A Survey of Statistical Model Checking

GUL AGHA and KARL PALMSKOG, University of Illinois at Urbana-Champaign, USA

Interactive, distributed, and embedded systems often behave stochastically, for example, when inputs, message delays, or failures conform to a probability distribution. However, reasoning analytically about the behavior of complex stochastic systems is generally infeasible. While simulations of systems are commonly used in engineering practice, they have not traditionally been used to reason about formal specifications. Statistical model checking (SMC) addresses this weakness by using a simulation-based approach to reason about precise properties specified in a stochastic temporal logic. A specification for a communication system may state that within some time bound, the probability that the number of messages in a queue will be greater than 5 must be less than 0.01. Using SMC, executions of a stochastic system are first sampled, after which statistical techniques are applied to determine whether such a property holds. While the output of sample-based methods are not always correct, statistical inference can quantify the confidence in the result produced. In effect, SMC provides a more widely applicable and scalable alternative to analysis of properties of stochastic systems using numerical and symbolic methods. SMC techniques have been successfully applied to analyze systems with large state spaces in areas such as computer networking, security, and systems biology. In this article, we survey SMC algorithms, techniques, and tools, while emphasizing current limitations and tradeoffs between precision and scalability.

CCS Concepts: • **Mathematics of computing → Hypothesis testing and confidence interval computation**; **Statistical paradigms**; • **Theory of computation → Verification by model checking**;

Additional Key Words and Phrases: Statistical model checking, hypothesis testing, temporal logic, simulation, estimation

ACM Reference format:
Gul Agha and Karl Palmskog. 2018. A Survey of Statistical Model Checking. *ACM Trans. Model. Comput. Simul.* 28, 1, Article 6 (January 2018), 39 pages.
https://doi.org/10.1145/3158668

1 MOTIVATION

The behavior of interactive, distributed, and embedded systems is often stochastic in nature, for example, when input data, message delays, or failures conform to a probability distribution. In some cases, the behavior of individual actors in a system may also be probabilistic—in fact, the actors may be implementing a randomized algorithm [5].

Reasoning purely analytically about the behavior of complex stochastic systems is generally infeasible. Instead, numerical techniques that can provide ever-higher accuracy at low additional

This work is supported by the National Science Foundation, under Grants No. NSF CCF 14-38982 and No. NSF CCF 16-17401, and by AFOSR/AFRL Air Force Research Laboratory and the Air Force Office of Scientific Research under Agreement No. FA8750-11-2-0084 for the Assured Cloud Computing at the University of Illinois at Urbana-Champaign.
Authors' addresses: G. Agha and K. Palmskog, University of Illinois at Urbana-Champaign, Department of Computer Science, Urbana, IL, 61801; emails: {agha, palmskog}@illinois.edu.

https://doi.org/10.1145/3158668

cost may be used [14]. However, because of their computational and memory requirements, numerical techniques do not scale well. Moreover, numerical techniques require an explicit system description that includes state transition probabilities. Due to these limitations, numerical techniques are suitable only for those real-world systems that have a small state space and an explicit system description. In addition, for systems with complex dynamics, e.g., timed and hybrid systems, many basic problems have no efficient solution methods. In contrast, an approach based on statistical analysis relies only on the ability to obtain *samples* of properties. Such samples can be the result of *simulations* or a system execution. In fact, the statistical approach is applicable even when a system is a "black box" whose behavior is not fully understood or controllable [111]. Moreover, the statistical approach can be applied to systems with complex dynamics, as long as the behavior of such systems is stochastic and its execution traces are available.

Simulations and statistical inference over observations have a long history in engineering practice. In computer science, simulation and statistical inference have played a particularly important role in assessing the performance and reliability of computer systems and in analyzing networks [60, 119]. For example, in queuing theory, analysis of observed network traffic has identified a family of probability distributions (Poisson distributions) as common for the rate of message arrival. This has allowed the behavior of networks to be explored through simulations, which varies parameters such as throughput and delay.

However, simulations have traditionally not been used to reason about formal specifications—and not often for the behavior of concurrent software. Statistical model checking (SMC) addresses this weakness by using a *simulation-based* approach to reason about precise properties specified in a stochastic temporal logic. Properties specified in such a temporal logic may be simple: for example, that within some time bound, the probability that the number of messages in a queue will be greater than 5 is less than 0.01. This sort of property may also be informally stated and reasoned about in traditional queuing theory. Or the property may be more complex: for example, that in some bounded time period, if a node crashes, the probability that it will recover within 5000 steps is between 0.9 and 0.99. The temporal logics can get richer and allow logical deduction to be combined with statistical techniques.

In SMC, potential executions of a stochastic system are first sampled, after which statistical inference may be applied. Statistical inference comes in one of two flavors: (a) *hypothesis testing* is used to determine the extent to which observations "conform" to a given specification; and (b) *estimation* is used to determine likely values of parameters based on the assumption that the data is randomly drawn from a specified type of distribution. In either case, the result of an inference can used to evaluate a property specified in a stochastic temporal logic. Many such techniques used for statistical inference have been incorporated into SMC.

While the output from statistical inference techniques is not always correct, statistical inference nevertheless improves or weakens our confidence in a hypothesis. This confidence is based on the assumption that a probabilistic property (or its negation) holds (is true). For example, statistical inference may be used to conclude that, under the assumption that a coin is fair (has a 0.5 probability of producing a "Head"), the probability of a given finite sequence of observations is <0.01. At this point, we may doubt the hypothesis that the coin is fair.

The process of hypothesis testing follows the scientific method: we can never provide proof of the correctness of a hypothesis (theory or law). Instead, we conjecture that a model accurately represents the behavior of a system, and we then use this model to make predictions. If a model (theory) makes predictions that are consistent with the data, then our confidence in the model increases. If observations about the system "contradict" the predictions of the model, then we reject the model. Science is all about *conjectures and refutations* [99]. Of course, the process is more

complicated: the simplicity of a theory and the richness of its predictions are also considerations in how confident we are about a theory.

In the case of statistical reasoning, a conjectured model is probabilistic. We reject the model if the observations are sufficiently improbable based on the model. Estimation appears to be more *empiricist* in form [13]—but, in fact, it must also be based on a model. For example, we may base an estimate on the hypothesized model that the probabilistic behavior of a system follows some specific parameterized function (representing a family of probability distributions). Moreover, we often assume that the data has been randomly sampled. Based on these assumptions and the data in the samples, we estimate the probabilities of various possible values for the parameters. From these probabilities, we may also derive the probability of a given property holding under the assumptions.

This brings us to another basic question that must be answered before SMC can be applied: what should we assume the probabilistic model of the behavior of a system is, given that we need such a model as the basis of our statistical reasoning? One approach is as follows. Check if the property of interest (ignoring probabilistic operators) holds in a given execution (trace) of a system and treat it as a sample in a Bernoulli trial. Note that the Bernoulli distribution converges to normal distribution in the limit (in fact, fairly rapidly). Treat these samples as observations. Now assume that a given probabilistic specification does *not* hold (i.e., the negation of the specification holds). Under this assumption, compute the probability of the observations that have been made. If the observations are "sufficiently" improbable, then reject the conjecture that the system does not meet its specification. This is essentially the approach followed by Sen et al. [111].

Although simulations of stochastic systems (Monte Carlo methods) are well studied and have been applied extensively to real-world systems for many decades, SMC-related research has been active in the past 15 years, with important initial results appearing from 2002 to 2005 [46, 54, 111, 112, 126, 131]. In fact, some key SMC ideas may be traced a further decade back, in particular, to work in process algebra on establishing stochastic modal properties with arbitrary precision using hypothesis testing [74]. Note that the term *Statistical Model Checking* was first coined by Sen et al. [111].

SMC is a form of what has been called *stochastic model checking*, which also includes more precise and exhaustive numerical and symbolic methods. However, SMC can often elide the state space explosion problem that other model checking methods are prone to suffer from [61]. SMC techniques have been successfully applied to analyze large-scale systems in areas such as computer networking, security, and systems biology. This success has contributed to a recent proliferation of related specification formalisms, algorithms, and tools [6, 15, 25, 28, 34, 41, 65, 94, 109, 113, 129].

System analysis using SMC techniques generally follows the workflow in Figure 1, where key components are shown as boxes with rounded corners. Although a simulation driver, ③ in the figure, is a core requirement for performing SMC, the low-level details of how a simulation is performed are generally not of concern to SMC algorithms: improvements in discrete-event simulation of stochastic systems can immediately benefit SMC-based analysis. Conversely, SMC algorithms can usually be easily integrated with existing discrete-event simulators to provide extended analysis capabilities to such frameworks. The specifics of simulation engines are beyond the scope of this article. Moreover, note that the samples could also be obtained directly from the execution of the system itself.

Outline of the Paper. Section 2 introduces some background material, including stochastic processes and discrete event systems, stochastic logics, types of verification techniques, and types of errors. Section 3 describes strategies for sampling, testing, and estimation of parameters,

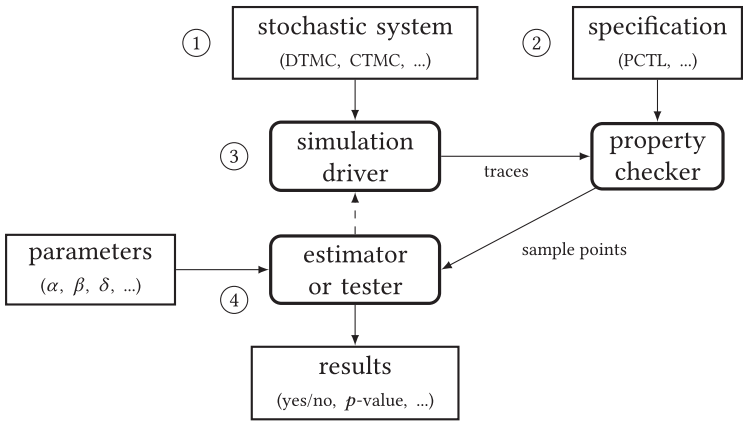


Fig. 1. SMC workflow.

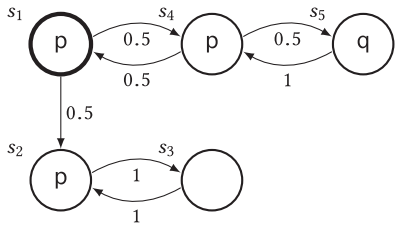
including specific methods that pertain to black box systems, Bayesian inference, and rarely realized properties. Note that our treatment of statistical inference will not be detailed; foundations and more details can be found in a textbook, such as that by Trivedi [119]. In Section 4, we turn our attention to two specific classes of more complex properties—namely, those involving nested formulas and those involving the "unbounded until" operator. Section 5 describes three specific classes of systems that require specialized logics and algorithms: systems that have (unquantified) nondeterminism as well as probabilistic behavior, timed systems, and hybrid systems. Section 6 lists key areas to which SMC has been applied. In Section 7, we provide some details about specific tools and their use, along with suggested heuristics for choosing a tool. Finally, Section 8 concludes with a discussion of the limitations and applicability of SMC as well as some directions for future research.

2 BACKGROUND

SMC requires inputting an executable system and a specification. In this section, we discuss some varieties of these inputs (① in Figure 1), specifications ②, and the two forms of SMC: hypothesis testing and estimation ④. We then provide some background relevant to the problem of reducing the chances of errors in judgment about hypotheses (the results).

Numerical and symbolic analysis methods for stochastic systems generally require explicit, finite-state ("white box") models, usually in the form memoryless (Markovian) stochastic processes. However, the stochastic model checking problem with respect to a specification in a stochastic logic is well defined even without such an explicit model. SMC only requires that the system to be analyzed can be simulated using discrete-event simulation and that there is a *probability space* on executions of the system. SMC can even be applied on infinite-state and implicit ("black box") stochastic systems where measures on the probability space are not completely known [111].

For example, SMC is directly applicable to systems whose behavior can modeled by a *Discrete Time Markov Chain* (DTMC) or *Continuous Time Markov Chain* (CTMC) [131], by a *(Generalized) Semi Markov Process* (GSMP) [111], or by a *Generalized Stochastic Petri Net* (GSPN) [15]. More recently, SMC analysis has been extended to *timed and hybrid systems* with stochastic behavior [39, 41], as well as *Markov Decision Processes* (MDPs), where some purely nondeterministic choices (i.e., choices without a probability assignment) may also be made by a system [79].



M	s_1	s_2	s_3	s_4	s_5
s_1	0	0.5	0	0.5	0
s_2	0	0	1	0	0
s_3	0	1	0	0	0
s_4	0.5	0	0	0	0.5
s_5	0	0	0	1	0

Fig. 2. A discrete time Markov chain and its transition matrix.

2.1 Stochastic Processes and Discrete Event Systems

A *stochastic process* is a collection of random variables indexed by some set, whose elements are traditionally interpreted as points in time. Each random variable assumes values from a *state space*, and processes are said to occupy some state from this space at any time.

While conventional analysis techniques require an explicit stochastic process, SMC is applicable to any process, implicit or explicit, that can be considered a *stochastic discrete event system*, which occupies a state for some duration before an event causes a transition to occur [111, 126]. As alluded to above, SMC works as long as sequences of states (paths) generated by the system are *measurable*. In particular, a DTMC represents a stochastic process indexed over the (non-negative) integers, which can be considered as a discrete event system evolving over discrete time.

In this article, we will sometimes use fully defined finite-state DTMCs as examples to concretize the presentation of SMC concepts and algorithms. However, ideas will typically carry over to other types of stochastic systems, such as CTMCs, with only minor differences.

Definition 2.1. Let AP be a finite set of atomic propositions. A DTMC is a tuple $\mathcal{M} = (S, s_i, M, L)$ where S is a finite set of *states*, $s_i \in S$ is the *initial state*, $M : S \times S \to [0, 1]$ is a *transition probability function* s.t. for all $s \in S$, $\sum_{s' \in S} M(s, s') = 1$, and $L : S \to 2^{AP}$ is a *labeling function* associating states with their respective true atomic propositions.

In the context of a DTMC $\mathcal{M}$, *paths* are infinite sequences $s_1, s_2, \ldots$ of elements of S. *Traces*, or *path prefixes*, are finite non-empty prefixes of such sequences. For some purposes, e.g., symbolic computation, it can be beneficial to view M in the DTMC $\mathcal{M}$ as a *matrix* rather than a function. To do so, some order relation on states must be fixed. $\mathcal{M}$ is at any moment of time in one of its states s, from which it makes a transition to some state s' according to the probabilities given by M—an operation that takes one unit of time.

Example 2.2. Consider a DTMC $\mathcal{M}$ with five states represented pictorially in Figure 2 (Note that this example is due to Roohi and Viswanathan [105]). There are two atomic propositions p and q, which hold as indicated in the figure, e.g., p is true in the state s_1. As emphasized by the circle's thickness, s_1 is the initial state. By the definition of the transition matrix, the probability that $\mathcal{M}$ traverses the path prefix s_1, s_2, s_3, s_2 is $M(s_1, s_2) \times M(s_2, s_3) \times M(s_3, s_2) = 0.5 \times 1 \times 1 = 0.5$.

2.2 Stochastic Logics

Functional, *qualitative* properties of systems such as state invariants are traditionally expressed in temporal logics over (non-stochastic) transition system models. One classic such logic is the branching-time Computational Tree Logic (CTL) [33]. In contrast, logics for stochastic systems must permit *quantitative* reasoning about probabilities and time.

As an example, we describe Probabilistic Computational Tree Logic (PCTL), which extends CTL with operators for probability and time bounds to allow quantitative property specification [48].

