirical evaluation confirms the theoretical complexity results discussed in Sect. 2. It demonstrates that the complexity of both the numerical and the statistical approach is typically linear in the time-bound of the property, but that the statistical approach scales better with the size of the state space. Furthermore, the statistical approach requires considerably less memory than the numerical approach, allowing us to verify models far beyond the scope of numerical solution methods. The principal advantage of numerical techniques based on uniformisation is that increased accuracy in the result comes at almost no price. The statistical solution method can very rapidly provide solutions with some uncertainty, however reducing the uncertainty is costly, making numerical techniques more appropriate when very high accuracy in the result is required.

2 CTMCs and Probabilistic Model Checking

Probabilistic model checking refers to a range of techniques for the formal analysis of systems that exhibit stochastic behaviour. The system is usually specified as a state transition system, with probability values attached to the transitions. In this paper, we consider the case where this model is a continuous-time Markov chain (CTMC).

A CTMC $\mathcal{C}$ is a tuple $\langle S, \mathbf{R}, L\rangle$ where S is a finite set of *states*, $\mathbf{R} : S \times S \to [0,\infty)$ is the *rate matrix* and $L : S \to 2^{AP}$ is a *labelling function*, mapping each state to a subset of the set of atomic propositions AP. For any state $s \in S$, the probability of leaving state s within t time units is given by $1 - e^{-E(s)\cdot t}$ where $E(s) = \sum_{s' \in S} \mathbf{R}(s,s')$. $E(s)$ is known as the *exit rate*. If $\mathbf{R}(s,s') > 0$ for more than one $s' \in S$, then there is a *race* between the transitions leaving s, where the probability of moving to s' in a single step equals the probability that the delay corresponding to moving from s to s' "finishes before" the delays of any other transition leaving s. A *path* of the CTMC is a sequence of states, between each of which there is a non-zero probability of making a transition. A path of the CTMC can be seen as a single execution of the system being modelled.

In probabilistic model checking, properties of the system to be verified are specified in a temporal logic. For CTMCs, we use the temporal logic CSL [2,3,6], an extension of CTL. The syntax of CSL is defined as

$$\Phi \ ::= \top \mid a \mid \Phi \wedge \Phi \mid \neg\Phi \mid \mathcal{P}_{\bowtie\theta}\left[\Phi\ \mathcal{U}^{\leqslant t}\ \Phi\right] \mid \mathcal{S}_{\bowtie\theta}\left[\Phi\right] \ ,$$

where $\theta \in [0,1]$, $t \in [0,\infty]$, $\bowtie \in \{<, \leqslant, \geqslant, >\}$ and a is an atomic proposition from the set AP used to label states of the CTMC. We use $\Phi\ \mathcal{U}\ \Psi$ as a short-form for $\Phi\ \mathcal{U}^{\leqslant\infty}\ \Psi$. Other logic operators, such as disjunction and implication, can be obtained using standard logic equivalence rules.

A state s of a CTMC satisfies the formula $\mathcal{P}_{\bowtie\theta}[\varphi]$, denoted $s \models \mathcal{P}_{\bowtie\theta}[\varphi]$, if $P(s,\varphi) \bowtie \theta$, where $P(s,\varphi)$ is the probability that a path starting in state s satisfies the path formula φ. Here, a path formula φ is either $\Phi\ \mathcal{U}^{\leqslant t}\ \Psi$ for $t \in [0,\infty)$, meaning that formula Ψ is satisfied within t time units and formula Φ is satisfied up until that point, or $\Phi\ \mathcal{U}\ \Psi$, meaning that $\Phi\ \mathcal{U}^{\leqslant t}\ \Psi$ holds for some $t \in [0,\infty)$. The value $P(s,\varphi)$ is defined in terms of the probability measure over paths starting in state s, as defined by Baier et al. [6]. The $\mathcal{S}_{\bowtie\theta}[\Phi]$ operator describes the behaviour of the CTMC in the *steady-state* or long-run. The precise semantics of this and the other CSL operators can be found in [6]. In this paper, we will focus our attention on CSL formulae of the form $\mathcal{P}_{\bowtie\theta}\left[\Phi\ \mathcal{U}^{\leqslant t}\ \Psi\right]$ with *finite* time-bound t.

2.1 Numerical Probabilistic Model Checking

The numerical model checking approach for verifying a time-bounded until formula $\mathcal{P}_{\bowtie\theta}\left[\Phi\ \mathcal{U}^{\leqslant t}\ \Psi\right]$ in a state $s \in S$ is based on first computing the probability $P(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi)$, and then testing if $P(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi) \bowtie \theta$ holds.

First, as initially proposed by Baier et al. [5], the problem is reduced to the computation of transient probabilities on a modified CTMC. For a CTMC $\mathcal{C} = (S, \mathbf{R}, L)$, we construct the CTMC $\mathcal{C}' = (S, \mathbf{R}', L)$ by making all states satisfying $\neg\Phi \vee \Psi$ absorbing, i.e. removing all of their outgoing transitions. Hence, $\mathbf{R}'$ is obtained from $\mathbf{R}$ by removing all entries from the appropriate rows. The probability $P(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi)$ in the CTMC $\mathcal{C}$ is now equal to the probability of, in the CTMC $\mathcal{C}'$, being in a state satisfying Ψ at time t having started in state s.

The computation of this probability is carried out via a process known as *uniformisation* (also known as *randomisation*), originally proposed by Jensen [16]. We construct the *uniformised* discrete-time Markov chain (DTMC) of $\mathcal{C}'$. The probability transition matrix $\mathbf{P}$ of $\mathcal{C}'$ equals $\mathbf{I} + (\mathbf{R}' - \mathbf{E}')/q$, where $\mathbf{I}$ is the identity matrix, $\mathbf{E}'$ is a diagonal matrix containing exit rates of $\mathcal{C}'$, i.e. $\mathbf{E}'(s,s')$ equals $E'(s)$ if $s = s'$ and 0 otherwise, and $q \geqslant \max\{E'(s) \mid s \in S\}$ is the *uniformisation constant* of the CTMC $\mathcal{C}'$.

It then follows that $P(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi)$ can be computed simultaneously for all states $s \in S$ by computing the vector of probabilities

$$\underline{P}(\Phi\ \mathcal{U}^{\leqslant t}\ \Psi) = \sum_{k=0}^{\infty} \gamma(k, q\cdot t) \cdot \left(\mathbf{P}^k \cdot \underline{\Psi}\right) \ , \qquad (1)$$

where $\gamma(k, q\cdot t)$ is the kth Poisson probability with parameter $q\cdot t$ (i.e. $\gamma(k, q\cdot t) = e^{-q\cdot t} \cdot (q\cdot t)^k / k!$), and $\underline{\Psi}$ characterises the set of states satisfying Ψ (i.e. $\underline{\Psi}(s) = 1$ if $s \models \Psi$, and 0 otherwise). If we are only interested in verifying $\mathcal{P}_{\bowtie\theta}\left[\Phi\ \mathcal{U}^{\leqslant t}\ \Psi\right]$

in a single state s, then we only need to carry out the summation in (1) for $P(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi)$, which in practice can save us both memory and time. However, as pointed out by Katoen et al. [17], the asymptotic time complexity is the same when computing the entire vector $\underline{P}(\Phi\ \mathcal{U}^{\leqslant t}\ \Psi)$. In this paper, we only compute the entire vector for nested probabilistic formulae.

In practice, the infinite summation in (1) is truncated from the left and the right by using the techniques of Fox and Glynn [10] so that the truncation error is bounded by an a priori error tolerance ϵ. This means that if $\underline{\hat{P}}(\Phi\ \mathcal{U}^{\leqslant t}\ \Psi)$ is the solution vector obtained with truncation, then

$$0 \leqslant P(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi) - \hat{P}(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi) \leqslant \epsilon \quad \forall s \in S\ . \quad (2)$$

The left and right truncation points, denoted L_ϵ and R_ϵ respectively, depend on the truncation error ϵ. Note that, since iterative squaring is generally not attractive for sparse matrices due to fill-in [30,26], the matrix products $\mathbf{P}^k$ are typically computed in an iterative fashion: $\mathbf{P}^k \cdot \underline{\Psi} = \mathbf{P} \cdot \left(\mathbf{P}^{k-1} \cdot \underline{\Psi}\right)$. Thus, although the left truncation point L_ϵ allows us to skip the first L_ϵ terms of (1), we still need to compute $\mathbf{P}^k \cdot \underline{\Psi}$ for $k < L_\epsilon$, making the total number of iterations required by the algorithm R_ϵ. The value of R_ϵ is $q{\cdot}t + k\sqrt{2q{\cdot}t} + 3/2$, where k is $o\left(\sqrt{\log(1/\epsilon)}\right)$ [10]. This means that the number of iterations grows very slowly as ϵ decreases. For large values of $q{\cdot}t$, the number of iterations is essentially $O(q{\cdot}t)$. Each iteration involves a matrix-vector multiplication and each such operation takes $O(M)$ time, where M is the number of non-zero entries in the rate matrix $\mathbf{R}$. The time complexity for the numerical solution technique is therefore $O(q{\cdot}t{\cdot}M)$ (cf. [23]).