Table 1. PCTL Syntax

$\phi ::= \top \mid a \mid \neg\phi \mid \phi \wedge \phi' \mid \mathsf{P}_{\geq\theta}(\psi)$	$a \in AP$	atomic proposition
$\psi ::= \phi \mid \mathsf{X}\,\phi \mid \phi\,\mathsf{U}^{\leq t}\,\phi' \mid \phi\,\mathsf{U}\,\phi'$	$\theta \in [0,1]$	probability bound
	$t \in \mathbb{Z}^{\geq 0}$	time bound

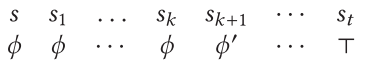


Fig. 3. Path where $\phi\,\mathsf{U}^{\leq t}\,\phi'$ holds.

As in CTL, formulas in PCTL quantify over either states or paths, with ϕ used to denote the former kind of formula and ψ to denote the latter. Table 1 gives the syntax definition of PCTL formulas. We use only the $\geq$ (greater than or equal to) comparison operator on probabilities. Since the probabilities we deal with are continuous, there is no meaningful distinction between $>$ and $\geq$ (or between $<$ and $\leq$). Moreover, $<$ can be defined in terms of $\geq$ and $\neg$.

The semantics of a PCTL formula is defined with respect to a DTMC $\mathcal{M}$ and a state $s \in S$. More specifically, the formula $\top$ is always true in s, and a is true when a is in the set of labels associated with s, i.e., when $a \in L(s)$. Negation and conjunction have the usual meanings. Intuitively, the formula $\mathsf{P}_{\geq\theta}(\psi)$ is true in s when the probability that ψ holds on paths starting from s is greater than or equal to θ.

For a path $\pi = s, s_1, \ldots$, the formula ϕ is true for π if ϕ holds in s, and $\mathsf{X}\,\phi$ is true for π if ϕ holds in the next state after s, i.e., in s_1. As illustrated in Figure 3, the formula $\phi\,\mathsf{U}^{\leq t}\,\phi'$ is true for a path $\pi = s, s_1, \ldots, s_t, \ldots$ whenever there is a state s_{k+1} with $k+1 \leq t$ in π where ϕ' holds, and for all states up to and including s_k, ϕ holds. The corresponding formula without the condition on t, $\phi\,\mathsf{U}\,\phi'$, loosens the requirement for ϕ' to be true within an *a priori* fixed number of steps in a path. In particular, the formula $\top\,\mathsf{U}\,\phi'$ holds if ϕ' is true at some undetermined point along the path. In other words, $\top\,\mathsf{U}\,\phi'$ encodes the property that an event "eventually" occurs—a property often specified for systems using non-stochastic temporal logics.

Example 2.3. Consider a queue system with stochastic task arrivals modeled as a DTMC, with $\mathtt{pending}, \mathtt{done} \in AP$, `pending` true for states where a received task is being processed, and `done` true for states where processing a task is completed. Then, the PCTL formula $\mathsf{P}_{\geq 0.98}(\mathtt{pending}\,\mathsf{U}^{\leq 10}\,\mathtt{done})$ holds in paths π starting from a given state s if, with probability 0.98 or more, `pending` holds for all states along the path up to a state s' where `done` holds, with a distance (i.e., number of units of time) in π from s to s' of at most 10. With the condition ≤ 10 omitted for the until operator U in the formula, there is no upper bound on the number of units of time before `done` must be true; as discussed in Section 4.2, such *unbounded until* formulas are particularly difficult to verify algorithmically for stochastic systems in general and thus for DTMCs.

There are several variants and extensions of PCTL. Our PCTL syntax notably omits a long-run (steady-state) operator. Some tools such as PRISM [69] allow users to specify that they want to obtain (an estimate of) the probability for a path formula rather than a comparison of that probability to a given threshold. However, since some SMC algorithms do not compute an explicit estimate of the probability, users will not always be able to obtain such probabilities from SMC tools.

The literature contains many stochastic logics, which allow different kinds of properties to be expressed, defined over a variety of stochastic systems. For example, *Continuous Stochastic Logic* (CSL) [10, 11] extends CTL to capture quantitative properties of CTMCs, and *Quantitative*

Temporal Expressions (QuaTEx), which are defined over probabilistic rewrite systems, allow properties related to expected values of any expression to be queried [5].

2.3 Verification Techniques

Assume we are given some process that defines a stochastic discrete event system, along with a specification of a state property in a stochastic logic. Does the the property hold for some specific state in the state space? This is the stochastic analogue to the classic *model checking problem* for temporal logics such as CTL for which many algorithms and tools for automated verification have been developed [33].

For example, if we are given a DTMC $\mathcal{M}$, and a PCTL specification $\mathsf{P}_{\geq\theta}(\psi)$, where ψ is a path formula, does the specification hold in the initial state s_i of $\mathcal{M}$? This stochastic model checking problem on DTMCs (and its analogue for CSL specifications on CTMCs) can in principle be solved using numeric techniques. In such algorithms, the probability p of (the measure of) ψ in $\mathcal{M}$ for s_i is computed up to some precision and then compared with θ. One established approach is to construct a new DTMC based on $\mathcal{M}$, where states that immediately make the specification true when reached are made self-looping with probability 1, i.e., *absorbing*. From this DTMC, a vector of probabilities representing the measure for each state can be obtained via an infinite sum [130]. Arguably, however, such methods solve a *harder* problem than the original one.

Numeric-symbolic manipulations of a Markov chain can be made more efficient by operating on optimized representations, e.g., binary decision diagrams and sparse matrices and arrays. The potential for sparse representations for transition matrices is highlighted in the example DTMC in Figure 2, where M has at most two entries in any row or column. Depending on the Markov chain structure, analysis on one particular representation may be faster than on others, but require more memory—limiting the state space that can be explored even when using tools that employ a plethora of techniques to improve efficiency. For example, Younes et al. concluded in 2011 that the numeric approach was then feasible only when the state space of the system under analysis was of size 10^9 or less [129].

In contrast, using SMC, a system is executed repeatedly using discrete event simulation (Monte Carlo experiments) without the need for an explicit representation. The path formula ψ in $\mathsf{P}_{\geq\theta}(\psi)$ is checked on each path prefix (trace) generated by the simulation. A checked path prefix provides an observation, or sample point, of a Bernoulli random variable X, which is 1 if ψ holds and 0 otherwise. Thus, SMC may be used for hypothesis testing or for probability estimation.

Hypothesis Testing. Suppose H_0 and H_1 are two competing and complementary hypotheses: H_0 is the hypothesis that the probability measure p of ψ for s_i is greater than or equal to θ ($H_0 : p \geq \theta$), while the alternate hypothesis H_1 states that p is less than θ ($H_1 : p < \theta$). Let H_0 be the "null hypothesis": what we would accept until proved otherwise. Now, if we claim H_0 is false, while H_0 is true, then we make what is called a Type I error. In hypothesis testing, we use a statistics to estimate the probability of the observations assuming H_0 is true. Call this probability $1 - \alpha$. Thus, α is the probability of Type I error if we reject H_0 based on these observations. We can also use statistics to estimate the probability of the observations assuming H_1 is true. Now if we decide H_0 is true (reject H_1) while H_1 is true, we make what is called a Type II error. The probability of Type II error is denoted by β.

Observe that what we are really interested in is the probability that H_0 is true (or false). However, as we noted earlier, there is no mathematical way to get at this probability: we can only rigorously estimate the probability of observations based on a hypothesis, not the probability of the hypothesis itself being true based on the observations. Theories about the probability that

Table 2. Schema of Errors in a Hypothetical Statistical Test with Parameters α and β

	Decision	
Truth	*accept* H_0, *reject* H_1	*reject* H_0, *accept* H_1
$p \geq \theta$: H_0 *true*, H_1 *false*	correct ($>1-\alpha$)	type I error ($\leq\alpha$)
$p < \theta$: H_0 *false*, H_1 *true*	type II error ($\leq\beta$)	correct ($>1-\beta$)

The conditions inside parentheses are on the probability for the given outcome.

a hypothesis is true are *beliefs*, which may be informed by estimates based on observations. The *Bayesian method* provides what appears to be a reasonable way of revising beliefs about hypothesis based on observations (see Section 3.4).

Estimation. Assuming that a given set of observations represents a random sample, and that the behavior of the underlying process follows a specified family of probability distributions, observations may be used to estimate the parameters of the distribution. For example, if we assume that the system being tested follows a Bernoulli process, then we may estimate a probability measure p characterizing the distribution. The estimated probability measure may be compared to θ to reach a conclusion that $p \geq \theta$ (or $p < \theta$) is likely. The probability estimate may also be provided to the user along with the result.

2.4 Error Control for Hypothesis Testing

Given that a sample size is always finite, decisions about hypotheses (and estimates) using a statistical test run the risk of either a *Type I* or *Type II error*. Typically, tests take parameters α and β, for $\alpha + \beta \leq 1$, that bound the probability of a Type I error and a Type II error, respectively.

Consider as above a Bernoulli random variable X that is observed after simulating a DTMC $\mathcal{M}$ and checking ψ in traces for the PCTL specification $\mathrm{P}_{\geq\theta}(\psi)$. Suppose X has parameter p, i.e., $\Pr[X = 1] = p$ and $\Pr[X = 0] = 1 - p$. The possible outcomes of a hypothetical statistical test are then as shown in Table 2.

Let L_p be the probability of accepting H_0 in a hypothetical statistical test. To provide the desired error bounds, the test has to guarantee that whenever $p < \theta$, we have $L_p \leq \beta$, and additionally that whenever $p \geq \theta$, we have $L_p > 1 - \alpha$. As shown, e.g., by Younes et al. [130], these requirements on the test can be illustrated by plotting p against L_p as in Figure 4(a). However, this graph reveals the precarious case when $p \approx \theta$, where $L_p > 1 - \alpha$ holds, but where also $L_p \leq \beta$ holds for p infinitesimally smaller than θ. Ensuring both conditions are obeyed by the test requires either sampling more extensively or setting $\beta = 1 - \alpha$ [132]. The former may be infeasible, and the latter implies that error probabilities for Type I and Type II errors are not independently controllable.

To enable construction of practical and flexible tests, an often-used solution is to define an *indifference region* around the bound θ. When p is contained in this region, tests provide no error-related guarantees. Typically, tests take a parameter δ, which is the half-width of the indifference interval; Figure 4(b) plots p against L_p for such a test. Note that the test obeys the bounds indicated in Figure 4(a) everywhere but inside the indifference region. To take the indifference region into account, the original hypothesis $H_0 : p \geq \theta$ is changed to $H_0 : p \geq \theta + \delta$, and its converse $H_1 : p < \theta$ is changed to $H_1 : p \leq \theta - \delta$. Consequently, both H_0 and H_1 are false when p is inside the indifference region. The resulting schema of errors is shown in Table 3.

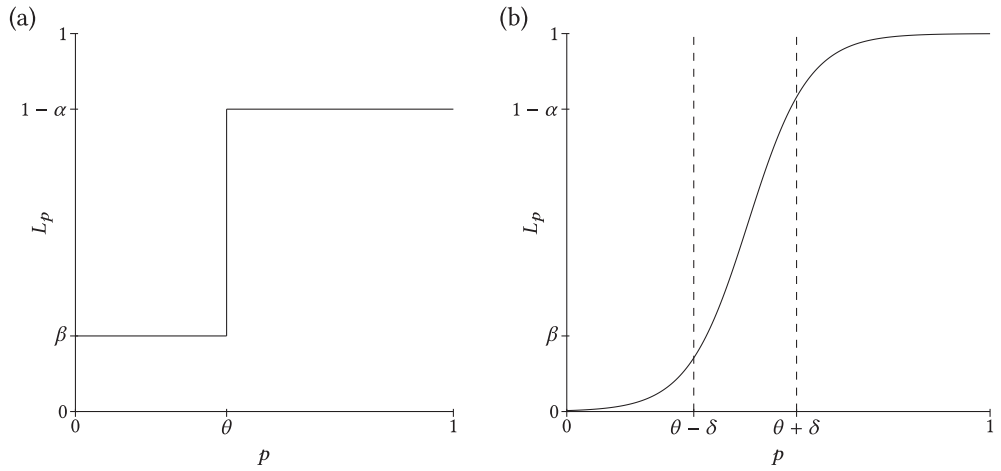


Fig. 4. Probability L_p of accepting the hypothesis $H_0 : p \geq \theta$, as function of p, with and without an indifference region.

Table 3. Schema of Errors in a Practical Statistical Test with Parameters α, β, and δ

	Decision	
Truth	*accept* H_0, *reject* H_1	*reject* H_0, *accept* H_1
$p \geq \theta + \delta$: H_0 *true*, H_1 *false*	correct ($>1-\alpha$)	type I error ($\leq\alpha$)
$\theta - \delta < p < \theta + \delta$: H_0, H_1 *false*	indifferent	indifferent
$p \leq \theta - \delta$: H_0 *false*, H_1 *true*	type II error ($\leq\beta$)	correct ($>1-\beta$)

The conditions inside parentheses are on the probability for the given outcome.

2.5 Error Control in Estimation

Early work on estimation-based SMC introduced sample-based probability estimation based on Chernoff bounds [54]. Applications of Chernoff bounds can limit the deviation of a sampled expectancy from a true expectancy for sums of Bernoulli random variables. More precisely, given an error probability bound α and precision ϵ, an algorithm can compute a probability p' such that $|p' - p| \leq \epsilon$ with confidence α. In other words, the probability that $|p' - p| \leq \epsilon$ is greater than or equal to $1 - \alpha$. To make a judgment about the threshold comparison with the desired confidence, the implied interval that contains p, i.e., $[p' - \epsilon, p' + \epsilon]$, must be taken into account. In terms of the hypotheses defined for statistical testing, we obtain p' and conclude that H_0 holds if $p' - \epsilon > \theta$ and that H_1 holds if $p' + \epsilon < \theta$.

More general estimation methods that establish confidence intervals, based on the Student's t-distribution (and its normal distribution approximation), have also been implemented in SMC tools [100], and yield similar guarantees but with different tradeoffs on the sample size and confidence parameters, as explained in more detail in Section 3.

2.6 Mitigating Risk

The above treatment may lead one to think that there is a symmetry between the cost of Type I errors and the cost of Type II errors. In general, such a conclusion would be misleading. For example, a specification may say something about a Quality of Service (QoS) parameter. If SMC

erroneously suggests that the specification is violated, then the erroneous conclusion may lead to an unnecessary provisioning of extra resources to improve the robustness of the system. The price of such an error is the cost of over-engineering. On the other hand, if SMC erroneously suggests that the specification will be satisfied, then the error in judgment may lead to a violation of a specification during deployment—something that may be very expensive.

Statistical decision theory can be used to account for risk. It can also be used to determine how large a sample to take and when to continue sampling. The classic work of Wald [122] provides an excellent treatment of the statistical decision functions.

3 SAMPLING, TESTING, AND ESTIMATION

Hypothesis testing can be performed on a set of samples that has been collected. If we can generate samples on demand, then the size of this sample may be predetermined—this is called *fixed-size* sampling. Alternatively, samples may be collected and statistical tests performed as they are collected, until a decision is reached—this is called *sequential testing* [121]. For fixed-size sampling, the number of sample points must be precomputed, based on the risk functions associated with error probabilities α and β [122]. In sequential testing, the key idea is to repeatedly generate sample points until enough information has been obtained to return an answer with the required confidence; as a consequence, the number of observations required need not be known beforehand.

We first discuss sampling strategies where we can control the sample and then discuss "black box" systems, where we are given a fixed set of traces up front as the basis for our analysis. We then discuss the problem of estimating probabilities, Bayesian analysis, and finally sampling and reasoning about rare events.

To reason about a system's behavior, we need to know that the system follows a well-defined stochastic model for which properties can be expressed in a specific logic. As in Section 2, we assume for purposes of exposition a DTMC $\mathcal{M}$, such that we can define a PCTL specification $\mathrm{P}_{\geq\theta}(\psi)$, a probability p for ψ in the initial state of $\mathcal{M}$, an indifference interval half-width δ, antagonistic test hypotheses $H_0 : p \geq p_0$ and $H_1 : p \leq p_1$ where $p_0 = \theta + \delta$ and $p_1 = \theta - \delta$, and error probability parameters α and β. Finally, we let X_i range over Bernoulli random variables associated with simulations of $\mathcal{M}$, and x_i range over observations of such variables, i.e., $x_i = 0$ or $x_i = 1$.