2.1.1 Steady-State Detection

To potentially reduce the number of iterations required by the numerical model checking algorithm, we can use on-the-fly steady-state detection in conjunction with uniformisation [26,23]. If the uniformised DTMC reaches steady-state after $k_s < R_\epsilon$ iterations, then $\mathbf{P}^k \cdot \underline{\Psi} = \mathbf{P}^{k_s} \cdot \underline{\Psi}$ for all $k \geqslant k_s$, which means that we can compute $\underline{\hat{P}}(\Phi\ \mathcal{U}^{\leqslant t}\ \Psi)$ as follows using only k_s iterations:

$$\begin{aligned}\underline{\hat{P}}(\Phi\ \mathcal{U}^{\leqslant t}\ \Psi) = &\sum_{k=L_\epsilon}^{k_s} \gamma(k, q{\cdot}t) \cdot \left(\mathbf{P}^k \cdot \underline{\Psi}\right) \\ &+ \left(1 - \sum_{k=L_\epsilon}^{k_s} \gamma(k, q{\cdot}t)\right) \cdot \left(\mathbf{P}^{k_s} \cdot \underline{\Psi}\right)\ .\end{aligned} \quad (3)$$

We can ensure that a steady-state vector actually exists by choosing q strictly greater than $\max\{E(s) \mid s \in S\}$ [26,23].

Malhotra et al. [23] derive an error bound for (3) under the assumption that the steady-state point can be detected exactly within a given error tolerance. Let $\underline{\Pi}^*$ denote the true steady-state vector. Malhotra et al. claim that if $\|\mathbf{P}^{k_s} \cdot \underline{\Psi} - \underline{\Pi}^*\| \leqslant \epsilon/4$ for $L_\epsilon < k_s < R_\epsilon$, then the same error bound as in (2) is guaranteed. The error analysis is flawed, however, in that it results in an error region twice as wide as the original error region. This is a result of the error due to steady-state detection being two-sided, while the truncation error is one-sided. To guarantee an error region of width ϵ instead of 2ϵ, it is necessary to bound $\|\mathbf{P}^{k_s} \cdot \underline{\Psi} - \underline{\Pi}^*\|$ by $\epsilon/8$ instead of $\epsilon/4$. This correction yields the error bounds $-\epsilon/4 \leqslant P(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi) - \hat{P}(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi) \leqslant 3\epsilon/4$ for all $s \in S$.

In practice, the true steady-state vector $\underline{\Pi}^*$ is not known a priori, so instead we stop when the norm of the difference between successive iterates is sufficiently small (at most $\epsilon/8$ by the above analysis), as suggested by Malhotra et al. [23].

2.1.2 The Sequential Stopping Rule

To potentially reduce the number of iterations even further, we note that the CSL query $\mathcal{P}_{\bowtie\theta}\left[\Phi\ \mathcal{U}^{\leqslant t}\ \Psi\right]$ does not require that we compute $P(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi)$ with higher accuracy than is needed to determine whether $P(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi) \bowtie \theta$ holds. In the following analysis we restrict $\bowtie$ to $\geqslant$ as the other three cases are essentially the same.

Let $\hat{P}^k(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi)$ denote the accumulated probability up until and including iteration k. Because each term in (1) is non-negative, we have $\hat{P}^i(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi) \geqslant \hat{P}^k(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi)$ for all $i > k$. Therefore if $\hat{P}^k(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi) \geqslant \theta$ holds for some $k < R_\epsilon$, then we can answer the query $\mathcal{P}_{\geqslant\theta}\left[\Phi\ \mathcal{U}^{\leqslant t}\ \Psi\right]$ affirmatively after only k iterations instead of R_ϵ (or k_s) iterations.

For early termination with a negative result, we can use the upper bound on the right Poisson tail provided by Fox and Glynn [10] for $k \geqslant 2 + \lfloor q \cdot t \rfloor$ to determine if $\mathcal{P}_{\geqslant\theta}\left[\Phi\ \mathcal{U}^{\leqslant t}\ \Psi\right]$ is false before completing R_ϵ iterations. Let $\hat{T}$ be the upper bound on the right Poisson tail. If $\hat{P}^k(s, \Phi\ \mathcal{U}^{\leqslant t}\ \Psi) + \hat{T} < \theta$, then we know already after k iterations that $\mathcal{P}_{\geqslant\theta}\left[\Phi\ \mathcal{U}^{\leqslant t}\ \Psi\right]$ is false.

We now have a sequential stopping rule for our algorithm, but note that the potential savings are limited by the fact that the positive part of the rule applies first after L_ϵ iterations and the negative part first after $2 + \lfloor q{\cdot}t \rfloor$ iterations, and both L_ϵ and R_ϵ are of the same order of magnitude as $q{\cdot}t$. We will see later that the sequential component of the statistical approach is much more significant.

2.1.3 Symbolic Representation

PRISM uses binary decision diagrams (BDDs) [7] and multi-terminal BDDs (MTBDDs) [9,4,11] to construct a CTMC from a model description in the PRISM language, a variant of Alur and Henzinger's Reactive Modules formalism [1]. For numerical computation though, PRISM includes three separate *engines* making varying use of symbolic methods.

The first engine uses MTBDDs to store the model and iteration vector, while the second uses conventional data structures for numerical analysis: sparse matrices and arrays. The latter nearly always provides faster numerical computation than its MTBDD counterpart, but sacrifices the ability to save memory by exploiting structure. A third, *hybrid*, engine provides a compromise by storing the models in an MTBDD-like structure, which is adapted so that numerical computation can

be carried out in combination with array-based storage for vectors. This hybrid approach is generally faster than MTBDDs, while handling larger systems than sparse matrices, and hence is the one used in this paper. For further details and comparisons between the engines see [19,24].

2.2 *Statistical Probabilistic Model Checking*

Statistical techniques, involving Monte Carlo simulation and sampling, have been in use for decades to analyse stochastic systems. Younes and Simmons [38] show how discrete event simulation and acceptance sampling can be used to verify properties of general discrete event systems expressed as CSL formulae not including $\mathcal{P}_{\bowtie\theta}\left[\Phi\,\mathcal{U}\,\Phi\right]$ and $\mathcal{S}_{\bowtie\theta}\left[\Phi\right]$. We focus here on $\mathcal{P}_{\geqslant\theta}\left[\Phi\,\mathcal{U}^{\leqslant t}\,\Psi\right]$, noting that $\mathcal{P}_{\leqslant\theta}\left[\Phi\,\mathcal{U}^{\leqslant t}\,\Psi\right] \equiv \neg\mathcal{P}_{>\theta}\left[\Phi\,\mathcal{U}^{\leqslant t}\,\Psi\right]$, and also that $>$ ($<$) is practically indistinguishable from $\geqslant$ ($\leqslant$) to any acceptance sampling test. The same can of course be said of numerical approaches as well due to the use of finite precision floating-point arithmetic.

Given a CTMC $\mathcal{C}$, a state s of $\mathcal{C}$, and a CSL formula $\mathcal{P}_{\geqslant\theta}\left[\varphi\right]$, we wish to test whether the probability $P(s,\varphi)$, for the remainder of this section simply denoted p, is above or below the threshold θ. We can formulate this as the problem of testing the hypothesis $H : p \geqslant \theta$ against the alternative hypothesis $K : p < \theta$. In this section, we discuss how to solve such a problem using statistical hypothesis testing.

2.2.1 Statistical Hypothesis Testing

Let X_i be a random variable representing the verification of the path formula φ over a path for $\mathcal{C}$ drawn at random from the set of paths starting in state s. If we choose $X_i = 1$ to represent the fact that φ holds over a random path, and $X_i = 0$ to represent the opposite fact, then X_i is a *Bernoulli variate* with parameter p, i.e. $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = 1 - p$. An observation of X_i, denoted x_i, is the verification of φ over a specific path σ_i. If σ_i satisfies the path formula φ, then $x_i = 1$, otherwise $x_i = 0$. In our case, an observation is obtained by generating a sample path, σ_i, using discrete event simulation, and then testing if σ_i satisfies φ. Note that we can perform simulation at the level of the PRISM language and never need to generate the underlying CTMC.

Whenever we consider statistical approaches for solving hypothesis testing problems, we generally have to tolerate that any test procedure we use can accept a false hypothesis, but this is satisfactory so long as the probability of error is sufficiently low. In particular, the test procedure should limit the probability of accepting the hypothesis K when H holds (known as a type I error, or false negative) to α, and the probability of accepting H when K holds (a type II error, or false positive) should be at most β, with $\alpha + \beta \leqslant 1$. Figure 1 plots the probability of accepting H as a function of p, denoted L_p, for a hypothetical acceptance sampling test with ideal performance in the sense that the type I error is exactly α and the type II error is exactly β. The parameters α and β determine the *strength* of an acceptance sampling test.

The above problem formulation is flawed, however, since, in the case where p is equal to the threshold θ, we simultaneously require the hypothesis H to be accepted both with probability at least $1-\alpha$ and with probability at most β. For this to work, we need to have $1 - \alpha = \beta$, which means that we cannot control the two error probabilities independently. In order to obtain the desired control over the two error probabilities, we relax the hypothesis testing problem by introducing two thresholds p_0 and p_1, with $p_0 > \theta > p_1$. Instead of testing H against K, we choose to test the hypothesis $H_0 : p \geqslant p_0$ against the alternative hypothesis $H_1 : p \leqslant p_1$. 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. Figure 2 shows the typical performance characteristic of a realistic acceptance sampling test. If the value of p is between p_0 and p_1, 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 referred to as the *indifference region* and it is shown as a gray area in Fig. 2.

For CSL model checking, the two thresholds p_0 and p_1 can be defined in terms of a single threshold θ and the half-width of the indifference region δ, i.e. $p_0 = \theta + \delta$ and $p_1 = \theta - \delta$. Testing H_0 against H_1 can then be interpreted as testing the hypothesis $H : p \geqslant \theta$ against the alternative hypothesis $K : p < \theta$, as originally specified, where acceptance of H_0 results in acceptance of H and acceptance of H_1 results in acceptance of K. The probability of accepting H is therefore at least $1 - \alpha$ if $p \geqslant \theta + \delta$ and at most β if $p \leqslant \theta - \delta$. If $|p - \theta| < \delta$, then the test gives no bounds on the probability of accepting a false hypothesis. In this case, however, we say that p is sufficiently close to the threshold θ so that we are indifferent with respect to which of the two hypotheses, H or K, is accepted. By narrowing the indifference region, we can get arbitrarily close to the ideal performance shown in Fig. 1.

2.2.2 The Sequential Probability Ratio Test

We use Wald's *sequential probability ratio test* [32] to test the hypothesis $H_0 : p \geqslant p_0$ against the alternative hypothesis $H_1 : p \leqslant p_1$. The sequential probability ratio test does not use a predetermined number of observations, but instead determines after each observation if another observation is needed or if the information currently available is sufficient to accept a hypothesis so that the test has the prescribed strength. A statistical procedure that takes observations into account as they are made is called a *sequential* procedure.

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

$$\frac{p_{1m}}{p_{0m}} = \prod_{i=1}^{m} \frac{\Pr[X_i = x_i | p = p_1]}{\Pr[X_i = x_i | p = p_0]} = \frac{p_1^{d_m}(1-p_1)^{m-d_m}}{p_0^{d_m}(1-p_0)^{m-d_m}} \quad ,$$

where $d_m = \sum_{i=1}^{m} x_i$. The quantity p_{jm} is the probability of the observation sequence $x_1, \ldots, x_m$ given that $\Pr[X_i = 1] = p_j$ making the computed quantity a ratio of two probabilities, hence the phrase *probability ratio* in the name of the

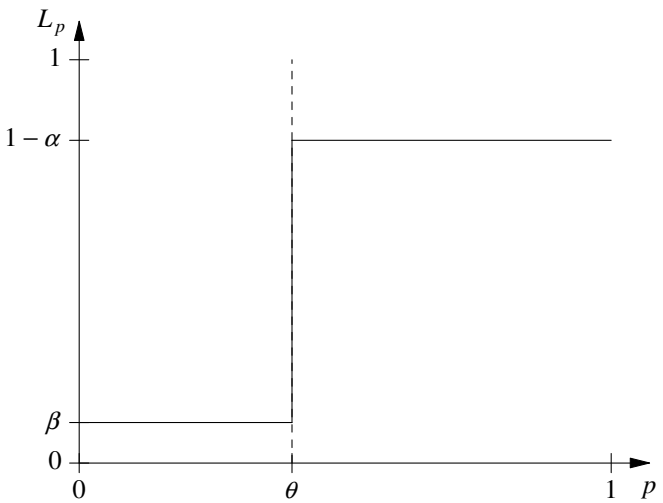


Fig. 1. Probability, L_p, of accepting the hypothesis $H : p \geqslant \theta$ as a function of p for a hypothetical statistical test.

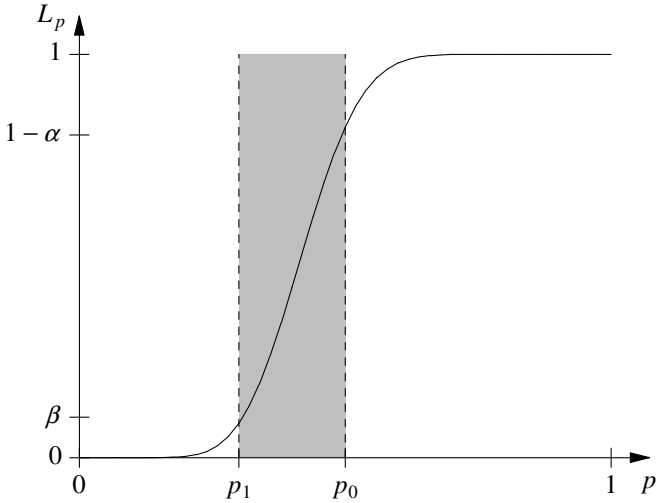


Fig. 2. Probability, L_p, of accepting the hypothesis $H_0 : p \geqslant p_0$ as a function of p for a statistical test with indifference region.

test. Hypothesis H_0 is accepted if

$$\frac{p_{1m}}{p_{0m}} \leqslant B \tag{4}$$

and hypothesis H_1 is accepted (i.e. H_0 is rejected) if

$$\frac{p_{1m}}{p_{0m}} \geqslant A \tag{5}$$

where 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. Otherwise, additional observations are made until either (4) or (5) is satisfied.

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 [32] shows that the following inequalities hold:

$$\alpha' \leqslant \frac{\alpha}{1-\beta} \tag{6}$$

$$\beta' \leqslant \frac{\beta}{1-\alpha} \tag{7}$$

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 [32] also shows that $\alpha' + \beta' \leqslant \alpha + \beta$, so at least one of the inequalities $\alpha' \leqslant \alpha$ and $\beta' \leqslant \beta$ must hold, and in practice we often find that both inequalities hold.

2.2.3 Complexity of the Statistical Solution Method

The complexity of the statistical approach depends on the number of observations, also called the *sample size*, required to reach a decision, as well as the time required for each observation. An observation involves the verification of a CSL path formula φ over a sample path σ_i. The sample size for the sequential probability ratio test is a random variable, and so is the time per observation, which means that we can only talk about the *expected* complexity of the method.

First, consider the time per observation. A sample path σ_i may very well be infinite, but in order to verify the path formula $\varphi = \Phi \, \mathcal{U}^{\leqslant t} \, \Psi$, we only need to generate a prefix of σ_i. We stop as soon as we reach a state satisfying $\neg\Phi \vee \Psi$, or if the time limit t is exceeded. In each state along the prefix, we verify the formulae Φ and Ψ. If both of these formulae are classical logic expressions, i.e. contain no probabilistic operators, then we can treat the time required per state as a constant. Consequently, the time per observation is proportional to the length of the prefix of σ_i required to determine the truth-value of the path formula φ. If we are lucky, a state satisfying $\neg\Phi \vee \Psi$ is reached early, but in the worst case we have to continue until the time limit t is exceeded. The expected time per observation is therefore $O(q \cdot t)$, where q is the maximum exit rate of the CTMC model. In order to obtain a tighter bound, we would have to know more about the CTMC than just its maximum exit rate.

The second component of the complexity is the expected sample size, which for a sequential test depends on the unknown parameter p. Let N_p denote the expected sample size as a function of the parameter p. The complexity of the statistical model checking approach is then $O(q \cdot t \cdot N_p)$, which can be compared to $O(q \cdot t \cdot M)$ for the numerical solution method. The expected sample size for statistical hypothesis testing, in general, depends on the error bounds α and β, and the probability thresholds p_0 and p_1 (alternatively expressed using the threshold θ and the half-width of the indifference region δ). There is, however, no immediate dependence between the expected sample size and the size of the state space for the CTMC model. This is in sharp contrast to the time complexity for the numerical method where the factor M, the number of non-zero entries in the rate matrix $\mathbf{R}$, is at least linear in the size of the state space.

An exact formula for the expected sample size required by the sequential probability ratio test is not available, but

p	$\tilde{N}_p$
0	$\dfrac{\log \dfrac{1-\beta}{\alpha}}{\log \dfrac{1-p_1}{1-p_0}}$
p_1	$\dfrac{\beta \log \dfrac{\beta}{1-\alpha} + (1-\beta) \log \dfrac{1-\beta}{\alpha}}{p_1 \log \dfrac{p_1}{p_0} + (1-p_1) \log \dfrac{1-p_1}{1-p_0}}$
s	$\dfrac{-\log \dfrac{\beta}{1-\alpha} \log \dfrac{1-\beta}{\alpha}}{\log \dfrac{p_1}{p_0} \log \dfrac{1-p_0}{1-p_1}}$
p_0	$\dfrac{(1-\alpha) \log \dfrac{\beta}{1-\alpha} + \alpha \log \dfrac{1-\beta}{\alpha}}{p_0 \log \dfrac{p_1}{p_0} + (1-p_0) \log \dfrac{1-p_1}{1-p_0}}$
1	$\dfrac{\log \dfrac{\beta}{1-\alpha}}{\log \dfrac{p_1}{p_0}}$

Table 1. Approximate expected sample size for the sequential probability ratio test. The approximation formulae for $p = 0$ and $p = 1$ differ from those derived by Wald [33]. This is because we assume $p_0 > p_1$, while Wald assumes the opposite.