3.1 Sampling and Tests

There are two sampling strategies: fixed sample sizes or incremental testing. These sampling strategies involve a tradeoff in terms of the cost of sampling and the reliability of tests.

Fixed-Size Sample Testing. A strategy for statistical testing with fixed sample size is usually called a single sampling plan (SSP). An SSP can be completely defined by its number of samples n and a constant c, where H_0 is accepted when $\sum_{i=1}^{n} x_i > c$ and H_1 is accepted otherwise. This schema leaves room for many variations; for example, Sen et al. proposed to use SSPs for SMC where the hypothesis H_0 is accepted when $\sum_{i=1}^{n} x_i/n \geq \theta$ [112]. In general, there is no analytical expression for n that achieves a given confidence and indifference interval for an SSP. Adcock provides an overview of different methods to determine n for several distributions, including the here pertinent binomial distribution [2].

SSPs are oblivious as to whether conclusions can be made on-the-fly as simulations are performed. For example, it may be the case that $\sum_{i=1}^{m} x_i > c$ for some m such that $m < n$, allowing H_0 to be concluded immediately with the required confidence, but nevertheless the remaining $n - m$

planned observations are still carried out. Sequential testing can be viewed as an attempt to limit the expected number of sample points by allowing such on-the-fly decisions.

Sequential Testing. In *sequential testing*, we do not start with a set of samples or even fix a sample size in advance. Instead, hypothesis testing is done as more samples are collected [121]. Note that any SSP can in fact be turned into a sequential test by obtaining sample points one-at-a-time and checking the appropriate acceptance conditions after each observation. Sequential sampling can often result in significantly smaller sample sizes. On the other hand, sequential testing has its own limitations. First, there is a cost associated with setting up an experiment and gathering samples. Thus sequential testing may involve collecting a bundle of samples and then running a test. The size of the bundle will depend on the relevant cost and risk functions. Second, one must decide what properties to test for before samples are taken. If one wishes to subsequently test for other properties, then the sample taken may be inadequate. In addition, sequential sampling is not possible when new sample points cannot be generated on demand.

Sequential Probability Ratio Test (SPRT). In early work on SMC, Younes and Simmons advocate the use of the sequential probability ratio test (SPRT) on account of its average-case efficiency [131]. The SPRT was originally proposed by Wald in the 1940s and intuitively maintains a Bayesian *likelihood ratio* that expresses how the hypotheses fare against each other based on all hitherto observed outcomes [123]. After m observations have been made in the SPRT, i.e., after $x_1, \ldots, x_m$ have been obtained, let $d_m = \sum_{i=1}^{m} x_i$ and define

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

For example, suppose $m = 2$, with $x_1 = 1$ and $x_2 = 0$; then $\Pr[X_1 = x_1 \,|\, p = p_1] = p_1$, $\Pr[X_2 = x_2 \,|\, p = p_1] = 1 - p_1$, and symmetrically for p_0, whence $f_m = \frac{p_1(1-p_1)}{p_0(1-p_0)}$. In the general case, after f_m has been calculated using Equation (1), SPRT accepts H_0 if $f_m \leq \beta/(1-\alpha)$, accepts H_1 if $f_m \geq (1-\beta)/\alpha$, or moves on to the next sample point if neither condition holds.

SPRT has an impressive optimality property of minimizing the expected sample size under certain strict assumptions. However, these assumptions may not be true in practice. For example, if the actual value of a parameter lies within an "indifference region" between two alternate hypotheses, Wald's SPRT performs worse than the classic fixed sample size test. A number of other analysis methods rules have been developed to address these weaknesses. Lai gives a comprehensive general overview of sequential analysis and its applications [73], while Tartakovsky et al. provide a detailed treatment of sequential testing and stopping rules [117].

Undecided Results. Conventional hypothesis tests establish bounds on the probability p of a path property. Nevertheless, as a consequence of the use of an indifference interval, there are no error-related guarantees associated with the result when p falls within δ of either side of the threshold θ. In effect, a test can return any convenient answer in such cases with no obvious indication, a fact that an SMC user must be aware of.

In the context of testing black box systems, explained in more detail below, Sen et al. first introduced the idea that SMC algorithms can return an undecided result [111]. Their idea is to first test the null hypothesis—and only if it fails, test the alternate. On the other hand, Younes proposes to simultaneously test *two* pairs of hypotheses against each other and draw a conclusion based on both outcomes. Permitting such explicit undecided results in hypothesis tests can provide unconditional error-related guarantees for any decided result [128].

More precisely, in the first test $H_0^{\perp} : p \geq \theta$ is evaluated against $H_1^{\perp} : p \leq \theta - \delta$, and in the second test $H_0^{\top} : p \geq \theta + \delta$ is evaluated against $H_1^{\top} : p \leq \theta$. The formula $\mathrm{P}_{\geq\theta}(\psi)$ is deemed to be true if

Table 4. Schema of Errors in a Statistical Test that Allows Undecided Results with Parameters α, β, δ, and γ

	Decision		
Truth	*accept* $H_0^{\perp}, H_0^{\top}$	*accept* $H_0^{\perp\top}, H_1^{\perp\top}$	*accept* $H_1^{\perp}, H_1^{\top}$
$p \geq \theta$	correct ($>1-\alpha-\gamma$)	undecided	type I error ($\leq\alpha+\gamma$)
$p < \theta$	type II error ($\leq\beta+\gamma$)	undecided	correct ($>1-\beta-\gamma$)

The conditions inside parentheses are on the probability for the given outcome. "*accept* $H_i^{\perp\top}$" means accepting precisely one of $H_i^{\perp}$ and $H_i^{\top}$.

both $H_0^{\perp}$ and $H_0^{\top}$ are accepted, and deemed false if both $H_1^{\perp}$ and $H_1^{\top}$ are accepted. Otherwise, an undecided result is returned.

To perform the tandem tests, an additional parameter γ must be provided along with α, β, and δ. Intuitively, γ is the upper bound on the probability of an undecided outcome. Consequently, when $p \geq \theta$, a correct result is returned with a probability greater than $1-\alpha-\gamma$, and when $p < \theta$, a correct result is returned with a probability greater than $1-\beta-\gamma$. Table 4 shows the resulting schema of errors for statistical tests based on the tandem tests approach.

3.2 Samples from Black Box Systems

SMC loosens the requirement of numerical techniques for an explicit system model. However, the above discussion of sampling assumes that traces can be obtained on command using discrete-event simulation. Not all stochastic systems are controllable, in the sense that it is possible to randomly generate a trace starting from some state. For example, we may wish to analyze an already deployed system, and only have access to a fixed set of "historical" traces obtained during its execution. A pertinent question is how well these traces support the conclusion that some property in a stochastic logic holds in a state in such otherwise unknown "black box" systems. Clearly, the methods for hypothesis testing and estimation described above cannot be applied exactly as given, since there can be no *a priori* known error-related guarantees with the sample size fixed in advance. In particular, a sequential hypothesis test may run out of sample points long before the specified error bounds can be established.

Sen et al. first proposed an algorithm for SMC on samples from uncontrollable black box systems [111]. Along with a decision on whether a formula $\phi = \mathrm{P}_{\geq\theta}(\psi)$ holds for the system, the algorithm provides a value in the interval $[0, 1]$ that intuitively represents the confidence in the decision. In statistics jargon, the number is a p-value, with outcomes near 0 indicating high confidence and outcomes near 1 low confidence. Younes extends the work of Sen et al. by providing a closely related algorithm that improves the initial algorithm in several ways [126].

Both the algorithm of Sen et al. and that of Younes are similar to an ordinary SSP in that they define a constant c, such that the hypothesis $H_0 : p \geq \theta$ is accepted when $\sum_{i=1}^{n} x_i > c$, for n the sample size and x_i the respective outcomes of checking ψ. For Sen et al., the choice of c is based on the normal approximation of the binomial distribution and defined as $\lceil n\theta \rceil - 1$. In contrast, in the algorithm of Younes, c is chosen so that the cumulative distribution function for the binomial distribution with parameters c, n, and θ is near 0.5. More precisely, c is chosen so that the sum $\sum_{i=1}^{c} \binom{n}{i} \theta^i (1-\theta)^{n-i}$ is as close to 0.5 as it can be. As a consequence, the algorithm of Younes always accepts the hypothesis with the smallest p-value. As discussed earlier, this may not be desirable if the risk associated with the alternate hypothesis is not symmetric. Additionally, the algorithm of Younes is less conservative in the calculation of p-value for conjunctive formulas,

and does not assume that traces are always long enough to enable evaluation of given bounded until formulas.

There is no general algorithm that can determine the sample size required to achieve a certain p-value for an arbitrary system. However, for particular families of probability distributions, techniques exist to estimate the sample size needed for a given confidence level [2, 83, 116].

3.3 Estimation Methods

Estimation (sometimes called *approximation*) methods for SMC based on the application of Chernoff bounds were first suggested by Lassaigne and Peyronnet, and later explored in detail by Hérault et al. in the context of the implementation of an SMC tool [54, 75]. With m sample points obtained, the proposed approximation of the underlying parameter p of the random variables is defined as $p' = \sum_{i=1}^{m} x_i/m$. Application of a bound proved by Hoeffding then yields $\Pr(|p' - p| \geq \epsilon) \geq 2e^{-2m\epsilon^2}$ [57]. Letting $m = \lceil \ln(2\alpha)/(2\epsilon^2) \rceil$, we obtain $\Pr(|p' - p| \leq \epsilon) \geq 1 - \alpha$, as required. The needed sample size n when using this method is thus uniquely determined by the confidence α and the precision ϵ.

The SMC component of the model checker PRISM implements two estimation methods in addition to the one based on Chernoff bounds [100]. The first method is based on Student's t-distribution and takes the familiar precision ϵ, confidence α, and sample size n as parameters. Since the parameters are interdependent, either n or ϵ is normally omitted. Ultimately, the guarantees about the produced estimate p' are analogous to those obtained via Chernoff bounds, i.e., p is contained in $[p' - \epsilon, p' + \epsilon]$ with confidence α. The second method approximates the t-distribution using the normal distribution, and is thus more appropriate for large sample sizes.

Lassaigne and Peyronnet argue that estimation is more appropriate than hypothesis testing when the actual probability that some property holds is required, rather than simply a threshold comparison as expressed in PCTL formulas [76].

3.4 Bayesian Analysis

The statistical methods covered so far are *frequentist*, in the sense that a probability intuitively corresponds to a relative frequency of some property in a sequence of observed outcomes. The Bayesian approach, on the other hand, is based on the notion of probabilities as *beliefs* that can be modified through observations. One advantage of the Bayesian approach is that prior beliefs in specific domains, refined by accumulation of many observations of a long time, can be used as a basis for new attempted refinements.

Jha et al. propose an algorithm for SMC based on Bayesian hypothesis testing [65]. Intuitively, in each step, the algorithm produces a *Bayes factor* $\mathcal{B}$ that measures the relative confidence in the specification of a system. This factor is then compared against a fixed threshold $T > 1$, and the algorithm terminates if either $\mathcal{B} > T$ or $\mathcal{B} < \frac{1}{T}$. In the former case, the null hypothesis that the given specification holds is accepted, and in the latter case, it is rejected. The algorithm also provides a way for SMC users to specify prior knowledge in the form of a probability distribution; such distributions are often available in the systems biology domain. The algorithm shown requires fewer samples than frequentist SMC in the form of the SPRT, on benchmarks from systems biology.

Zuliani et al. describe an algorithm for Bayesian estimation-based SMC [134]. The algorithm works by repeatedly producing a system trace, computing the so-called Bayesian estimator of the underlying probability of the specification using a provided prior distribution, computing an interval estimate, and finally deriving a posterior probability. If the posterior probability is greater than or equal to a provided coefficient, then the algorithm terminates and returns an interval and the current estimator. The algorithm is used to perform SMC analysis of a Stateflow/Simulink

model of a fuel control system with hybrid behavior. Zuliani et al. demonstrate the algorithms use for analysis of Stateflow/Simulink models of hybrid systems, and show that it is orders of magnitude faster than earlier frequentist estimation-based algorithms.

3.5 Analysis of Rare Properties

A stochastic logic formula may describe a property in stochastic system that is rarely realized. For example, for a PCTL specification $\phi = \mathrm{P}_{\geq\theta}(\psi)$, the probability that a randomly sampled path prefix from a DTMC $\mathcal{M}$ satisfies ψ may be low. With standard SMC techniques, establishing such rare properties with reasonable confidence requires large sample sizes and long traces. In particular, if a system is treated as a Bernoulli trial, with a sufficiently large sample, the distribution will approach a normal distribution with a standard deviation of $\sqrt{npq}$ where n is the sample size, p is the probability of success in a Bernoulli trial and $q = 1 - p$. Thus, the variance of the probability of success will be $\sqrt{pq/n}$. When the values of p are much higher or lower, far larger samples are required. More specifically, for a desired error probability bound α and precision ϵ, the required sample size will be $n = z_\alpha^2(\frac{pq}{\epsilon^2})$, where z_α is the ordinate on the normal curve corresponding to α [93]. This relationship means that sample size grows faster than linearly both as ϵ becomes smaller and as α becomes smaller.

When more reliable numeric-symbolic techniques are infeasible to use to analyze rare properties, a *curtailed single sampling plan* may be used. Under such a plan, a system is deemed unsafe after an *a priori* known number k of failures have been observed, or deemed safe after an *a priori* known number of total observations have been made and the number of failures is less than k [97]. However, this approach gives no control over the probability of falsely rejecting a system that has acceptable chances of failure (although systems without possibility of failures are guaranteed to pass the test). Younes and Simmons suggest to use SMC with a curtailed single sampling plan with $k = 1$ for rare properties. They argue that the tradeoff is attractive and describe an experiment on properties with probability of failure of 10^{-5} or less where the SPRT sometimes requires more than an order of magnitude larger sample size than a curtailed SSP [132].

General methods for reduction of variance in samples from Monte Carlo simulations have been explored since the 1940s; one key technique is *importance sampling* [115]. Intuitively, when importance sampling is applied in SMC, simulations on a system are first weighted in favor of realizing the rare property, and the hypothesis test or estimation method then compensates appropriately when considering the outcomes. The general idea is to introduce a weighting function on the random variables to be observed, without changing their expectancy. The problem then lies in finding a distribution for the weighting function, which results in lower variance than for the standard, crude estimator. In practice, optimal biasing (zero variance) is difficult to achieve, but reasonable distributions may be found by examining members of a parameterized family of distributions to which the optimal distribution belongs. In particular, the *cross-entropy method* is able to select family members that minimize a certain type of divergence from the optimally biasing distribution. To select the appropriate distribution family member, the method requires sampling from the original unbiased distribution.

Reijsbergen et al. present a method based on importance sampling for SMC analysis of CTMCs for a restricted class of CSL properties, demonstrating significant performance gains for a benchmark model of a distributed database system with rare disk failures [104]. Clarke and Zuliani show preliminary results of using importance sampling with the cross-entropy method to analyze a Stateflow/Simulink model in MATLAB of a fault-tolerant fuel control system [35]. Even when adding up the sample points required to compute the biasing distribution and those

in the subsequent biased sampling, unbiased sampling requires many orders of magnitude larger sample size to maintain a reasonable relative error (10^{14} vs. 10^5).

Barbot et al. introduce an importance sampling method for SMC that provides explicit guarantees on the variance reduction and provides a true confidence interval for the estimated probability of the alternative random variable (as opposed to approximate intervals obtained through the normal distribution) [16]. The method assumes that the system can be represented as a Markov chain $\mathcal{M}$ with certain structural properties and finds the biasing parameters by performing numeric-symbolic analysis on a reduced version of $\mathcal{M}$. Barbot et al. automate several, but not all, steps of their method in the SMC tool COSMOS [36] and analyze rarely realized properties in four benchmark systems, aided by the numerical model checker PRISM. As in the experiments of Clarke and Zuliani, importance sampling is shown to improve the scalability of SMC by several orders of magnitude. Jegourel et al. show that when stochastic models are parameterized at the level of syntax using guarded commands, there is an efficient algorithm that finds an optimal parameterized importance sampling distribution [64]. The algorithm avoids avoids an explicit representation of a transition matrix and minimizes a vector of parameters of the model according to the cross-entropy method, ultimately producing a unique optimum. An implementation of the algorithm in the SMC tool PLASMA Lab [98] is shown to give orders-of-magnitude variance reduction over classic SMC on benchmark systems. Developing powerful new importance sampling techniques for use in Monte Carlo simulations is an active area of research, and such techniques are generally applicable in SMC [47, 118].

Importance *splitting* is an alternative to importance sampling for variance reduction. Intuitively, importance splitting decomposes a single simulation where a certain property has a low probability of realization into a *series* of simulations where conditional properties have higher probability to be realized. A central problem in applying importance splitting for SMC is how to perform decomposition of the specification property to enable separate simulations in this way. Jegourel et al. suggest that the decomposition may be expressed as a score function that discriminates paths where the rare property occurs from those where it does not [62]. They describe two importance splitting algorithms, where the first takes a (possibly manually specified) score function as input and the second finds the details of a score function adaptively [30], and evaluate them on benchmarks in PLASMA Lab, showing substantial variance reductions. Jegourel et al. propose further improvements in the form of a heuristic, fine-grained score function and an optimized adaptive algorithm [63]. Legay et al. give an overview of their previous work on importance sampling and importance splitting for SMC and its implementation in PLASMA Lab [80].

Finally, Randomized Quasi Monte Carlo (RQMC) is a collection of methods for reducing variance in statistical analysis. RQMC can be combined with importance splitting or importance sampling, or both, to yield better variance reduction than either alone could achieve. L'Ecuyer et al. discuss RQMC methods and their applications and limitations in a general Monte Carlo simulation setting [78].

4 COMPLEX PROPERTIES

We now deal with more complicated types of properties in stochastic logics that require special treatment. Specifically, we describe techniques to check nested formulas and formulas with unbounded until operators.

4.1 Nested Formulas

Although the PCTL syntax in Table 1 permits nested occurrences of the probabilistic operator P, we have not previously considered such formulas directly. For example, the formula $\mathrm{P}_{\geq\theta_1}(\mathrm{P}_{\geq\theta_2}(\psi)\,\mathrm{U}^{\leq t}\,\phi)$, for given probability bounds θ_1 and θ_2, state formula ϕ, and path formula ψ,

has a well-defined meaning on a DTMC. In one notable SMC case study, nesting is used to express that a robot moving on a grid reaches a point (n, n) within a number of time units with probability at least $\theta_1 = 0.9$, all the while periodically communicating with a base station with probability at least $\theta_2 = 0.5$ [132]. With c true whenever the robot communicates, and $x = i$ and $y = j$ true whenever the robot's position is (i, j) on the grid, the robot specification is:

$$\mathsf{P}_{\geq 0.9}(\mathsf{P}_{\geq 0.5}(\top \,\mathsf{U}^{\leq t_2}\, c)\,\mathsf{U}^{\leq t_1}\, x = n \wedge y = n) \tag{2}$$

Nested formulas are difficult to accommodate with reasonable efficiency using SMC, and then only with significant caveats. Intuitively, a formula and probability bound in the scope of a probability operator represents a hypothesis test on that formula, which means, in the case of nesting, that the outcome of one test depends on the outcome of another test using a different trace. Since tests may have errors, it can then no longer be assumed that path properties can be checked with certainty on a trace. In effect, what is observed when checking a trace is no longer the outcome x_i of a random variable X_i as defined above, but some other outcome y_i of a random variable Y_i related to X_i.

Younes and Simmons initially proposed to handle bounded until formulas with nesting by, in turn, nesting the tests [131]. However, to deal with the uncertainty of outcome in an inner test, and guarantee the outer error bounds α and β, an inner test uses a narrower indifference interval and error bounds α' and β' such that $\alpha' < \alpha$ and $\beta' < \beta$. Generally, these requirements lead to an increased sample size compared to formulas without nesting. In addition, the approach only works for systems that are Markov chains. Younes and Simmons later suggested improvements in handling of nesting to address previous shortcomings in efficiency and soundness [132]. Younes also extended the approach that permits undecided results to the nested case [128].

For scenarios where an explicit definition of the system as a Markov chain is available, Younes et al. present an alternative approach using a combination of numeric-symbolic and SMC methods [130]. The outermost bound and formula ranged over by a probabilistic operator is handled via hypothesis testing, while the inner ones use numeric-symbolic algorithms. The combination is attractive for several reasons. First, numeric-symbolic results restore certainty when checking path formulas in the scope the outermost probability operator. Second, numeric-symbolic methods compute the probabilities that nested formulas hold for all states at once. Third, compared to only relying on numeric-symbolic methods, the use of SMC for the outermost probabilistic operator gives better scaling characteristics for large state spaces. Obviously, this combination of methods is not possible to use for black box systems.

4.2 Unbounded Until Formulas

Consider PCTL state formulas of the form $\mathsf{P}_{\geq\theta}(\phi\,\mathsf{U}\,\phi')$, i.e., formulas where the until operator U carries no restriction on the units of time until ϕ' must become true. For a trace to provide a sample point in statistical testing or estimation, the path formula $\psi = \phi\,\mathsf{U}\,\phi'$ must be unambiguously true or false in the trace. In the case of the corresponding bounded formula, the maximum trace length required to obtain the sample point is equal to the time bound t. However, no finite trace length is sufficient to guarantee obtaining a sample point for unbounded until formulas; for example, paths in a DTMC $\mathcal{M}$ that do not satisfy $\top\,\mathsf{U}\,\phi'$ are all infinite. Consequently, none of the SMC methods described so far in this section directly support the unbounded until operator. In fact, in a recent evaluation of algorithms proposed for this purpose, Roohi and Viswanathan conclude that the problem has not yet been satisfactorily solved [105]. We describe methods that have been proposed thus far and note their deficiencies.

Sen et al. proposed the first SMC algorithm that circumvents the ostensible requirement for infinite paths in sample points for unbounded until formulas [112]. The algorithm builds on the

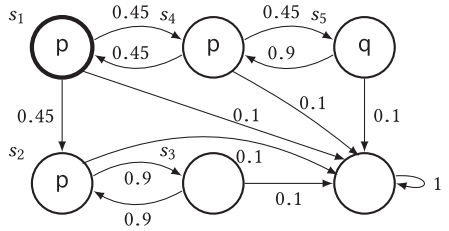


Fig. 5. Discrete time Markov chain with termination probabilities.

insight that for ψ to be true in a path π in $\mathcal{M}$, it must be the case that ϕ' eventually becomes true in π. Hence, if we are simulating $\mathcal{M}$ and have reached a state where it is possible to determine with some certainty that $\neg\phi'$ always holds regardless of how the current trace is extended, we can terminate with the conclusion that ψ is false. Since the remaining possibilities for ψ to be true or false in $\mathcal{M}$ can be witnessed by finite paths, sampling of traces can then proceed as usual. More generally, as the simulation traverses a state, the strategy of the algorithm is to decide statistically whether the PCTL formula $\mathsf{P}_{\leq 0}(\phi \,\mathsf{U}\, \phi')$ holds (which subsumes the case where $\neg\phi'$ always holds) and use the result in a nested test for ψ.

To enable statistical testing of $\mathsf{P}_{\leq 0}(\phi \,\mathsf{U}\, \phi')$, the algorithm introduces a certain *termination probability* $p_T < 1$ to each state in the original DTMC $\mathcal{M}$, so that simulation runs will end. In effect, a new DTMC $\mathcal{M}_T$ is constructed from $\mathcal{M}$, adding an absorbing state s_T to which all other states have transition probability p_T. In addition, if p_s is a transition probability between states s and s' in $\mathcal{M}$, the corresponding transition probability for $\mathcal{M}_T$ becomes $(1 - p_T)p_s$. For example, if we add a termination probability $p_T = 0.1$, then the Markov chain in Figure 2 becomes as shown in Figure 5.

In an SMC implementation, the DTMC $\mathcal{M}_T$ does not need to be constructed and simulated explicitly. Instead, when simulating $\mathcal{M}$, the modified transition probabilities can be taken into account on-the-fly. In each state, such instrumented simulations then terminate and provide a "premature" trace with probability p_T, i.e., the state s_T is never directly reached. Clearly, however, a sample point obtained straightforwardly from a trace produced by the instrumented simulation represents an outcome of a different Bernoulli random variable X_T than the one we originally wanted to observe in $\mathcal{M}$. In particular, the parameter p' of X_T is different from the parameter p of the corresponding random variable X in $\mathcal{M}$. However, as shown by Sen et al., a bound on the parameter of X can be expressed in terms of the parameter of X_T, specifically, $p(1 - p_T)^N \leq p' \leq p$, where N is the number of states in $\mathcal{M}$. Finally, when $p_T = c/N$ for some constant c, this bound allows reducing the model checking problem for $\mathsf{P}_{\leq 0}(\phi \,\mathsf{U}\, \phi')$ with indifference interval half-width δ on $\mathcal{M}$ to the corresponding problem on $\mathcal{M}_T$ with half-width $\delta(1 - p_T)^N$.

Younes et al. argue that the bound relating the parameters of the variables X and X_T in the algorithm of Sen et al. are inadequate in practice and propose two alternative algorithms [129]. The first algorithm relies on precisely deciding $\mathsf{P}_{\leq 0}(\phi \,\mathsf{U}\, \phi')$ on states by performing reachability analysis. In effect, the algorithm considers a non-probabilistic CTL formula on states in $\mathcal{M}$, and removes outgoing transitions from states where the formula is true. From Markov chain theory, we obtain that sample points from the modified DTMC $\mathcal{M}'$ are observations of a stochastic variable with the same parameter as the corresponding variable for $\mathcal{M}$. In addition, if $\mathcal{M}$ is homogeneous (i.e., transition probabilities are independent of the number of steps taken thus far in a simulation), and simulations are terminated in the absorbing states of $\mathcal{M}'$, then finite paths will be obtained with certainty in $\mathcal{M}'$. Reachability analysis is resource-intensive in practice and limits the applicability of the algorithm, excluding. However, the use of statistical techniques in the final analysis step avoids the principal overhead of numeric-symbolic methods, allowing scaling analysis to larger

state spaces than such methods can handle. Obviously, reachability analysis is not applicable to black box systems.

The second algorithm proposed by Younes et al. uses a termination probability p_T in the same way as the algorithm by Sen et al., i.e., for constructing the new DTMC $\mathcal{M}_T$ with altered transition probabilities and extra state s_T as above. However, $\mathcal{M}_T$ is here used for sampling a certain non-Bernoulli stochastic variable X_T, which, as the authors show, has the same expectation as the parameter p of the variable X in the original DTMC $\mathcal{M}$. Due to X_T being non-Bernoulli, statistical testing cannot be used on samples from $\mathcal{M}_T$. Instead, the algorithm uses an estimation method due to Chow and Robbins that works on variables with unknown but finite variance [32]. Following the established pattern for estimation, the output value p', in this method defined as the arithmetic mean of the sampled observations, is such that the probability that $|p' - p| \leq \delta$ is greater than or equal to $1 - \alpha$. The method proceeds sequentially, by checking a stopping rule after each obtained observation. Younes et al. give a characterization of when the condition of finite variance holds in terms of the termination probability p_T. More precisely, X_T has finite variance if and only if $p_T < 1 - |\lambda_*|$, where λ_* is the subdominant eigenvalue of the transition matrix M of $\mathcal{M}$. Computing the exact subdominant eigenvalue of a matrix is a hard problem in general, so approximations based, e.g., on the number of states in $\mathcal{M}$ may be used to determine p_T in practice, at the risk of bad performance and low accuracy.

Younes et al. note that their approach to constructing the non-Bernoulli random variable is originally due to von Neumann and Ulam, who use it in a Monte Carlo algorithm for computing the inverse of a matrix. The algorithm of von Neumann and Ulam can be viewed as a randomized version of a numerical algorithm for solving systems of equations. Both the randomized algorithm and its numerical counterparts depend on the subdominant eigenvalue of a matrix to achieve a certain precision, which is echoed in the analysis of the SMC algorithm.

Basu et al. propose an algorithm for deciding unbounded until formulas based purely on estimation [17]. The algorithm computes an estimate p' of $\psi' = \phi \, \mathsf{U}^{\leq t} \, \phi'$, for carefully chosen (large) values of t, which then becomes the estimate of ψ in $\mathcal{M}$. The underlying insight is that, as t becomes arbitrarily large, the probability measure of ψ' and ψ coincide, and additionally, there is some point $k_0 = t$ where the difference in the measures becomes small enough to be negligible with respect to the required precision. The first phase of the algorithm identifies such a point k_0 by iteratively estimating measures of formulas of the form:

$$\psi_k = (\phi \, \mathsf{U}^{\leq k} \, \phi) \vee (\neg\phi' \, \mathsf{U}^{\leq k} \, (\neg\phi \wedge \phi')). \tag{3}$$

Such formulas state that ψ is either true or false in at most k steps. The second phase computes an estimate of the measure of ψ_{k_0} and returns this as the final result. With all intermediate formulas using the bounded until operator, the standard estimation techniques of, e.g., Hérault et al. become applicable [54]. However, the first phase of the algorithm as presented by Basu et al. comes with the caveat that $\mathsf{P}_{\leq 0}(\neg(\top \, \mathsf{U} \, (\phi' \vee \neg\phi)))$ is assumed to hold in the state s in $\mathcal{M}$ for which the unbounded until formula is considered, i.e., it assumes that there are no positive-probability infinite paths in $\mathcal{M}$ starting from s where $\phi \wedge \neg\phi'$ holds in every state. Roohi and Viswanathan show that the algorithm can be modified to work with any reliable estimate of the measure of $\neg(\top \, \mathsf{U} \, (\phi' \vee \neg\phi))$ in s, which may nevertheless be a considerable burden to obtain in practice [105].

Both Lassaigne and Peyronnet as well as El Rabih and Pekergin propose to use the subdominant eigenvalue of the matrix M of ergodic Markov chains to obtain a time bound k_0 that allows the measure of the bounded until formula be close to that of the corresponding unbounded until formula [44, 76]. However, Roohi and Viswanathan give an ergodic counterexample where the bounded and unbounded formulas have measures far apart [105]. Another algorithm idea due to

El Rabih and Pekergin is based on sampling until a previously sampled state in encountered, in effect forming a finite *lasso* [44, 46]; Roohi and Viswanathan show that the measure spaces of stochastic logics are incompatible with this approach [105].

Younes et al. compare the performance of variations of their algorithms based on reachability analysis and a termination probability with that of two iterative numeric algorithms implemented in PRISM, for two standard benchmarks (analysis of a polling system and a tandem queuing network, respectively) [129]. Similarly to algorithms for bounded until, the numeric algorithms come out on top for small state spaces, but their performance degrades quickly as the state space grows. The two statistical algorithms alternately win the performance race for large state spaces, with the cost of reachability analysis ultimately determining the outcome. Roohi and Viswanathan compare the performance of the algorithm of Sen et al. (Algorithm 1), the reachability analysis algorithm and termination probability algorithm of Younes et al. (Algorithms 2 and 3, respectively), and the algorithm of Basu et al. (Algorithm 4) on the polling system and tandem queueing network benchmarks [105]. For the polling system, Algorithm 2 is the fastest and Algorithm 4 is the slowest, with Algorithm 1 the runner-up when a polling failure probability is used, and Algorithm 3 the runner-up without such failures. For the queueing network, Algorithm 1 was too slow to include in the comparison; Algorithm 4 was slowest among the rest, with Algorithm 3 narrowly edging out Algorithm 2 for large state spaces.