Wald [32] provides

$$\tilde{N}_p = \frac{L_p \log \dfrac{\beta}{1-\alpha} + (1-L_p) \log \dfrac{1-\beta}{\alpha}}{p \log \dfrac{p_1}{p_0} + (1-p) \log \dfrac{1-p_1}{1-p_0}} \tag{8}$$

as a good approximation of N_p when p_1 is not far from p_0, which is typically the case in practice. The quantity L_p is the probability of accepting H_0 when $\Pr[X_i = 1] = p$. Wald provides an approximation formula for L_p as well, but the formula is not suited for computing an approximation of L_p for an arbitrary p. Approximating N_p for an arbitrary p is therefore non-trivial, but we can provide explicit formulae for a few cases of special interest shown in Table 1, with

$$s = \frac{\log \dfrac{1-p_0}{1-p_1}}{\log \dfrac{p_1(1-p_0)}{p_0(1-p_1)}} \ . \tag{9}$$

The expected sample size increases as p goes from 0 to p_1 and decreases as p goes from p_0 to 1. In the indifference region (p_1, p_0), the sample size increases from p_1 to some point p' and decreases from p' to p_0. The point p' is generally equal to s or at least very near s [33].

An amazing property of the sequential probability ratio test is that it minimises the expected sample size at both p_0 and p_1, i.e. no other statistical test with the same strength will have a lower expected sample size if the unknown parameter p is equal to either p_0 or p_1 [34]. 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 happens to be in the indifference region. Alternative approaches have therefore been suggested, in particular so called *Bayesian* approaches where the objective is to minimise the expected cost subject to a cost c per observation and a unit cost for accepting a false hypothesis [27,21]. While this and other 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 [22] for a more detailed account of the developments in the field of sequential hypothesis testing since the ground breaking work of Wald.

2.3 Mixing Numerical and Statistical Techniques

Although the algorithm of Younes and Simmons [38] can handle CSL formulae with nested probabilistic operators, the way in which it is done requires that the nested formulae be verified in each state along a sample path. Since the verification of path formulae now involves acceptance sampling, there is some probability of error associated with each observation used for the verification of the outer probabilistic operator. To cope with this uncertainty in the observations, the indifference region of the outer probabilistic operator must be reduced, which leads to an increase in the expected sample size. In addition, the nested formulae must be verified with lower values for α and β than used with the outer probabilistic operator. The numerical approach, on the other hand, can verify the nested formulae for all states simultaneously at the same (asymptotic) cost as verifying the formulae for a single state. This is beneficial when dealing with nested probabilistic operators.

We therefore propose a mixed approach, implemented in YMER, where statistical sampling is used to verify the outermost probabilistic operator, while numerical techniques are used to verify the nested probabilistic operators. We can mix the numerical and statistical techniques by assuming that the result of the numerical technique holds with certainty (i.e. $\alpha = \beta = 0$ in terms of a statistical test). The nested formulae are first verified for all states using numerical methods. When verifying a path formula over a sample path we only need to read the value for each state along the path without any additional verification effort for the nested formulae. The cost for verifying the nested components of a formula is exactly the same for the mixed approach as for the numerical approach, but the use of sampling for the outermost probabilistic operator can provide a faster solution.

The time complexity for the pure numerical approach is $O(q{\cdot}t{\cdot}M + q{\cdot}t'{\cdot}M)$, when used to verify a CSL formula with a single nested probabilistic operator and time bounds of t and t' for the outer and nested until formulas, respectively. The mixed solution method replaces the uniformisation step for the outer probabilistic operator with statistical hypothesis testing, which therefore yields an overall time complexity of $O(q{\cdot}t{\cdot}N_p + q{\cdot}t'{\cdot}M)$.

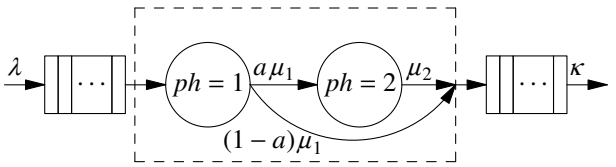


Fig. 3. Tandem queueing network with a two-phase Coxian distribution governing the routing time between the queues.

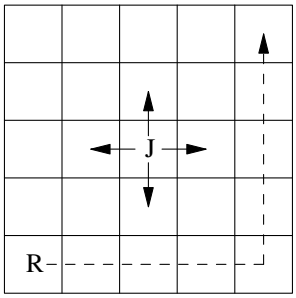


Fig. 4. A grid world with a robot (R) in the bottom left corner and a janitor (J) in the centre. The dashed arrow indicates the path of the robot. The janitor moves with equal probability to any of the adjacent squares.

3 Case Studies

We now present two case studies, taken from the literature on performance evaluation and probabilistic model checking, on which we will base our empirical evaluation. A third simple example is also introduced to illustrate the use of nested probabilistic operators in CSL. More information about all of these case studies can be found on the PRISM web site [25].

3.1 Tandem Queueing Network

The first case study is based on a CTMC model of a tandem queueing network presented by Hermanns et al. [12]. The network consists of an $\mathrm{M}/\mathrm{Cox}_2/1$ queue sequentially composed with an $\mathrm{M}/\mathrm{M}/1$ queue. The capacity of each queue is n, and the state space is $O(n^2)$. The property of interest is given by the CSL formula $\mathcal{P}_{<0.5}\left[\top\, \mathcal{U}^{\leqslant T}\ \mathit{full}\right]$ which is true in a state if there is a less than 50% chance of the queueing network becoming full within T time units. We verify the correctness of this property in the state where both queues are empty.

Figure 3 shows a schematic view of the tandem queueing network. Messages arrive at the first queue with rate λ, and exit the system from the second queue with rate κ. If the first queue is not empty and the second queue is not full, then messages are routed from the first to the second queue. The routing time is governed by a two-phase Coxian distribution with parameters μ_1, μ_2, and a. Here, μ_i is the exit rate for the ith phase of the distribution, and $1 - a$ is the probability of skipping the second phase. Let $s_i \in \{0, \ldots, n\}$, for $i \in \{1, 2\}$, denote the number of messages currently in queue i, and let $ph \in \{1, 2\}$ denote the current phase of the Coxian distribution. We express the condition that the system is full with the formula $\mathit{full} \equiv s_1{=}n \wedge s_2{=}n \wedge ph{=}2$, and the state in which we verify the CSL formula $\mathcal{P}_{<0.5}\left[\top\, \mathcal{U}^{\leqslant T}\ \mathit{full}\right]$ is given by $s_1{=}0 \wedge s_2{=}0 \wedge ph{=}1$, i.e. when the system is empty.

3.2 Symmetric Polling System

For the next case study, we consider an n-station symmetric polling system described by Ibe and Trivedi [14]. 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, as is the time before arrival of a new message at each station. The fact that arrival rates are equal for all stations makes the system symmetric. 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 the buffer of station 1 is full, then it is polled within T time units with probability at least 0.5. We do so 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 CSL as $m_1{=}1 \rightarrow \mathcal{P}_{\geqslant 0.5}\left[\top\, \mathcal{U}^{\leqslant T}\ \mathit{poll1}\right]$, where $\mathit{poll1} \equiv s{=}1 \wedge a{=}0$, and the state in which we verify the formula is given by $s{=}1 \wedge a{=}1 \wedge m_1{=}1 \wedge \cdots \wedge m_n{=}1$.

3.3 Grid World

The final case study involves a robot navigating in a grid world, and is introduced to illustrate the verification of formulae with nested probabilistic operators. We have an $n \times n$ grid world with a robot moving from the bottom left corner to the top right corner. The robot first moves along the bottom edge and then along the right edge. In addition to the robot, there is a janitor moving randomly around the grid. Figure 4 provides a schematic view of a grid world with $n = 5$.

The robot moves at rate λ_R, unless the janitor occupies the destination square, in which case the robot remains stationary. The janitor moves around randomly in the grid world at rate λ_J, selecting the destination from the set of neighboring squares according to a discrete uniform distribution. The robot initiates communication with a base station at rate μ, and the duration of each communication session is exponentially distributed with rate κ.

The objective is for the robot to reach the goal square at the top right corner within T_1 time units with probability at least 0.9, while maintaining at least a 0.5 probability of periodically communicating with the base station (on average, at least once every T_2 time units). The CSL formula $\mathcal{P}_{\geqslant 0.9}\left[\left(\mathcal{P}_{\geqslant 0.5}\left[\top\, \mathcal{U}^{\leqslant T_2}\ \mathit{comm}\right]\right)\ \mathcal{U}^{\leqslant T_1}\ \mathit{goal}\right]$ expresses the

given objective. There is nothing in the model that obstructs communication, but the communication requirement will not be satisfied in any states if T_2 is too low. The size of the state space is $O(n^3)$ for this case study.

4 Empirical Evaluation

We base our empirical evaluation on the case studies presented in Sect. 3. We have verified the time-bounded until formulae for the first two case studies using both the numerical and the statistical approach, varying the problem size (and thereby the size of the state space) and the time bound. All results, for both the numerical and the statistical approach, are for the verification of a CSL property in a *single* state. The results were generated on a 500 MHz Pentium III PC running Linux, and with a 700 MB memory limit set per process.