More recently, Daca et al. present an algorithm for unbounded untils that only requires only requires a lower bound on the minimum transition probability of the underlying assumed Markov chain [37]. The main idea of the algorithm is monitor the process of obtaining a sample path and use the obtained information to build knowledge about the structure of the Markov chain. The algorithm of Daca et al. shows substantial improvements over earlier algorithms in experiments on DTMC and CTMC benchmarks in the PRISM model checker. In addition, a modified version of the algorithm can handle mean payoff (long-run average reward) properties of Markov chains. Similar ideas for SMC on-the-fly analysis are explored by Legay and Traonouez in a different context to detect and flag changes in the underlying probability of some property [81].

The problem of handling unbounded untils is distinct from, but related to, handling *steady-state* formulas (alluded to in Section 2). Such formulas can express bounds on the probability that some property holds in the long run for a stochastic process. El Rabih and Pekergin propose to use *perfect sampling* [44] from the steady-state distribution to handle these formulas, via an approach that determines on-the-fly how far to run the system [101].

5 NONDETERMISTIC, TIMED, AND HYBRID SYSTEMS

Systems in the real world often have properties such as nondeterministic behavior, real-time deadlines, and behaviors governed by continuous variables. SMC algorithms and tools have been developed for analyzing these models. Nondeterminism is a natural model of concurrency in a system where the probability distribution of the inputs or communication delays may not known. In fact, the inputs or delays may not even conform to a probability distribution. This makes it difficult for a simulator to choose schedules.

In both nondeterministic as well as timed or hybrid systems, expressing properties requires specialized logics. Moreover, timed and hybrid systems add complexity in simulations as these simulations have to include timing constraints and, in case of hybrid systems, may have to solve systems of differential equations. However, timed and hybrid systems may be fully stochastic and therefore their integration with SMC is more direct than it is in case of nondeterministic systems. Thus the techniques for these systems are very different. We treat these cases of real-world systems separately.

5.1 Nondeterminism

Successful application of SMC depends crucially on that a suitable probability space can be defined for a system. Generally, this means that the system must be *fully stochastic*, i.e., it never performs nondeterministic steps. However, a practically useful class of systems that interleaves stochastic state transitions and nondeterministic choices have been formalized as MDPs and are well studied in the literature. MDPs have also found many applications [124]. MDPs can have rewards associated with nondeterministic choices, which may be used to model certain optimization problems. More generally, MDPs can reflect the behavior of concurrent probabilistic systems, where probabilistic transitions correspond to local stochastic behavior that depends on state, and nondeterministic transitions correspond to behavior where several nodes or sites perform actions concurrently in some unspecified order. For example, such actions may be triggered by events external to the system. MDPs only become amenable to classic SMC analysis if nondeterminism is resolved by the use of a scheduler, in which case the system becomes an ordinary Markov chain. Schedulers may be arbitrarily complex in general, whence assumptions are often made on how they are represented, and on their memory requirements, to enable analysis in practice.

Bogdoll et al. extend SMC to handle systems with spurious nondeterminism, i.e., systems where all schedulers give rise to the same transition probabilities, and consequently the same Markov chain [19]. Such spuriousness may arise in practice, e.g., when using compositional approaches in system design, and is then a hindrance to application of simulation techniques. Bogdoll et al. use networks of probabilistic automata, which are similar to MDPs, as a basis for an algorithm for SMC based on partial order reduction. The algorithm is implemented in the tool MODEST [22] and compared on benchmarks with spurious nondeterminism against an approach that resolves nondeterminism uniformly.

Lassaigne and Peyronnet extend estimation-based probabilistic planning, which is related to but distinct from verification of stochastic logic properties as in SMC, to MDPs for the restricted class of memoryless schedulers [77]. Henriques et al. similarly assume memoryless schedulers, but in a more standard verification-based setting where the objective is find schedulers that maximize or minimize the probability of some property [53]. They propose an algorithm that resolves nondeterminism probabilistically and then applies reinforcement learning techniques to guide resolution of actions towards an optimum. The algorithm is implemented in the PRISM tool and shown to scale better than numeric-symbolic techniques.

Legay et al. describe an approach to SMC analysis of MDPs based on sampling of history-dependent schedulers that avoids some of the issues with memory requirements in previous solutions [79]. In related work, D'Argenio et al. describe sophisticated scheduler sampling techniques that make the approach of Legay et al. more practical [38].

As part of work that straddles MDPs and statistical methods, Lukina et al. present a synthesis algorithm of optimal plans for taking the initial state of some MDP to a state with cost below some given threshold [85]. The algorithm is inspired by importance splitting, and uses extensively a randomized approximation algorithm called Particle Swarm Optimization, inspired by bird flocking.

5.2 Timed Systems

Many real-world systems are subject to hard requirements on time, or, more generally, consumption of resources such as energy. Classical approaches to analysis of such systems focus on modeling them as timed automata and expressing requirements in variants of CTL that include operators that take resource bounds as parameters. Tools and techniques can then establish worst-case bounds on execution time and resource consumption, and perform schedulability analysis. However, there may still be a need to choose among appropriate schedulers, preferring, say, the one

that provides the most attractive properties in the expected or average case. At the same time, many basic problems for timed automata and generalizations, e.g., priced timed automata (PTAs), are known to be undecidable. For example, model checking the CTL variant for PTAs, WCTL (Weighted CTL), is undecidable even when only using three different clocks to represent time and consumption of resources [24].

David et al. define a stochastic semantics for networks of PTAs (NPTAs), where each component decides independently based on clock variables, and subject to probability distributions, when and what to output to other components [41]. Simulation and SMC analysis of systems in the semantics using the stochastic logic PWCTL have been implemented as an extension to the UPPAAL timed model checker, and applied in many case studies, e.g., communication protocols for wireless sensor networks and job-shop problems [26, 42].

Bogdoll et al. implement a discrete-event simulator for Stochastic Timed Automata (STA), called MODES [20]. Similarly to NPTAs, STAs can express real-time behavior and coupled delays with arbitrary probability distributions. In addition, STAs allow nondeterministic choices. Together with other components in the MODEST tool set [22], MODES enables sound SMC analysis of a class of nondeterministic models expressed in the MODES language. Bohlender et al. provide a comparison of SMC performance for analysis of timed systems in the tools UPPAAL and MODES [21].

5.3 Hybrid Systems

Hybrid systems consist of a discrete part and a continuous part that interact in various ways, with evolution of the discrete subsystem governed programmatically and that of the continuous part by differential equations. Abate et al. show that the dynamics of a stochastic hybrid systems can sometimes be reduced to that of a DTMC, whence SMC analysis methods become applicable [1].

Meseguer and Sharykin extend the probabilistic rewriting logic language PMaude to enable specifying object-based stochastic hybrid systems [89]. The resulting language, SHYMaude (Stochastic Hybrid Maude), allows transitions in such systems to be defined using probabilistic rewrite rules. SHYMaude's implementation on top of PMaude [5], which has an interface to the PVESTA tool, gives a way to perform SMC analysis on SHYMaude models. Properties about SHYMaude models specified in PMaude's QuaTEx language. Meseguer and Sharykin describe two case studies in SHYMaude, a repeated second-price auction and a thermostat model, and report that VESTA is able to establish the respective QuaTEx correctness properties with 95% confidence.

Martins et al. introduce the notion of Distributed Discrete-Time Hybrid Automata (DDTHA), which intuitively allows expressing interconnected systems of hybrid automata that communicate asynchronously [88]. By probabilising the schedulers for the hybrid automata, they obtain a stochastic semantics for DDTHA for which properties of executions can be expressed in a bounded stochastic logic. Finally they describe an algorithm for Bayesian SMC to analyze such systems of stochastic hybrid automata.

David et al. extend their stochastic semantics for NPTAs to let clocks rates depend on values from other clocks, which amounts to defining ordinary differential equations, and modify the SMC capabilities of UPPAAL to permit analysis of the resulting stochastic networks of hybrid automata (NHAs) [39]. In later work, David et al. extend the approach using NHAs and its UPPAAL implementation to permit modeling systems with dynamic creation and termination of individual hybrid automata [40]. This extension requires a dynamic specification logic, since not all hybrid automata ("threads") are known beforehand. The proposed logic is called Quantified Dynamic Metric Temporal Logic (QDMTL), and is defined over an arbitrary collection of *templates* for processes that may be statically or dynamically instantiated in an execution. Template variables can occur in formulas and are placeholders for the runtime names of processes. As required for SMC analysis, a proper measure on execution paths can be defined for QDMTL formulas. David et al. analyze a

variant of a case study originally due to Younes [132], where robots are dynamically spawned on a grid, as well as a client-server example using UPPAAL. Notably, the UPPAAL implementation of SMC with QDMTL uses formula rewriting to allow concurrent generation of a path prefix and path formula evaluation, lowering total analysis time.

Kalajdzic et al. propose a feedback-control framework for statistical checking of cyber-physical systems [67]. The key ideas of the framework is to learn the system model by observing outputs, determine the hidden state by statistical inference, and finally infer the system control policy, to estimate the probability of rare events. This is achieved by applying importance sampling and importance splitting in a feedback-control setting. The framework is demonstrated on a CPS model of the classic dining philosophers scenario and a model of a sequence of Bernoulli trials.

SMC requires obtaining truth values of path properties from system traces. However, it may not be feasible in practice to keep entire traces in memory or on storage. A more efficient approach is to equip system executions with runtime monitors synthesized from the path properties [49]; this is possible even when execution is distributed in a network [110]. Most directly, simulators can use instrumentation to add monitors, but more complex systems such as hybrid system may require custom on-the-fly monitoring [86].

6 APPLICATIONS

Within computer science, probability theory is generally useful for performance analysis of non-stochastic systems, analysis of randomized algorithms, describing environment uncertainty, and capturing aggregate behavior of large populations. Typical applications of stochastic model checking for analysis are partly reflected in benchmarks commonly used to compare tools: leader election and synchronization protocols in distributed systems, queueing systems, and birth-death processes [61]. However, recent developments of SMC-related tools are often motivated by the needs of analysis in specific application areas that are ultimately outside of computer science, such as systems biology and implementation of control mechanisms for, e.g., energy systems.

6.1 Communication Protocols

Since SMC provides a randomized approach to stochastic model checking, SMC can be used wherever the corresponding numeric-symbol techniques are applicable, which historically includes probabilistic analysis of network protocols. For example, the CSMA/CD protocol, used in networking over Ethernet, has been analyzed using both the estimation-based APMC tool and the numeric-symbolic engine in PRISM [43]. Haverkort et al. describe a case study of dependable cluster of workstations [50], which has been successfully analyzed using SMC techniques, in, e.g., the Ymer tool for large parameters [132]. Aspects of protocols other than average-case performance, such as resistance to denial-of-service attacks, have also been considered using SMC techniques [4]. More recently, SMC techniques have been used to perform quantitative analysis of the consistency of distributed key-value stores [84].

MDPs can model distributed systems with local probabilistic actions where nondeterministic choices are deferred to a scheduler. Legay et al. consider a wireless LAN protocol modeled as an MDP and analyze probability of collisions in the protocol using SMC techniques [79].

6.2 Queueing Systems

Queuing systems generally assume specific probability distributions that govern arrival and processing rates, whence no additional assumptions about the environment need to be made to enable analysis. An often-used case study for SMC is a tandem queuing network originally due to Hermanns et al. [56]. The property of interest for the system is whether the probability that queues become full after t time units is less than 0.5. Younes and Simmons provide a comprehensive

analysis of such networks using Ymer [132], and Jansen et al. perform a similar analysis with many different tools [61]. Another popular SMC benchmark analyzed in Ymer and other SMC tools is an n-station symmetric polling system described by Ibi and Trivedi, where a single server goes through all stations and processes messages that have arrived [59]. A correctness property states that if the first station has a message, will be attended to by the server within t time units with probability at least 0.5.

6.3 Real-Time and Cyber-Physical Systems

Many communication protocols have real-time or energy consumption considerations, and may thus benefit from SMC average-case analysis. For example, David et al. consider the Lightweight Medium Access Protocol (LMAC) for wireless sensors and explore using SMC techniques how to minimize energy consumption and collisions [41].

Proper cyber-physical systems (CPSs) whose dynamics can be modeled by differential equations have been analyzed using SMC techniques, e.g., fuel control systems [35, 134] and thermostats [89]. Martins et al. use SMC to analyze a model of a dynamically adaptable power grid expressed as interconnected hybrid automata [88].

Distributed Adaptive Real-time Systems (DARTs) consist of independent nodes that communicate to achieve some goal, and are able to increase their chances of success via self-adaptation. Kyle et al. introduce the DART Modeling and Programming Language (DMPL) that can express DART systems and their specifications [72]. They implement a logging tool for DART nodes and perform SMC analysis on all collected logs to estimate the probability of a property in a DART. The approach is evaluated in several scenarios involving mobile robots, which perform various time-critical maneuvers and adapt to environmental conditions.

A System-of-Systems (SoS) is a dynamic conglomeration of distributed and independent systems that collaborate to achieve an overall goal. Both the constituent systems and the overall architecture of an SoS may evolve over time, e.g., by systems leaving or arriving. An air traffic management systems at an airport is one example of an SoS. Arnold et al. propose an approach to verification of SoSs based on reducing the problem to SMC using the Plasma Lab tool [9]. In this approach, requirements are first expressed in formalism called the Goal Contract Requirement Language (GCRL), which constrains inputs and outputs of constituent systems. Then, GCRL formulas are translated to BLTL formulas supported by Plasma Lab. An SoS is specified by means of its architecture in the UPDM modeling language, and constituent systems as so-called Functional Mock-up Units (FMUs). At analysis time, FMUs are jointly simulated to produce traces for SMC-based verification.

Interlocking systems ensure the safety of train traffic in stations. Interlockings are configured using data on track layout and interlocking alternatives. While safety (e.g., absence of train collisions) is of paramount concern, interlocking systems must also provide various liveness properties, such as eventual progress of trains away from a station. Cappart et al. present an SMC-based approach to verification of interlocking systems that avoids the state space explosion problem of previous analyses based on regular model checking [29]. The idea is to express liveness properties in the stochastic logic BLTL and use a variety of statistical tests on simulated interlocking scenarios to verify the system.

A hierarchical scheduling systems consists of a set of scheduling units (e.g., CPU tasks) organized in a tree. Each scheduling unit is equipped with a deadline, and an objective is to determine the *schedulability* of some task with respect to a system, i.e., whether the task can be executed without missing its real-time deadline under the specific assumptions on the total CPU time. Such problems arise, for example, in avionics, and have been addressed primarily by analytical means that do no scale well. Chadli et al. present a framework for capturing schedulability problems

using timed and stochastic automata in the UPPAAL tool [31]. For the latter kind of automata, this enables the use of SMC using the UPPAAL-SMC extension.

Dynamic software architectures are able to capture the dynamism in the environment present in modern systems by being capable of being reconfigured over time while still upholding key properties. Quilbeuf et al. present a stochastic logic called DynBLTL that can describe such properties [103]. They implement support for the logic in the PLASMA tool, enabling support for verifying systems specified in the π-ADL architectural language using SMC.

6.4 Systems Biology

The dynamics of biological systems can sometimes be modeled as stochastic processes, usually finite-state or countably infinite CTMCs. Most biological models lack key parameters; these parameters can usually only be obtained by expensive real-world experiments. In addition, even if available, parameter values can be affected by imprecisions in the initial conditions of experiments. As Zuliani describes in a recent survey, the combination of models with large state spaces and the fact that precise analysis methods often do not yield additional predictive power is a strong argument in favor of applying SMC for biology [133].