Figs. 5 and 6 present statistics for model checking of the tandem queueing network and symmetric polling system case studies, respectively. In each case, we include the verification time for both the numerical solution method and the statistical solution method using various test strengths (subfigures (a) and (b)). We also give, for the statistical approach, details of both sample size (subfigures (c) and (d)) and path length (subfigures (e) and (f)). For all data, we plot the results against both model size (subfigures (a), (c) and (e)) and against the CSL until time bound (subfigures (b), (d) and (f)).

For the numerical approach, we used a precision of $\epsilon = 10^{-6}$. For the statistical approach, we used an indifference region with $2\delta = 10^{-2}$, unless otherwise noted, and the results obtained correspond to the average over ten runs.

4.1 Memory Requirements

In the case of the numerical solution method, all experiments were run using the hybrid engine (see Sect. 2.1) which, although not necessarily the fastest engine, in general allows the analysis of larger problems than the other engines. The limiting factor in this approach is the space required to store the iteration vector: however compact the matrix representation is, memory proportional to the number of states is required for numerical solution.

More precisely, the hybrid engine with steady-state detection requires storage of three double precision floating point vectors of size $|S|$, which for the memory limit of 700 MB means that systems with at most 31 million states can be analysed. We need an additional floating point vector of size $|S|$ for verifying a formula in all states simultaneously, which is done for nested probabilistic formulae, and this would make the limit 23 million states. In practice, for the first two case studies, we were able to handle systems with about 27 million states, showing that the symbolic representations of the probability matrices are fairly compact.

The memory requirements for the statistical approach are very conservative. In principle, all that we need to store during verification is the current state, which only requires memory logarithmic in the size of the state space. We never exhausted memory during verification when using the statistical solution method.

4.2 Performance of the Numerical Solution Method

For model checking time-bounded until formulae using the numerical approach, PRISM computes the Poisson probabilities (see Sect. 2.1) using the Fox-Glynn algorithm [10]. This yields, for the hybrid engine, an overall time complexity of $O(q{\cdot}t{\cdot}M)$, where q is the uniformisation constant described earlier, t is the time bound of the until formula and M is the number of non-zero entries in $\mathbf{R}$.

In all the examples considered, the number of non-zeros in the rate matrix is linear in the size of the state space. Hence, the verification time for a given time-bounded until formula is linear in the size of the state space, as can be observed in Figs. 5(a) and 6(a). The *decrease* in Fig. 5(a) of the verification time is caused by the fact that, for models above a certain size, on-the-fly steady-state detection is triggered during the numerical computation (see Sect. 2.1.1).

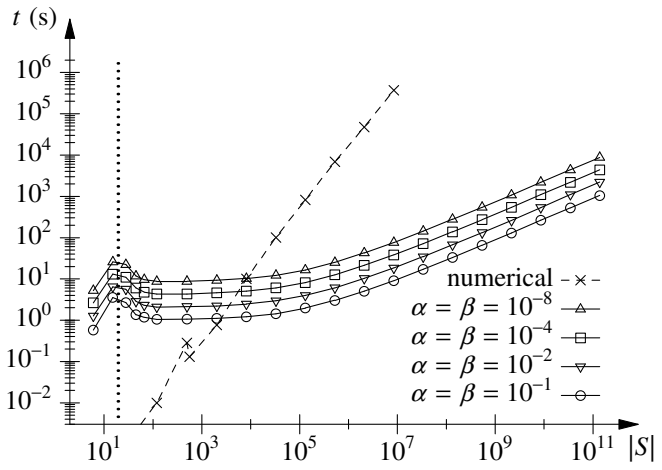
For a single model, the complexity is linear in the time bound, as demonstrated by the results in Figs. 5(b) and 6(b). Note that the verification time in both these cases becomes constant once the time bound has become sufficiently large. This is caused by steady-state detection, which gives the numerical approach a distinct advantage over the statistical approach in the tandem queueing network case study.

The sequential stopping rule (see Sect. 2.1.2) for the numerical solution method also helps to reduce the verification time, but the reduction is typically moderate. Figure 7 shows, for the symmetric polling system ($T = 20$), the number of iterations required for solution, as a fraction of R_ϵ (the upper truncation point provided by the Fox-Glynn algorithm, i.e. the maximum possible number of iterations). Note that, in all cases, this fraction is less than one. For the smaller models, the large reduction in iterations is due to steady-state detection. On the otherhand, for larger models the reduction is caused by the sequential stopping rule (corresponding to the plateau in the graph). The reduction stays at around 10 percent, and starts to get smaller for larger models because the probability of the path formula holding gets closer to the threshold 0.5.

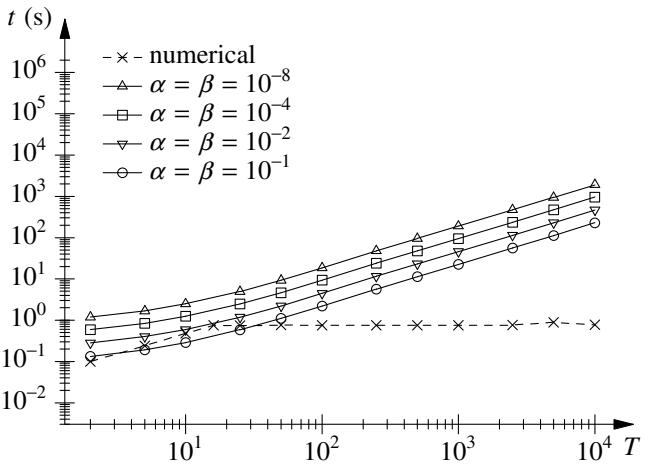
4.3 Performance of the Statistical Solution Method

As discussed in Sect. 2.2, there are two main factors influencing the verification time for the statistical approach: the number of samples and the length of sample paths (in terms of state transitions). Consider the problem of verifying the formula $\mathcal{P}_{\geqslant\theta}[\varphi]$ in state s, with $p = P(s,\varphi)$ denoting the probability that φ holds over paths starting in s.

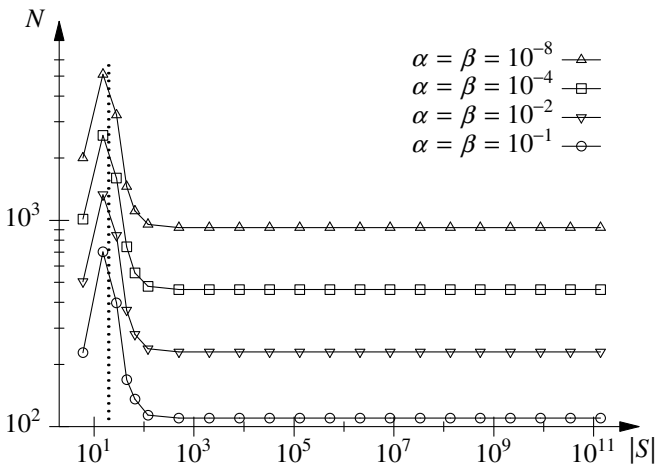
For fixed α and β (test strength) and δ (indifference region), the number of samples grows larger the closer θ gets to the probability p. The peaks in the curves for the statistical



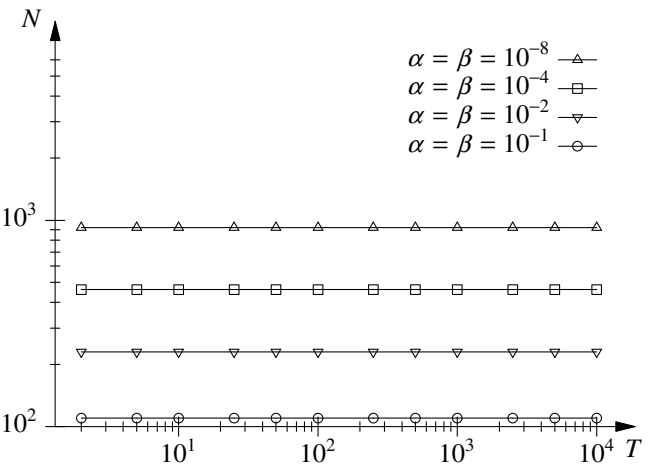
(a) Verification time as a function of state space size.



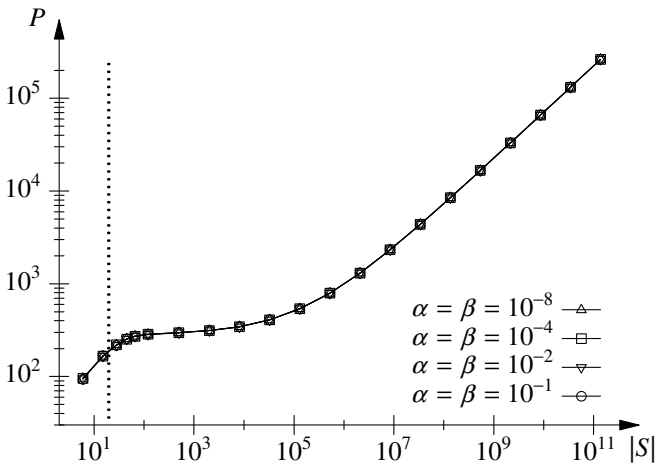
(b) Verification time as a function of formula time bound.