The main idea of many biological models is to represent a series of transformations of bags of molecular types (chemical species) to other bags. The dynamics of the transformations can be represented either via ordinary differential equations (ODEs) or using a stochastic simulation algorithm (SSA); for high numbers of molecules, an SSA can approximate the more precise ODEs.

Using an SSA, SMC can be used either for verification of model properties or for parameter estimation and synthesis; both have been considered in detail for various biological systems. Jha et al. analyze several benchmarks from MATLAB's Systems Biology Toolbox, modeling, e.g., chemical reactions, using Bayesian SMC methods [65]. Sankaranarayanan and Fainekos model the insulin-glucose regulatory system to capture possibly-faulty behavior from insulin pumps, and they use Bayesian SMC to estimate the probability of dangerous conditions, as specified using Metric Temporal Logic [107]. Jha and Langmead use SMC with Bayesian hypothesis testing to check rare properties of biological models defined using stochastic differential equations; in particular, they consider properties on systems that model tumor progression [66]. Palaniappan et al. use SMC to calibrate models of pathway signaling based on ODEs [95].

Peng et al. consider the problem of composing existing models from population biology into one while adapting and ensuring preservation of key properties of the constituents [96]. Their proposed approach is to generate simulations for the composed model and check hypotheses synthesized from previous properties by applying SMC. The models in their case study of several predator-prey models are expressed in a formalism having CTMC semantics with properties in CSL and non-stochastic linear temporal logic.

Heiner et al. perform a comparison of analysis methods for Generalized Stochastic Petri Nets (GSPNs) that capture biochemical networks [52]. Specifically, they compare using (a) exact numerical analysis, (b) approximate numerical analysis, and (c) simulation. No overall winner is declared, but they note that simulation is a more generally applicable technique than the others. In addition, Heiner et al. conclude that memory is the limiting factor for numerical methods, while accuracy requirements limit simulation-based methods.

7 TOOLS

In this section, we describe some key SMC tools and their relation to specific applications of SMC. While there are many tools that support numeric-symbolic stochastic model checking with the same input as SMC (e.g., a DTMC and a PCTL formula), we do not include such tools unless they incorporate SMC techniques in some form. We also restrict ourselves to tools that are currently

Table 5. SMC Tool Property Comparison; HT Means Hypothesis Testing, E Means Estimation, NS Means Numeric-symbolic Methods

Tool	Systems	Logics	Area	Techniques
Ymer	DTMC/CTMC/GSMP	PCTL/CSL	general	HT+NS
(P)VESTA	DTMC/CTMC/PMaude	PCTL/CSL/QuaTEx	general	HT
MultiVESTA	DTMC/CTMC/...	PCTL/CSL/MultiQuaTEx	general	HT
PRISM	DTMC/CTMC	PCTL/CSL/pLTL/PCTL*	general	HT+E+NS
UPPAAL	NPTA/NHA	PWCTL	timed systems	HT+E
COSMOS	DESP	HASL	general	HT+E
SBIP	DTMC	PBLTL	component-based	HT+E
PLASMA Lab	DTMC/CTMC/...	BLTL/ABLTL/GCSL	general/biology	HT+E
MRMC	DTMC/CTMC	PCTL/CSL	general	E+NS
SAM	StoKLAIM	MoSL	mobile systems	E+NS
APMC	DTMC/CTMC	pLTL/PCTL	general	E
MODES	STA	MODEST	timed systems	HT
MARCIE	XSPN	CSRL/PLTLc	general	E+NS
GreatSPN	GSPN/SWN	CSLTA	general	E+NS

available directly to the academic community in some form. This means that we exclude the SMC tool BioLab [34], since to the best of our knowledge it has never seen a public release.

Table 5 lists the tools surveyed, along with their fundamental properties, i.e., the systems and logics they support, what application areas they are targeting, and the SMC techniques they implement.

7.1 PRISM

PRISM is a general stochastic model checking tool that supports both numeric-symbolic techniques and SMC, written primarily in Java and developed mainly at the University of Oxford [69]. PRISM has its own language for specifying stochastic systems in the form of DTMCs, CTMCs, MDPs, and Probabilistic Time Automata PTAs. Properties can be specified in PCTL, CSL, pLTL, and PCTL*. PRISM can simulate systems at the level of the language (on-the-fly), and hence does not need to store complete representations in memory. The PRISM SMC component [100] supports two different confidence interval methods, one estimation method based on Chernoff bounds, and the sequential probability ratio hypothesis test, with only the latter being restricted to threshold-based properties such as those defined in the PCTL fragment in Section 2. While PTAs are not supported at all by PRISM's simulator, MDPs can be checked as usual, but with potentially misleading results, since nondeterminism is resolved in a uniformly random way. PRISM is available under the open source GNU GPL license, version 2.

7.2 Ymer

Ymer is an SMC tool written in C and C++, developed initially at Carnegie Mellon University and later at Google [125]. Ymer uses a dialect of the PRISM language to describe systems, which can be DTMCs, CTMCs, or GSMPs. Properties can be defined in either PCTL or CSL. Like PRISM, Ymer simulates systems at the language level, and even includes a numeric-symbolic engine from PRISM that can be used in tandem with its standard sequential hypothesis tests to more quickly decide

nested formulas. Ymer makes use of the fact that system traces are to be observed independently to enable parallelization via multithreading on a single single machine and distribution of work across multiple machines. Ymer is a general-purpose SMC tool and has been used to, e.g., verify simulated stochastic behavior of robots on grids [127]. Ymer is available under the open source GNU GPL license, version 2.

7.3 VESTA and PVESTA

VESTA is an SMC tool written in Java and developed at the University of Illinois at Urbana-Champaign, initially released in 2004 [113]. PVESTA is an extension of VESTA [102] that adds a client-server architecture for parallel, distributed sampling and checking [6]. PVESTA includes the original DTMC/CTMC simulation engine from VESTA and supports a format for system definitions close to that of PRISM, along with models in the PMaude probabilistic rewriting logic dialect [5]. Properties can be defined in PCTL, CSL, or the more powerful language of Quantitative Temporal Expressions (QuaTEx). PVESTA is released without source code under a University of Illinois license.

VESTA and PVESTA have been applied to analyze denial-of-service attacks using PMaude models for TCP-related protocols, using the Maude rewriting logic engine for Monte Carlo simulations [4]. More recently, PVESTA has been used to analyze the performance of models of protocols for distributed key-value stores [84].

7.4 MultiVESTA

MultiVESTA [92] is an extension of PVESTA, developed at IMT Lucca, that builds on the premise that VESTA's analysis algorithms are independent of the specific system model and simulation engine used. Hence, MultiVESTA adds general support for standalone discrete event simulators, e.g., the ARGoS swarm robotic library and the DEUS simulation environment. In addition, the tool supports the MultiQuaTEx logic that in contrast to regular QuaTEx allows specification of multiple measures to be observed in a single simulation. MultiVESTA has been applied to analyze cloud computing scenarios where tasks must be evenly distributed among volunteer participants. Through the use of MultiQuaTEx formulas, an order of magnitude improvement in the number of simulations required can sometimes be achieved. MultiVESTA is available under the same license as PVESTA.

7.5 UPPAAL

UPPAAL is a tool for modeling and verification of real-time systems, developed jointly by researchers at Uppsala University and Aalborg University [120]. The basic underlying framework is that of timed automata, and the problem considered by UPPAAL is called timed model checking. UPPAAL has been extended with discrete-event simulation and SMC analysis capabilities for stochastic analysis of timed systems [42]. UPPAAL can perform distributed SMC while avoiding output bias by splitting simulations optimally [26]. UPPAAL binaries are provided under a proprietary license that permits free non-commercial use.

7.6 COSMOS

COSMOS is an SMC tool for model checking of Discrete Event Stochastic Processes (DESPs) in the form of General Stochastic Petri Net (GSPN), implemented in C++ [36]. DESPs resemble GSMPs and are thus more general than Markov chains; they can have an infinite number of states and arbitrary delay distributions. An execution of a DESP is an infinite sequence of states with both an *event* and an *occurrence time* attached to each transition.

To express properties about DESPs, COSMOS supports specifications in the Hybrid Automata Stochastic Logic (HASL). A formula in HASL consists of two parts, the first of which is a Linear Hybrid Automaton (LHA) that attempts to synchronize with an execution of the given DESP. The second part of a HASL formula is an expression related to the automaton, based on moments of a path random variable. While classical statistical model checking focuses on the first moment of a binomial stochastic variable, COSMOS can produce estimates of general expressions involving random variables related to the specification formulas. The tool uses confidence interval computation and exhibits performance comparable to PRISM on several benchmarks [15]. COSMOS is released under the open sourcen GNU GPL license, version 3.

7.7 SBIP

SBIP is an SMC tool with support for a stochastic extension of the Behavior Interaction Priority (BIP) language for modeling component-based systems [108]. A system in Stochastic BIP (SBIP) behaves equivalently to a DTMC. Since stochastic BIP systems are assumed to have finite life, properties are specified in PBLTL, a stochastic bounded variant of LTL. SBIP supports SMC based on both hypothesis testing and confidence interval computation. One motivation for extending BIP with probabilities was to enable capturing uncertainty, e.g., with respect to deployment environments, when designing systems.

SBIP has been applied to analyze the accuracy of a clock synchronization protocol, to analyze the quality of service of an implementation of an MPEG2 video codec, and in verification of protocols used in embedded systems for avionics [94]. SBIP is developed at Verimag and is implemented primarily in Java, with some components only available as Linux binaries.

7.8 PLASMA Lab

PLASMA Lab is a Java library developed at INRIA that enables applying SMC techniques in a variety of contexts, e.g., as part of other tools [98]. For example, the tool has been used to enable SMC analysis through SciLab and MATLAB [25]. PLASMA Lab supports Bounded LTL (BLTL) specifications, along with Adaptive BLTL and GCSL. The tool offers three analysis methods in the form of simple Monte Carlo, Monte Carlo using Chernoff bounds, and sequential hypothesis testing. PLASMA Lab also supports parallel and distributed sampling. One of the key points in developing PLASMA Lab was to avoid tying the tool to a specific high-level language for stochastic systems. PLASMA Lab supports a Biological Language (BL) to define systems that targets applications of SMC in systems biology.

7.9 MRMC

Markov Reward Model Checker (MRMC) is a tool for verification of both classical DTMCs and CTMCs and various Markov reward models, developed primarily at RWTH Aachen [91]. MRMC supports PCTL and CSL for property specification (along with analogues for reward models). MRMC offers SMC with confidence interval computation as an alternative to its main numeric-symbolic methods for PCTL/CSL model checking. However, since MRMC always loads Markov chain representations into memory completely, some of the benefits of simulation-based techniques are not available.

MRMC supports the *.tra* format generated by other tools for stochastic Petri Nets and process algebras. MRMC is implemented in C and is available under the open source GNU GPL license, version 3.

7.10 SAM

Stochastic Analyzer for Mobility (SAM) is a statistical model checking tool for systems defined in the StoKLAIM language, implemented in OCaml and developed at the University of Florence [106]. StoKLAIM is a stochastic extension of KLAIM, which targets modeling and verification of mobile and distributed systems. SAM supports specifications of StoKLAIM systems in the Mobile Stochastic Logic (MoSL), which is similar to CSL but additionally supports referencing the spatial structure of a network. SAM can also perform numeric-symbolic model checking by using the MRMC tool as backend. SAM has been used to analyze and compare several leader election protocols [28].

7.11 APMC

Approximate Probabilistic Model Checker (APMC) is an SMC tool that uses Chernoff bounds to compute confidence intervals after trace generation and checking [54]. The last production release of APMC, version 2.0, is implemented in C [7]. A beta version of version 3.0 is implemented in a mix of Java and C, and added a layer for distributing the work of model checking [8, 55]. An experimental version of APMC for the Cell microprocessor (APMC-CA) was also developed and shown to outperform APMC 3.0 on two standard benchmarks [23]. APMC has been used to analyze the IEEE 802.3 CSMA/CD protocol [43], and MAC protocols for sensor networks [27].

7.12 MODES

MODES is discrete-event simulator for SMC analysis of timed systems in the form of STA [20, 90]. It is part of the comprehensive MODEST toolset that can perform analysis of a variety of probabilistic models and also uses UPPAAL as a backend. The toolset takes stochastic timed systems defined in the MODEST language as input and can perform, e.g., the sequential probability ratio test on given properties.

7.13 MARCIE

MARCIE is a tool for analysis of GSPNs, including those that have been augmented with *rewards* [51, 87]. While the semantics of GSPNs can be mapped to that of CTMCs, MARCIE elides any conversion to CTMC representations internally. MARCIE has several analysis engines, among which is a stochastic simulation engine based on statistical estimation, which supports multi-core parallelization. For simulation-based analysis, properties of models can be expressed as formulas in Continuous Stochastic Reward Logic (CSRL) or probabilistic LTL with numerical constraints (PLTLc). The simulation engine also supports XSPNs, which extend GSPNs with deterministically timed transitions and do not have the Markovian (memoryless) property in general.

7.14 GreatSPN

GreatSPN is a tool for modeling and analysis of both timed and stochastic Petri nets [12, 18, 45]. GreatSPN supports both GSPNs and Stochastic Well-Formed Nets (SWNs). The latter allows distinguishing between different tokens by their *color*, and transitions to be parameterized on such colors. GreatSPN has a graphical interface to build a Petri net and specify so-called performance indexes, which can drive the analysis. The tool includes several analysis modules, among which is a module for event-driven simulation that can compute confidence intervals, i.e., perform statistical estimation.

7.15 Tool Comparison

While hypothesis testing (HT) of the kind applied in SMC is inherently parallelizable, both at the level of multiple processor cores and across different systems, not all tools that implement HT

Table 6. SMC tool Implementation Comparison

Tool	Initial	Last	System Language	Impl. Lang.	Distributed?	License
Ymer	2004	2015	PRISM	C/C++	✓	GPLv2
(P)VESTA	2004	2011	PRISM/PMaude	Java	✓	proprietary
MultiVESTA	2013	2015	PRISM/PMaude	Java	✓	proprietary
PRISM	2011	2015	PRISM	Java/C	✘	GPLv2
UPPAAL	2011	2014	UPPAAL		✓	proprietary
COSMOS	2011	2015	GSPN	C++	✘	GPLv3
SBIP	2013	2013	SBIP	Java/C++	✘	proprietary
PLASMA Lab	2013	2016	PRISM/ARML/BL	Java	✓	as-is
MRMC	2008	2011	.tra	C	✘	GPLv3
SAM	2010	2010	StoKLAIM	OCaml	✘	as-is
APMC	2004	2006	PRISM	C/Java	✘	GPLv2
MODES	2012	2017	MODEST	C#	✘	proprietary
MARCIE	2010	2017	GSPN/XSPN	C++	✘	proprietary
GreatSPN	1995	2013	GSPN/SWN	C/Java	✘	proprietary

offer either possibility. Sequential HT has the notable property that a test run in parallel may be considered a *speculation*, whose result is not part of the final output. SMC tools based on estimation may also be parallelized and distributed.

Table 6 compares the same tools as above for more practical properties, i.e., the year of initial and latest release, the language for defining systems, tool implementation language, whether model checking can be distributed across computers, and the tool license terms.

Jansen et al. carried out an in-depth performance comparison of some stochastic model checking tools [61]. Among the tools covered here, MRMC, PRISM, Ymer, and VESTA were included in the comparison. However, SMC was only used when running Ymer and VESTA. The comparison concluded that Ymer was the fastest and slimmest tool (even including numeric-symbolic tools), having low and nearly constant memory consumption when run on the selected benchmarks. While VESTA's memory use was also nearly constant, it was similar to that of numeric-symbolic tools in several cases. In addition, Jansen et al. note that VESTA's runtime varies considerably more than Ymer's, with nested properties in particular taking an unfeasibly long time to check in VESTA even on small systems. On the usability side, Jansen et al. commended PRISM and VESTA for their graphical user interfaces.