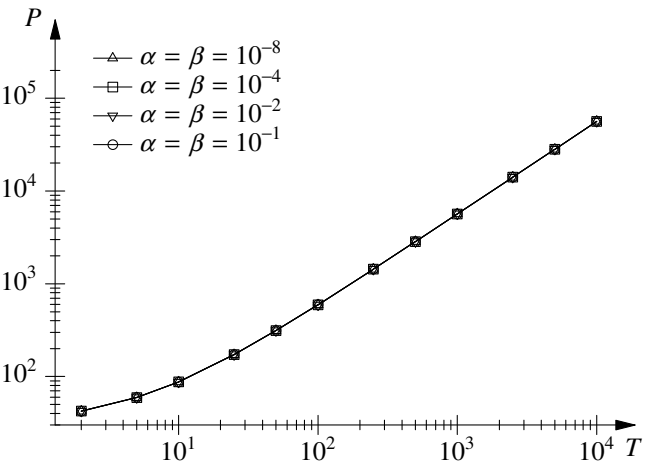
(c) Sample size as a function of state space size.



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

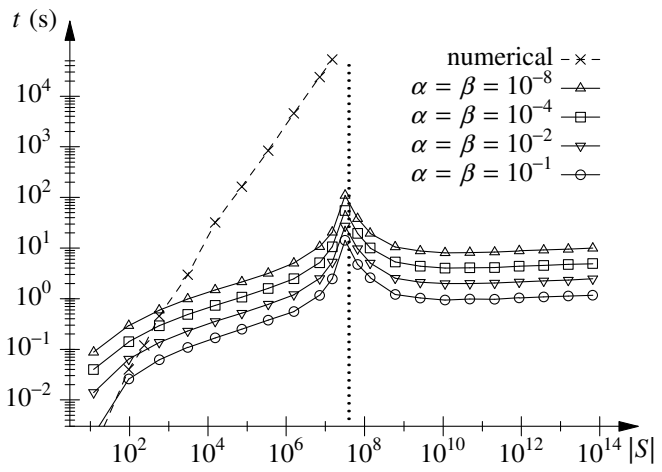


(e) Path length as a function of state space size.

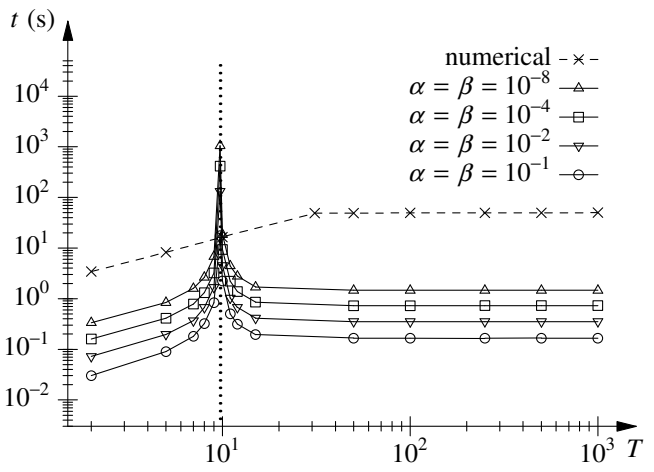


(f) Path length as a function of formula time bound.

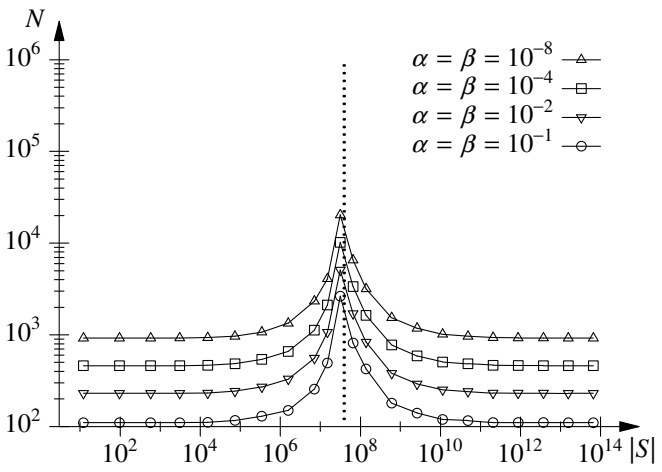
Fig. 5. Empirical results for the tandem queueing network, with $T = 50$ to the left and $n = 31$ to the right. The dashed curves in (a) and (b) are for the numerical approach using the hybrid engine with $\epsilon = 10^{-6}$. The solid curves are for the statistical approach with $2\delta = 10^{-2}$, and α and β as indicated. The dotted lines mark a change in the truth value of the formula being verified.



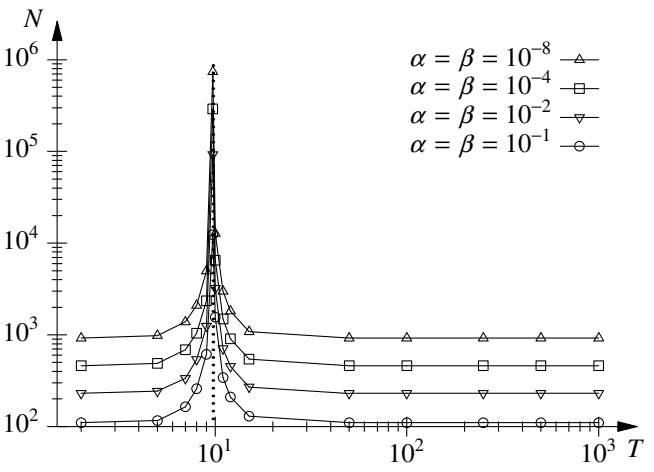
(a) Verification time as a function of state space size.



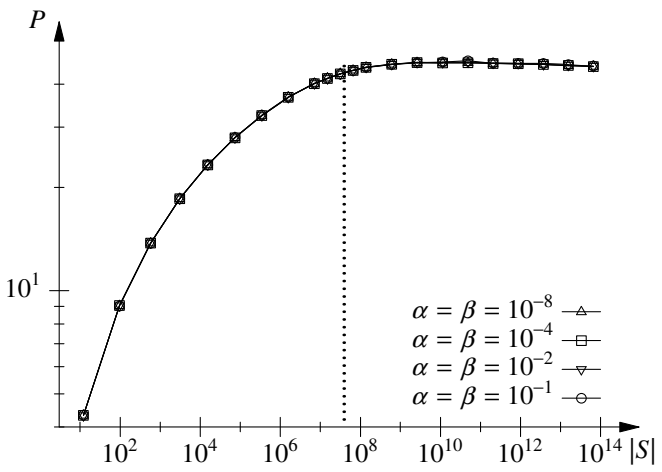
(b) Verification time as a function of formula time bound.



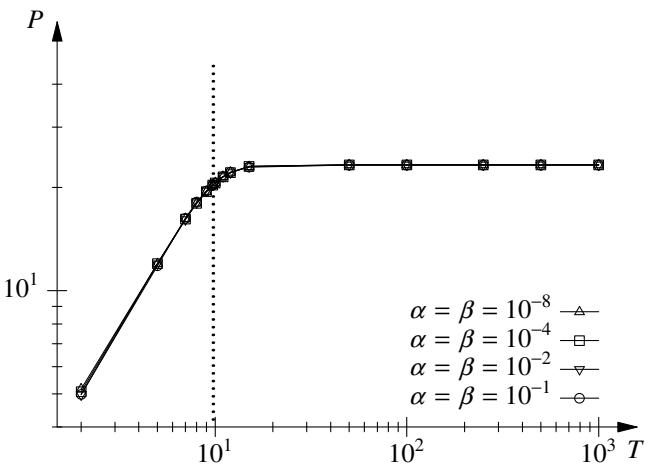
(c) Sample size as a function of state space size.



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



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



(f) Path length as a function of formula time bound.

Fig. 6. Empirical results for the symmetric polling system, with $T = 20$ to the left and $n = 10$ to the right. The dashed curves in (a) and (b) are for the numerical approach using the hybrid engine with $\epsilon = 10^{-6}$. The solid curves are for the statistical approach with $2\delta = 10^{-2}$, and α and β as indicated. The dotted lines mark a change in the truth value of the formula being verified.

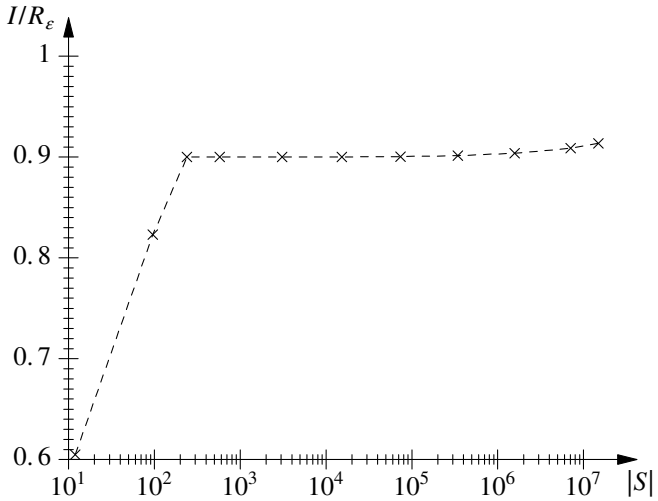


Fig. 7. Iterations as a fraction of R_ϵ for the symmetric polling system with $T = 20$.

solution method all coincide with p being close to or in the indifference region $(\theta - \delta, \theta + \delta)$.

The average sample size required to verify a formula of the form $\mathcal{P}_{\geqslant\theta}\left[\varphi\right]$ rapidly decreases as p gets further away from the threshold θ. We see this clearly in Figs. 5(c), 6(c), and 6(d) where the sample size is plotted against either the state space size or the time bound of the until formula. The empirical results also indicate that the sample size is proportional to the logarithm of the test strength, so that the required sample size doubles when α and β goes from 10^{-x} to 10^{-2x}. This is generally only true if p is not in the indifference region. From the approximation formula for $p = s$ in Table 1 (s refers to the quantity defined in (9), and not to a state), it follows that the numerator of $\tilde{N}_s$ is equal to $\left(\log(\alpha^{-1} - 1)\right)^2$ for $\alpha = \beta$, which means that N_s is approximately proportional to the *square* of $\log \alpha$ if p is in the indifference region. We can see an example of this in Fig. 6(d). The exceptionally sharp peak is for $T = 9.71$, which is almost in the centre of the indifference region.

Outside of the indifference region, where the sample size remains almost constant at a low level, the key performance factor instead becomes the length of sample paths. This factor depends on the exit rates of the CTMC and on the path formula φ. An upper bound on the expected sample path length for an until formula with time bound T, as noted in Sect. 2.2, is $O(q{\cdot}T)$, which is also the number of iterations for the numerical solution technique. We can see in Figs. 5(b) and 6(b) that the curves for the numerical and statistical solution techniques initially have very similar slope. For the tandem queueing network case study, the numerical approach benefits greatly from steady-state detection and outperforms the statistical approach as T increases. Figure 5(f) shows that the average sample path length for the statistical approach in this case is proportional to the time bound T across the entire range of time bounds considered.

For the tandem queueing network, the arrival rate for messages is $4n$, where n is the capacity of the queues. This means that in the worst case we can expect sample path lengths to be proportional to n. As n increases, the sample path length becomes the dominant performance factor because the sample size remains constant as seen in Fig. 5(c), meaning that verification time for the statistical approach becomes proportional to n. This is to be compared with the numerical approach, whose performance is linear in the size of the state space, which is quadratic in n. In the polling example, the arrival rate λ is inversely proportional to the number of polling stations n, while the other rates remain constant for all n. This explains the levelling-off of the expected sample path length plotted in Fig. 6(e).

Recall that we only need to generate as much of a sample path as is needed to determine the truth value of φ. For $\varphi = \Phi\ \mathcal{U}^{\leqslant T}\ \Psi$, we can stop if we reach a state satisfying $\neg\Phi \vee \Psi$ (cf. the CTMC $\mathcal{C}'$ constructed in the numerical approach in Sect. 2.1). The effect of this is seen most clearly for the polling case study as we increase the time bound. Once the path formula is satisfied the average length of the sample paths does not increase (Fig. 6(f)).

In general, we can say that the statistical solution technique scales better than the numerical solution technique with an increase in the size of the state space, but that steady-state detection can give the numerical approach an advantage when the time bound is high. It should also be clear from the results presented here that the sequential aspect of the statistical approach has a greater effect then the sequential component of the numerical approach. This means that the statistical is better at adapting to the difficulty of the verification problem at hand.

4.4 Trading Accuracy for Speed

With both solution methods, it is possible to adjust the accuracy of the result. For the statistical approach, we can control the parameters α, β, and δ so as to trade accuracy for efficiency. By setting these parameters high, we can get an answer quickly. We could then gradually tighten the error bounds and/or the indifference region to obtain higher accuracy. This approach is taken by Younes et al. [37], who modify the statistical solution method for verifying CSL formulae of the form $\mathcal{P}_{\geqslant\theta}\left[\varphi\right]$ without nested probabilistic operators so that it can be stopped at any time to return a result.

Figure 8 shows how the verification time for the symmetric polling system case study ($n = 10$, $T = 40$) depends on the strength of the test and the width of the indifference region. We can see that the verification time is inversely proportional both to the error bounds and the width of the indifference region, and that for some parameter values the numerical approach is faster while for others the statistical approach is the fastest. Using the statistical approach with error bounds $\alpha = \beta = 10^{-4}$ and half-width of the indifference region $\delta \approx 7 \cdot 10^{-5}$, for example, Fig. 8 demonstrates that we could obtain a verification result for the symmetric polling system problem in roughly the same time as is required by the numerical approach. For larger models, we would of course be

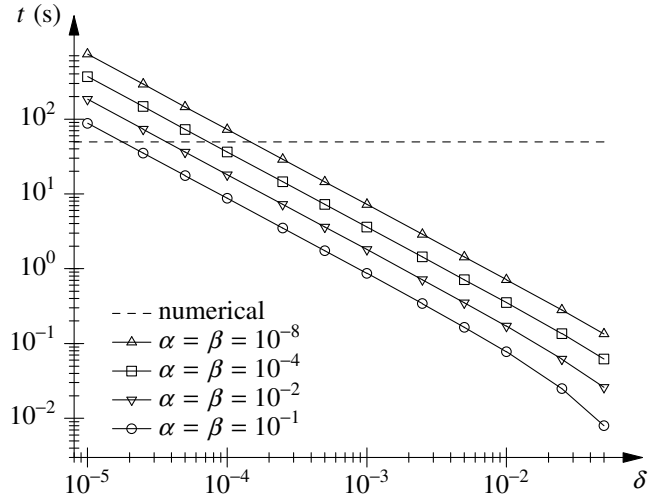


Fig. 8. Verification time for the symmetric polling system ($n = 10$ and $T = 40$) as a function of the half-width of the indifference region for different test strengths. The dashed line represents the verification time for the numerical approach.

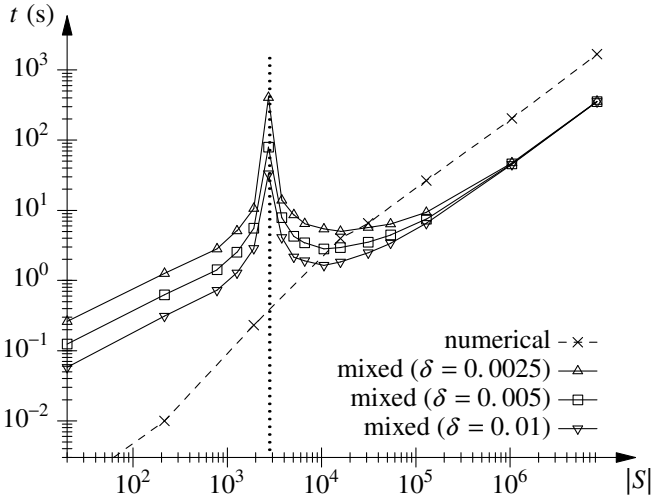


Fig. 9. Verification time as a function of state space size for the grid world example, with $T_1 = 100$ and $T_2 = 7$. The dotted line marks a change in the truth value of the formula being verified.

able to obtain even higher accuracy with the statistical approach if allowed as much time as needed by the numerical approach to solve the problem. The results in Fig. 8 also indicate that it is more costly to make the indifference region narrower than to reduce the error probabilities. For example, reducing the error probabilities from 10^{-2} to 10^{-4} roughly doubles the verification time, while the same reduction in the half-width of the indifference region leads to a hundredfold increase in the verification time.

We can adjust the accuracy for the numerical solution method by varying the parameter ϵ, but increasing or decreasing ϵ has very little effect on the verification time as shown by Reibman and Trivedi [26] and Malhotra et al. [23]. This means that the numerical solution method can provide very high accuracy without much of a performance degradation, while the statistical solution method is well suited if a quick answer with some uncertainty is more useful.

4.5 Mixing Solution Techniques

Finally, we present some results for the grid world case study, where the CSL property has nested probabilistic operators. We can see in Fig. 9 that the mixed approach shares performance characteristics with both approaches, outperforming the pure numerical solution method for larger state spaces. Verification times are shown for $\alpha = \beta = 10^{-2}$ and three different values of δ.

5 Discussion

In this paper, we have compared numerical and statistical solution techniques for probabilistic model checking both theoretically and empirically. The empirical evaluation has been carried out using case studies taken from the literature on performance evaluation and probabilistic model checking. We focused our attention on time-bounded properties as these are the type of properties most suited for statistical methods (the time-bound provides a natural limit for simulations).

The nature of CSL formulae allows us to use statistical hypothesis testing instead of *estimation* since we only need to know if the probability of a path formula holding is above or below some threshold. The use of sequential acceptance sampling allows the statistical approach to adapt to the difficulty of the problem: verifying a property $\mathcal{P}_{\geqslant\theta}\left[\varphi\right]$ in a state s takes more time if the true probability of φ holding in s is close to the threshold θ. This can give a statistical methods a distinct advantage over numerical approaches for model checking CSL formulae. Most previous assessments of statistical techniques (see, e.g., [29]) are based on parameter estimation problems, which are clearly harder in that they typically require a large number of samples. Our results show that the intuition from earlier studies does not necessarily carry over to CSL model checking.