Based on the currently available tools, their limitations, and published case studies, we suggest the following heuristics for applying SMC in practice. If the system under analysis has dynamics that can be captured in a timed or hybrid model, then UPPAAL and its extensions for SMC [120] is the primary choice. For biological systems, PLASMA Lab and its biological language [98] are particularly relevant. Actor-based stochastic systems are best expressed using probabilistic rewriting logic [5], and then analyzed using PVESTA [102] and other tools in the Maude toolset. For general stochastic systems whose behavior can be programmatically described, we suggest using PRISM [100] or Ymer [125]; for general implicit or black box systems, we suggest MultiVESTA [92], Ymer, or PLASMA Lab.

8 DISCUSSION

Compared to approaches based on numerical methods, the main advantage of SMC is that large state spaces become feasible to analyze. A secondary advantage is that SMC supports even systems whose implementation details are unknown and cannot be simulated on demand. However, in such black box systems, whether the key SMC requirement of stochastic behavior is satisfied may not be easily determined by simple inspection of traces; one way is to apply statistical tests [71].

Model Estimation. Statistical tests can also be used to reject proposed stochastic models of systems, e.g., DTMCs produced by estimation. Of course, such estimation of probabilities will produce values within some error bounds. We may use *Interval-valued Discrete Time Markov Chains* (IDTMCs) to account for the estimation error. An IDTMC is essentially a family of DTMCs, each with a different transition probability function drawn from the IDTMC. In this case, each execution path needs to follow a sample DTMC drawn from a family of DTMCs. It is also possible that the probabilities in the systems themselves suffer from jitter around a distribution—thus requiring sampling from the space of probability bounds on each transition (rather than fixing it for each execution path as in IDTMCs). To capture such jitter, *Interval Markov Decision Processes* may be used to model the behavior. Analyzing such systems exactly is possible but computationally intractable [114]. SMC techniques treating the system as a black box may be effective for these cases, but further study is required.

Whether to use actual traces as sample points directly or use them to validate a model that is subsequently analyzed using SMC is largely a contextual decision. White box models have, e.g., the advantage of efficient support for specification formulas with nesting via combined statistical and numeric-symbolic techniques. On the other hand, the precision of the analysis may be illusory if the model is inaccurate.

Support for large state spaces and the possibility to rely on Bayesian methods make SMC an attractive choice for domains with well-known priors, such as systems biology. Similarly, the intractability of exhaustive analysis of stochastic timed and hybrid systems puts SMC tools in a favorable light. However, even less so than is the case in regular model checking, it is not clear that SMC analysis can be performed at the push of a button given that the tradeoffs between error probabilities and bounds in specification formulas must be weighed against required sample sizes and overhead of simulation.

For some stochastic systems, the interesting quantitative properties may concern *aggregate* behaviors rather than soft deadlines concerning specific actions. In this case, it is possible to forego SMC by modeling the *evolution of probability distributions* of local states [70]. Such modeling allows us to express properties such as bounds on transient rewards and expected values in execution paths—properties that are not expressible in logics such as PCTL. If the behavior of a system is Markovian, then it is possible to (non-stochastically) verify aggregate properties on execution paths using Euclidian model checking [3, 71]. Analyzing the evolution of probability distributions is more complicated if the underlying system is governed by an MDP, though it has been shown to be decidable for certain classes of MDPs and applied to a pharmacokinetic model [68].

Limitations of SMC. SMC shares the limitations of all formal methods as well as those of statistical techniques. The fact that the observations are likely given a hypothesis is no guarantee of the correctness of the hypothesis itself. In cases where the reliability and safety of a piece of software is at issue, we believe the best strategy is to assume a specification is false (i.e., that the system is buggy—that the specified probabilistic property does not hold) and then try to refute this hypothesis with a high probability. In this case, the only consequence of a Type II error is over-engineering of a system to increase its safety.

SMC, like all formal methods, should be thought of as another bug finding technique—not a guarantee of "correctness." Properties are tested with respect to a hypothesized model. For one, the model is itself usually an abstraction, which may fail to capture some essential element. For another, we are often interested in some abstract "meta-properties" such as safety and security. The properties we test are specific manifestations of these desirable attributes and usually not exhaustive. Often, new requirements will be discovered as systems are deployed and new use cases developed. Testing and verification, including for properties appropriate to test using SMC, must be a continual process. Moreover, in stochastic systems, the probability distribution over processes may not be stationary but may change over time. Learning techniques and incremental verification become useful in such cases.

Integrated Modeling and Verification. Functional and non-functional properties of systems are generally verified separately, e.g., with Markov chains and queueing networks for performance and nondeterministic transition systems for safety. A more attractive workflow is one where a common system model is used to seamlessly perform both functional and non-functional verification. Since models for functional verification may lack the required probabilities to mirror uncertainties and environmental constraints, one possibility is to allow optional decoration of transitions with such measures. The PMaude extension to the Maude rewriting logic tool has been used in this way to enable verification of performance properties of a distributed key-value store [84]. Moreover, because SMC works on both white box and black box systems, it can be used to establish correspondence between implementations and abstract models for the purposes of non-functional verification, in the way similar to that of model-based or property-based testing for functional correctness. Finally, due to the diverse requirements of practitioners, it may be warranted to support stochastic systems with interrelated components described in different formalisms (e.g., part DTMC, part MDP) within the same toolset.

Certified SMC. The last decade has seen considerable work on formalizing parts of measure theory and probability relevant to probabilistic programming languages and stochastic processes in proof assistants such as HOL4, Coq, and Isabelle/HOL. More recently, significant parts of PCTL model checking on DTMCs have been verified in Isabelle/HOL [58]. A natural extension to this line of work is to formalize in proof assistants all the theoretical underpinnings of SMC, primarily for estimation and hypothesis testing. In the case of Isabelle/HOL and DTMCs, significant parts of the theory are already available. However, it would be necessary to generate the required distributions from Bernoulli streams of bits, and then apply the existing theories for random variable analysis and DTMCs.

Ultimately, such a collection of theoretical results can be used in a certified toolchain for SMC. This toolchain may be developed in at least two different ways. First, all SMC components (simulation, property checking, estimation, and hypothesis testing) can be developed jointly and integrated and proved correct inside the proof assistant, and then extracted to executable code in a way similar to the CompCert verified C compiler [82]. Second, verified SMC components can be integrated with an existing tool such as PRISM. The latter approach may necessitate generation of certificates for results from unverified components to minimize the trusted base, but may benefit from existing performance optimizations.

Tool Standardization and Integration. More effective benchmark performance comparisons would be facilitated if tools supported the same syntax for systems and logics or provide trusted converters. The PRISM format exhibits signs in becoming the de facto standard, but shows up in many different incompatible variations across tools. In addition, practitioners would benefit if there was better support for custom discrete-event generators in existing tools. Jansen et al. [61]

additionally suggest that stochastic model checking tools should support calling other tools from the same graphical user interface by the simple push of a button.

While some tools support distribution of, e.g., sampling and testing on many computers in a network, some tools such as PRISM do not. It may be called for to standardize the interface for clients of stochastic model checkers. This would facilitate easier building of clusters for large-scale tool deployment.

Although Jansen et al. established a suite of benchmarks for stochastic model checking [61], SMC tools may require their own benchmark suite with a range that reflects typical application areas. In particular, the suite would benefit from benchmarks with complicated nested properties, unbounded untils, and rarely exhibited properties.

The semantics of model checking with hypothesis testing (which includes possibility of errors) has been defined for various stochastic logics by Younes [128]. However, while new stochastic logics are generally formalized in ideal settings, many lack a corresponding semantics at the level of SMC, i.e., one that describes the guarantees provided at the level of the language. In particular, having such semantics for both hypothesis testing and estimation methods for the same logic would allow SMC users to better understand the tradeoffs inherent in both approaches.

Algorithmic Improvements. As explained in Section 4.2, algorithms for handling unbounded until properties still leave much to desire. Moreover, nested and rare properties sometimes increase model checking time substantially. When using SMC in some tools, the expressiveness may be substantially reduced when compared to other techniques. Combinations of numeric-symbolic techniques and SMC may therefore be warranted to model check specific properties. However, since the verification semantics is usually different for each technique and combination, it is important that tools that implement combinations of algo rithms accurately represent results along with a description of the confidence of the result.

ACKNOWLEDGMENTS

The authors thank Johannes Hölzl and Sylvain Peyronnet for their comments, Atul Sandur for providing valuable feedback on earlier drafts of the article, and the anonymous reviewers for their comments and suggestions.

REFERENCES

[1] Alessandro Abate, Joost-Pieter Katoen, John Lygeros, and Maria Prandini. 2010. Approximate model checking of stochastic hybrid systems. *Eur. J. Control* 16, 6 (2010), 624–641. DOI: http://dx.doi.org/10.3166/ejc.16.624-641

[2] C. J. Adcock. 1997. Sample size determination: A review. *J. Roy. Stat. Soci. Ser. D (The Statistician)* 46, 2 (1997), 261–283. DOI: http://dx.doi.org/10.1111/1467-9884.00082

[3] Gul Agha. 2013. Euclidean model checking: A scalable method for verifying quantitative properties in probabilistic systems. In *Proceedings of the 5th International Conference on Algebraic Informatics (CAI'13)*, Traian Muntean, Dimitrios Poulakis, and Robert Rolland (Eds.). Springer, Berlin, Germany, 1–3. DOI: http://dx.doi.org/10.1007/978-3-642-40663-8_1

[4] Gul Agha, Carl Gunter, Michael Greenwald, Sanjeev Khanna, José Meseguer, Koushik Sen, and Prasanna Thati. 2005. Formal modeling and analysis of DoS using probabilistic rewrite theories. In *Proceedings of the International Workshop on Foundations of Computer Security (PCS'05)*.

[5] Gul Agha, José Meseguer, and Koushik Sen. 2006. PMaude: Rewrite-based specification language for probabilistic object systems. *Electron. Notes Theoret. Comput. Sci.* 153, 2 (2006), 213–239. DOI: http://dx.doi.org/10.1016/j.entcs.2005.10.040; *Proceedings of the Third Workshop on Quantitative Aspects of Programming Languages (QAPL'05)*.

[6] Musab AlTurki and José Meseguer. 2011. PVeStA: A parallel statistical model checking and quantitative analysis tool. In *Proceedings of the 4th International Conference on Algebra and Coalgebra in Computer Science (CALCO'11)*, Andrea Corradini, Bartek Klin, and Corina Cîrstea (Eds.). Springer, Berlin, Germany, 386–392. DOI: http://dx.doi.org/10.1007/978-3-642-22944-2_28

[7] APMC. 2005. APMC2 tool. Retrieved from http://berbiqui.org/apmc2/apmc-2.0.0.tar.gz.

[8] APMCBeta. 2014. APMC3 Beta tool. Retrieved from https://web.archive.org/web/20140928144328http://sylvain.berbiqui.org/apmc.

[9] Alexandre Arnold, Massimo Baleani, Alberto Ferrari, Marco Marazza, Valerio Senni, Axel Legay, Jean Quilbeuf, and Christoph Etzien. 2016. An application of SMC to continuous validation of heterogeneous systems. In *Proceedings of the 9th EAI International Conference on Simulation Tools and Techniques (SIMUTOOLS'16)*. 76–85. DOI: http://dx.doi.org/10.4108/eai.1-2-2017.152154

[10] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert Brayton. 1996. Verifying continuous time Markov chains. In *Proceedings of the 8th International Conference on Computer Aided Verification (CAV'96)*, Rajeev Alur and Thomas A. Henzinger (Eds.). Springer, Berlin, Germany, 269–276. DOI: http://dx.doi.org/10.1007/3-540-61474-5_75

[11] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert Brayton. 2000. Model-checking continuous-time Markov chains. *ACM Trans. Comput. Logic* 1, 1 (July 2000), 162–170. DOI: http://dx.doi.org/10.1145/343369.343402

[12] Soheib Baarir, Marco Beccuti, Davide Cerotti, Massimiliano De Pierro, Susanna Donatelli, and Giuliana Franceschinis. 2009. The GreatSPN tool: Recent enhancements. *SIGMETRICS Perform. Eval. Rev.* 36, 4 (March 2009), 4–9. DOI: http://dx.doi.org/10.1145/1530873.1530876

[13] Francis Bacon. 1902. *Novum Organum*. P. F. Collier & Sons.

[14] Christel Baier, Boudewijn Haverkort, Holger Hermanns, and Joost-Pieter Katoen. 2003. Model-checking algorithms for continuous-time Markov chains. *IEEE Trans. Softw. Eng.* 29, 6 (June 2003), 524–541. DOI: http://dx.doi.org/10.1109/TSE.2003.1205180

[15] Paolo Ballarini, Benoît Barbot, Marie Duflot, Serge Haddad, and Nihal Pekergin. 2015. HASL: A new approach for performance evaluation and model checking from concepts to experimentation. *Perform. Eval.* 90 (2015), 53–77. DOI: http://dx.doi.org/10.1016/j.peva.2015.04.003

[16] Benoît Barbot, Serge Haddad, and Claudine Picaronny. 2012. Coupling and importance sampling for statistical model checking. In *Proceedings of the 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12)*, Cormac Flanagan and Barbara König (Eds.). Springer, Berlin, Germany, 331–346. DOI: http://dx.doi.org/10.1007/978-3-642-28756-5_23

[17] Samik Basu, Arka P. Ghosh, and Ru He. 2009. Approximate model checking of PCTL involving unbounded path properties. In *Proceedings of the 11th International Conference on Formal Engineering Methods and Software Engineering (ICFEM'09)*, Karin Breitman and Ana Cavalcanti (Eds.). Springer, Berlin, Germany, 326–346. DOI: http://dx.doi.org/10.1007/978-3-642-10373-5_17

[18] Marco Beccuti and Giuliana Franceschinis. 2012. Efficient simulation of stochastic well-formed nets through symmetry exploitation. In *Proceedings of the Winter Simulation Conference (WSC'12)*. Winter Simulation Conference, Article 296, 13 pages. DOI: http://dx.doi.org/10.1109/WSC.2012.6465256

[19] Jonathan Bogdoll, Luis María Ferrer Fioriti, Arnd Hartmanns, and Holger Hermanns. 2011. Partial order methods for statistical model checking and simulation. In *Proceedings of the Joint 13th International Conference on Formal Techniques for Distributed Systems (FMOODS'11) and 30th IFIP WG 6.1 International Conference (FORTE'11)*, Roberto Bruni and Juergen Dingel (Eds.). Springer, Berlin, Germany, 59–74. DOI: http://dx.doi.org/10.1007/978-3-642-21461-5_4

[20] Jonathan Bogdoll, Arnd Hartmanns, and Holger Hermanns. 2012. Simulation and statistical model checking for modestly nondeterministic models. In *Proceedings of the 16th International GI/ITG Conference on Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance (MMB & DFT'12)*, Jens B. Schmitt (Ed.). Springer, Berlin, Germany, 249–252. DOI: http://dx.doi.org/10.1007/978-3-642-28540-0_20

[21] Dimitri Bohlender, Harold Bruintjes, Sebastian Junges, Jens Katelaan, Viet Yen Nguyen, and Thomas Noll. 2014. A review of statistical model checking pitfalls on real-time stochastic models. In *Proceedings of the 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications (ISoLA'14)*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer, Berlin, Germany, 177–192. DOI: http://dx.doi.org/10.1007/978-3-662-45231-8_13

[22] Henrik Bohnenkamp, Pedro R. D'Argenio, Holger Hermanns, and Joost-Pieter Katoen. 2006. MODEST: A compositional modeling formalism for hard and softly timed systems. *IEEE Trans. Softw. Eng.* 32, 10 (Oct 2006), 812–830. DOI: http://dx.doi.org/10.1109/TSE.2006.104

[23] Alexandre Borghi, Thomas Hérault, Richard Lassaigne, and Sylvain Peyronnet. 2008. Cell assisted APMC. In *Proceedings of the 5th International Conference on Quantitative Evaluation of Systems (QEST'08)*. 75–76. DOI: http://dx.doi.org/10.1109/QEST.2008.36

[24] Patricia Bouyer, Thomas Brihaye, and Nicolas Markey. 2006. Improved undecidability results on weighted timed automata. *Inform. Process. Lett.* 98, 5 (2006), 188–194. DOI: http://dx.doi.org/10.1016/j.ipl.2006.01.012

[25] Benoît Boyer, Kevin Corre, Axel Legay, and Sean Sedwards. 2013. PLASMA-lab: A flexible, distributable statistical model checking library. In *Proceedings of the 10th International Conference on Quantitative Evaluation of Systems (QEST'13)*, Kaustubh Joshi, Markus Siegle, Mariëlle Stoelinga, and Pedro R. D'Argenio (Eds.). Springer, Berlin, Germany, 160–164. DOI: http://dx.doi.org/10.1007/978-3-642-40196-1_12

[26] Peter Bulychev, Alexandre David, Kim Guldstrand Larsen, Marius Mikučionis, Danny Bøgsted Poulsen, Axel Legay, and Zheng Wang. 2012. UPPAAL-SMC: Statistical model checking for priced timed automata. In *Proceedings of the 10th Workshop on Quantitative Aspects of Programming Languages and Systems (Electronic Proceedings in Theoretical Computer Science)*, Herbert Wiklicky and Mieke Massink (Eds.), Vol. 85. Open Publishing Association, 1–16. DOI: http://dx.doi.org/10.4204/EPTCS.85.1

[27] Michaël Cadilhac, Thomas Hérault, Richard Lassaigne, Sylvain Peyronnet, and Sébastien Tixeuil. 2007. Evaluating complex MAC protocols for sensor networks with APMC. *Electron. Notes Theoret. Comput. Sci.* 185 (2007), 33–46. DOI: http://dx.doi.org/10.1016/j.entcs.2007.05.027; *Proceedings of the 6th International Workshop on Automated Verification of Critical Systems (AVoCS'06)*.