Our results are otherwise in line with known complexity results for the two techniques. We show a linear complexity in the time-bound for both approaches and also confirm that statistical methods scale better with the size of the state space, but that high accuracy comes at a greater price than for numerical methods.

Sen et al. [28] have recently claimed to have developed a faster statistical solution technique than the one used in this paper, but their comparison is misleading. Their algorithm, unlike the one presented here, cannot guarantee any bound on the probability of accepting a false hypothesis, and instead reports a confidence (p-value [13]) in the computed result. The sample sizes reported by Sen et al. were selected *manually* based on prior empirical testing (K. Sen, personal communication, May 20, 2004), and there is in fact no fixed procedure by which they can determine the sample size required to achieve a certain p-value. The two algorithms are complementary rather than competing, and are useful under

disparate sets of assumptions. If we cannot generate sample paths on demand, then their algorithm (orthe variation described by Younes [36]) allows one to still reach conclusions regarding the behaviour of a system. If, however, we know the dynamics of a system well enough to enable simulation, then we are better off using the approach presented here as it gives full control over the probability of obtaining an incorrect result. It is of course not necessary to use the sequential probability ratio test as we have done. There exist other acceptance sampling tests, as discussed briefly in Sect. 2.2, that may outperform the sequential probability ratio test in some circumstances.

The case studies we considered in this paper were all CTMCs. To verify time-bounded properties of more complex models with general distributions, such as semi-Markov processes, more elaborate numerical techniques are required than those used for CTMC model checking (see e.g. [15, 20]). A statistical approach, on the other hand, would work just as well for semi-Markov processes (assuming, of course, that samples from the distributions used in the model can be generated in roughly the same amount of time as samples from the exponential distribution). Statistical solution techniques are also easier to parallelise, as each sample path can be generated independently. YMER does in fact include support for distributed acceptance sampling, and it is possible to get a performance improvement close to linear in the number of CPUs made available to YMER.

References

1. Alur, R. and Henzinger, T. A. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.
2. Aziz, A., Sanwal, K., Singhal, V., and Brayton, R. Verifying continuous time Markov chains. In Alur, R. and Henzinger, T. A., editors, *Proc. 8th International Conference on Computer Aided Verification*, volume 1102 of *LNCS*, pages 269–276. Springer, 1996.
3. Aziz, A., Sanwal, K., Singhal, V., and Brayton, R. Model-checking continuous-time Markov chains. *ACM Transactions on Computational Logic*, 1(1):162–170, 2000.
4. Bahar, R. I., Frohm, E. A., Gaona, C. M., Hachtel, G. D., Macii, E., Pardo, A., and Somenzi, F. Algebraic decision diagrams and their applications. In *Proc. IEEE/ACM International Conference on Computer-Aided Design*, pages 188–191. IEEE Computer Society Press, 1993.
5. Baier, C., Haverkort, B. R., Hermanns, H., and Katoen, J.-P. Model checking continuous-time Markov chains by transient analysis. In Emerson, E. A. and Sistla, A. P., editors, *Proc. 12th International Conference on Computer Aided Verification*, volume 1855 of *LNCS*, pages 358–372. Springer, 2000.
6. Baier, C., Haverkort, B. R., Hermanns, H., and Katoen, J.-P. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.
7. Bryant, R. E. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.
8. Buchholz, P. A new approach combining simulation and randomization for the analysis of large continuous time Markov chains. *ACM Transactions on Modeling and Computer Simulation*, 8(2):194–222, 1998.
9. Clarke, E. M., McMillan, K. L., Zhao, X., and Fujita, M. Spectral transforms for large Boolean functions with applications to technology mapping. In *Proc. 30th International Conference on Design Automation*, pages 54–60. ACM Press, 1993.
10. Fox, B. L. and Glynn, P. W. Computing Poisson probabilities. *Communications of the ACM*, 31(4):440–445, 1988.
11. Fujita, M., McGeer, P. C., and Yang, J. C.-Y. Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. *Formal Methods in System Design*, 10(2/3):149–169, 1997.
12. Hermanns, H., Meyer-Kayser, J., and Siegle, M. Multi terminal binary decision diagrams to represent and analyse continuous time Markov chains. In Plateau, B., Stewart, W. J., and Silva, M., editors, *Proc. 3rd International Workshop on the Numerical Solution of Markov Chains*, pages 188–207. Prensas Universitarias de Zaragoza, 1999.
13. Hogg, R. V. and Craig, A. T. *Introduction to Mathematical Statistics*. Macmillan Publishing Co., fourth edition, 1978.
14. Ibe, O. C. and Trivedi, K. S. Stochastic Petri net models of polling systems. *IEEE Journal on Selected Areas in Communications*, 8(9):1649–1657, 1990.
15. Infante López, G. G., Hermanns, H., and Katoen, J.-P. Beyond memoryless distributions: Model checking semi-Markov chains. In de Alfaro, L. and Gilmore, S., editors, *Proc. 1st Joint International PAPM-PROBMIV Workshop*, volume 2165 of *LNCS*, pages 57–70. Springer, 2001.
16. Jensen, A. Markoff chains as an aid in the study of Markoff processes. *Skandinavisk Aktuarietidskrift*, 36:87–91, 1953.
17. Katoen, J.-P., Kwiatkowska, M., Norman, G., and Parker, D. Faster and symbolic CTMC model checking. In de Alfaro, L. and Gilmore, S., editors, *Proc. 1st Joint International PAPM-PROBMIV Workshop*, volume 2165 of *LNCS*, pages 23–38. Springer, 2001.
18. Kwiatkowska, M., Norman, G., and Parker, D. PRISM: Probabilistic symbolic model checker. In Field, T., Harrison, P. G., Bradley, J., and Harder, U., editors, *Proc. 12th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, volume 2324 of *LNCS*, pages 200–204. Springer, 2002.
19. Kwiatkowska, M., Norman, G., and Parker, D. Probabilistic symbolic model checking with PRISM: A hybrid approach. *Software Tools for Technology Transfer*, 6(2):128–142, 2004.
20. Kwiatkowska, M., Norman, G., Segala, R., and Sproston, J. Verifying quantitative properties of continuous probabilistic timed automata. In Palamidessi, C., editor, *Proc. 11th International Conference on Concurrency Theory*, volume 1877 of *LNCS*, pages 123–137. Springer, 2000.
21. Lai, T. L. Nearly optimal squential tests of composite hypotheses. *The Annals of Statistics*, 16(2):856–886, 1988.
22. Lai, T. L. Sequential analysis: Some classical problems and new challenges. *Statistica Sinica*, 11(2):303–408, 2001.
23. Malhotra, M., Muppala, J. K., and Trivedi, K. S. Stiffness-tolerant methods for transient analysis of stiff Markov chains. *Microelectronics and Reliability*, 34(11):1825–1841, 1994.
24. Parker, D. *Implementation of Symbolic Model Checking for Probabilistic Systems*. PhD thesis, University of Birmingham, 2002.
25. PRISM web site: www.cs.bham.ac.uk/˜dxp/prism.
26. Reibman, A. and Trivedi, K. S. Numerical transient analysis of Markov models. *Computers & Operations Research*, 15(1):19–36, 1988.

27. Schwarz, G. Asymptotic shapes of Bayes sequential testing regions. *Annals of Mathematical Statistics*, 33(1):224–236, 1962.
28. Sen, K., Viswanathan, M., and Agha, G. Statistical model checking of black-box probabilistic systems. In Alur, R. and Peled, D. A., editors, *Proc. 16th International Conference on Computer Aided Verification*, volume 3114 of *LNCS*, pages 202–215. Springer, 2004.
29. Shanthikumar, J. G. and Sargent, R. G. A unifying view of hybrid simulation/analytic models and modeling. *Operations Research*, 31(6):1030–1052, 1983.
30. Stewart, W. J. A comparison of numerical techniques in Markov modeling. *Communications of the ACM*, 21(2):144–152, 1978.
31. Teichroew, D. and Lubin, J. F. Computer simulation—discussion of the techniques and comparison of languages. *Communications of the ACM*, 9(10):723–741, 1966.
32. Wald, A. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, 1945.
33. Wald, A. *Sequential Analysis*. John Wiley & Sons, 1947.
34. Wald, A. and Wolfowitz, J. Optimum character of the sequential probability ratio test. *Annals of Mathematical Statistics*, 19(3):326–339, 1948.
35. YMER web site: www.cs.cmu.edu/~lorens/ymer.html.
36. Younes, H. L. S. "Black-box" probabilistic verification. Technical Report CMU-CS-04-162, Carnegie Mellon University, 2004.
37. Younes, H. L. S., Musliner, D. J., and Simmons, R. G. A framework for planning in continuous-time stochastic domains. In Giunchiglia, E., Muscettola, N., and Nau, D. S., editors, *Proc. Thirteenth International Conference on Automated Planning and Scheduling*, pages 195–204. AAAI Press, 2003.
38. Younes, H. L. S. and Simmons, R. G. Probabilistic verification of discrete event systems using acceptance sampling. In Brinksma, E. and Larsen, K. G., editors, *Proc. 14th International Conference on Computer Aided Verification*, volume 2404 of *LNCS*, pages 223–235. Springer, 2002.