[28] Francesco Calzolai and Michele Loreti. 2010. Simulation and analysis of distributed systems in klaim. In *Proceedings of the 12th International Conference on Coordination Models and Languages (COORDINATION'10)*, Dave Clarke and Gul Agha (Eds.). Springer, Berlin, Germany, 122–136. DOI: http://dx.doi.org/10.1007/978-3-642-13414-2_9

[29] Quentin Cappart, Christophe Limbrée, Pierre Schaus, Jean Quilbeuf, Louis-Marie Traonouez, and Axel Legay. 2017. Verification of interlocking systems using statistical model checking. In *Proceedings of the 2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE'17)*. 61–68. DOI: http://dx.doi.org/10.1109/HASE.2017.10

[30] Frédéric Cérou and Arnaud Guyader. 2007. Adaptive multilevel splitting for rare event analysis. *Stochast. Anal. Appl.* 25, 2 (2007), 417–443. DOI: http://dx.doi.org/10.1080/07362990601139628

[31] Mounir Chadli, Jin Hyun Kim, Axel Legay, Louis-Marie Traonouez, Stefan Naujokat, Bernhard Steffen, and Kim Guldstrand Larsen. 2016. A model-based framework for the specification and analysis of hierarchical scheduling systems. In *Proceedings of the Joint 21st International Workshop on Formal Methods for Industrial Critical Systems and 16th International Workshop on Automated Verification of Critical Systems (FMICS-AVoCS'16)*, Maurice H. ter Beek, Stefania Gnesi, and Alexander Knapp (Eds.). Springer International Publishing, Cham, 133–141. DOI: http://dx.doi.org/10.1007/978-3-319-45943-1_9

[32] Y. S. Chow and Herbert Robbins. 1965. On the asymptotic theory of fixed-width sequential confidence intervals for the mean. *Ann. Math. Stat.* 36, 2 (1965), 457–462. DOI: http://dx.doi.org/10.1214/aoms/1177700156

[33] Edmund M. Clarke, E. Allen Emerson, and Aravinda Prasad Sistla. 1986. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.* 8, 2 (April 1986), 244–263. DOI: http://dx.doi.org/10.1145/5397.5399

[34] Edmund M. Clarke, James R. Faeder, Christopher J. Langmead, Leonard A. Harris, Sumit Kumar Jha, and Axel Legay. 2008. Statistical model checking in biolab: Applications to the automated analysis of T-cell receptor signaling pathway. In *Proceedings of the 6th International Conference on Computational Methods in Systems Biology (CMSB'08)*, Monika Heiner and Adelinde M. Uhrmacher (Eds.). Springer, Berlin, Germany, 231–250. DOI: http://dx.doi.org/10.1007/978-3-540-88562-7_18

[35] Edmund M. Clarke and Paolo Zuliani. 2011. Statistical model checking for cyber-physical systems. In *Proceedings of the 9th International Symposium on Automated Technology for Verification and Analysis (ATVA'11)*, Tevfik Bultan and Pao-Ann Hsiung (Eds.). Springer, Berlin, Germany, 1–12. DOI: http://dx.doi.org/10.1007/978-3-642-24372-1_1

[36] COSMOS. 2015. COSMOS tool. Retrieved from http://www.lsv.ens-cachan.fr/Software/cosmos/.

[37] Przemysław Daca, Thomas A. Henzinger, Jan Křetínský, and Tatjana Petrov. 2016. Faster statistical model checking for unbounded temporal properties. In *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'16)*, Marsha Chechik and Jean-François Raskin (Eds.). Springer, Berlin, Germany, 112–129. DOI: http://dx.doi.org/10.1007/978-3-662-49674-9_7

[38] Pedro D'Argenio, Axel Legay, Sean Sedwards, and Louis-Marie Traonouez. 2015. Smart sampling for lightweight verification of Markov decision processes. *Int. J. Softw. Tools Technol. Transfer* 17, 4 (2015), 469–484. DOI: http://dx.doi.org/10.1007/s10009-015-0383-0

[39] Alexandre David, Dehui Du, Kim G. Larsen, Axel Legay, Marius Mikučionis, Danny Bøgsted Poulsen, and Sean Sedwards. 2012. Statistical model checking for stochastic hybrid systems. In *Proceedings of the 1st International Workshop on Hybrid Systems and Biology (Electronic Proceedings in Theoretical Computer Science)*, Ezio Bartocci and Luca Bortolussi (Eds.), Vol. 92. Open Publishing Association, 122–136. DOI: http://dx.doi.org/10.4204/EPTCS.92.9

[40] Alexandre David, Kim G. Larsen, Axel Legay, Guangyuan Li, and Danny Bøgsted Poulsen. 2014. Quantified dynamic metric temporal logic for dynamic networks of stochastic hybrid automata. In *Proceedings of the 2014 14th International Conference on Application of Concurrency to System Design*. 32–41. DOI: http://dx.doi.org/10.1109/ACSD.2014.21

[41] Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikučionis, Danny Bøgsted Poulsen, Jonas Vliet, and Zheng Wang. 2011. Statistical model checking for networks of priced timed automata. In *Proceedings of the 9th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'11)*, Uli Fahrenberg and Stavros Tripakis (Eds.). Springer, Berlin, Germany, 80–96. DOI: http://dx.doi.org/10.1007/978-3-642-24310-3_7

[42] Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikučionis, and Zheng Wang. 2011. Time for statistical model checking of real-time systems. In *Proceedings of the 23rd International Conference on Formal Modeling and Analysis of Timed Systems (CAV'11)*, Ganesh Gopalakrishnan and Shaz Qadeer (Eds.). Springer, Berlin, Germany, 349–355. DOI: http://dx.doi.org/10.1007/978-3-642-22110-1_27

[43] Marie Duflot, Laurent Fribourg, Thomas Herault, Richard Lassaigne, Frédéric Magniette, Stéphane Messika, Sylvain Peyronnet, and Claudine Picaronny. 2005. Probabilistic model checking of the CSMA/CD protocol using PRISM and APMC. *Electron. Notes Theoret. Comput. Sci.* 128, 6 (2005), 195–214. DOI: http://dx.doi.org/10.1016/j.entcs.2005.04.012; *Proceedings of the 4th International Workshop on Automated Verification of Critical Systems (AVoCS'04)*.

[44] Diana El Rabih and Nihal Pekergin. 2009. Statistical model checking using perfect simulation. In *Proceedings of the 7th International Symposium on Automated Technology for Verification and Analysis (ATVA'09)*, Zhiming Liu and Anders P. Ravn (Eds.). Springer, Berlin, Germany, 120–134. DOI: http://dx.doi.org/10.1007/978-3-642-04761-9_11

[45] GreatSPN. 2012. GreatSPN tool. Retrieved from http://www.di.unito.it/ greatspn/index.html.

[46] Radu Grosu and Scott A. Smolka. 2005. Monte carlo model checking. In *Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05)*, Nicolas Halbwachs and Lenore D. Zuck (Eds.). Springer, Berlin, Germany, 271–286. DOI: http://dx.doi.org/10.1007/978-3-540-31980-1_18

[47] Stefan Hadjis and Stefano Ermon. 2015. Importance sampling over sets: A new probabilistic inference scheme. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, Marina Meila and Tom Heskes (Eds.). AUAI Press, Corvallis, OR, 355–364. Retrieved from http://auai.org/uai2015/proceedings/papers/143.pdf.

[48] Hans Hansson and Bengt Jonsson. 1994. A logic for reasoning about time and reliability. *Formal Aspects Comput.* 6, 5 (1994), 512–535. DOI: http://dx.doi.org/10.1007/BF01211866

[49] Klaus Havelund and Grigore Roşu. 2002. Synthesizing monitors for safety properties. In *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'02)*, Joost-Pieter Katoen and Perdita Stevens (Eds.). Springer, Berlin, Germany, 342–356. DOI: http://dx.doi.org/10.1007/3-540-46002-0_24

[50] Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. 2000. On the use of model checking techniques for dependability evaluation. In *Proceedings of the 19th IEEE Symposium on Reliable Distributed Systems (SRDS'00)*. 228–237. DOI: http://dx.doi.org/10.1109/RELDI.2000.885410

[51] Monika Heiner, Christian Rohr, and Martin Schwarick. 2013. MARCIE—Model checking and reachability analysis done efficiently. In *Proceedings of the 34th International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS'13)*, José-Manuel Colom and Jörg Desel (Eds.). Springer, Berlin, Germany, 389–399. DOI: http://dx.doi.org/10.1007/978-3-642-38697-8_21

[52] Monika Heiner, Christian Rohr, Martin Schwarick, and Stefan Streif. 2010. A comparative study of stochastic analysis techniques. In *Proceedings of the 8th International Conference on Computational Methods in Systems Biology (CMSB'10)*. ACM, New York, NY, 96–106. DOI: http://dx.doi.org/10.1145/1839764.1839776

[53] David Henriques, João G. Martins, Paolo Zuliani, André Platzer, and Edmund M. Clarke. 2012. Statistical model checking for markov decision processes. In *Proceedings of the 2012 9th International Conference on Quantitative Evaluation of Systems (QEST'12)*. 84–93. DOI: http://dx.doi.org/10.1109/QEST.2012.19

[54] Thomas Hérault, Richard Lassaigne, Frédéric Magniette, and Sylvain Peyronnet. 2004. Approximate probabilistic model checking. In *Proceedings of the 5th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'04)*, Bernhard Steffen and Giorgio Levi (Eds.). Springer, Berlin, Germany, 73–84. DOI: http://dx.doi.org/10.1007/978-3-540-24622-0_8

[55] Thomas Hérault, Richard Lassaigne, and Sylvain Peyronnet. 2006. APMC 3.0: Approximate verification of discrete and continuous time markov chains. In *Proceedings of the 3rd International Conference on the Quantitative Evaluation of Systems (QEST'06)*. 129–130. DOI: http://dx.doi.org/10.1109/QEST.2006.5

[56] Holger Hermanns, Joachim Meyer-Kayser, and Markus Siegle. 1999. Multi terminal binary decision diagrams to represent and analyse continuous time markov chains. In *Proceedings of the 3rd International Workshop on the Numerical Solution of Markov Chains*. 188–207.

[57] Wassily Hoeffding. 1963. Probability inequalities for sums of bounded random variables. *J. Amer. Stat. Assoc.* 58, 301 (1963), 13–30. DOI: http://dx.doi.org/10.1080/01621459.1963.10500830

[58] Johannes Hölzl and Tobias Nipkow. 2012. Verifying pCTL model checking. In *Proceedings of the18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12)*, Cormac Flanagan and Barbara König (Eds.). Springer, Berlin, Germany, 347–361. DOI: http://dx.doi.org/10.1007/978-3-642-28756-5_24

[59] Oliver C. Ibe and Kishor S. Trivedi. 1990. Stochastic Petri net models of polling systems. *IEEE J. Select. Areas Commun.* 8, 9 (Dec 1990), 1649–1657. DOI: http://dx.doi.org/10.1109/49.62852

[60] Raj Jain. 1991. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, Hoboken, NJ.

[61] David N. Jansen, Joost-Pieter Katoen, Marcel Oldenkamp, Mariëlle Stoelinga, and Ivan Zapreev. 2008. How fast and fat is your probabilistic model checker? An experimental performance comparison. In *Hardware and Software: Verification and Testing*, Karen Yorav (Ed.). Lecture Notes in Computer Science, Vol. 4899. Springer, Berlin, Germany, 69–85. DOI: http://dx.doi.org/10.1007/978-3-540-77966-7_9

[62] Cyrille Jegourel, Axel Legay, and Sean Sedwards. 2013. Importance splitting for statistical model checking rare properties. In *Proceedings of the 25th International Conference on Computer Aided Verification (CAV'13)*, Natasha Sharygina and Helmut Veith (Eds.). Springer, Berlin, Germany, 576–591. DOI: http://dx.doi.org/10.1007/978-3-642-39799-8_38

[63] Cyrille Jegourel, Axel Legay, and Sean Sedwards. 2014. An effective heuristic for adaptive importance splitting in statistical model checking. In *Proceedings of the 6th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications (ISoLA'14)*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer, Berlin, Germany, 143–159. DOI: http://dx.doi.org/10.1007/978-3-662-45231-8_11

[64] Cyrille Jegourel, Axel Legay, and Sean Sedwards. 2016. Command-based importance sampling for statistical model checking. *Theoret. Comput. Sci.* 649 (2016), 1–24. DOI: http://dx.doi.org/10.1016/j.tcs.2016.08.009

[65] Sumit K. Jha, Edmund M. Clarke, Christopher J. Langmead, Axel Legay, André Platzer, and Paolo Zuliani. 2009. A Bayesian approach to model checking biological systems. In *Proceedings of the 7th International Conference on Computational Methods in Systems Biology (CMSB'09)*, Pierpaolo Degano and Roberto Gorrieri (Eds.). Springer, Berlin, Germany, 218–234. DOI: http://dx.doi.org/10.1007/978-3-642-03845-7_15

[66] Sumit Kumar Jha and Christopher James Langmead. 2012. Exploring behaviors of stochastic differential equation models of biological systems using change of measures. *BMC Bioinfo.* 13, 5 (12 Apr 2012), S8. DOI: http://dx.doi.org/10.1186/1471-2105-13-S5-S8

[67] Kenan Kalajdzic, Cyrille Jegourel, Anna Lukina, Ezio Bartocci, Axel Legay, Scott A. Smolka, and Radu Grosu. 2016. Feedback control for statistical model checking of cyber-physical systems. In *Proceedings of the 7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques (ISoLA'16)*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer International Publishing, Cham, 46–61. DOI: http://dx.doi.org/10.1007/978-3-319-47166-2_4

[68] Vijay Anand Korthikanti, Mahesh Viswanathan, Gul Agha, and YoungMin Kwon. 2010. Reasoning about MDPs as transformers of probability distributions. In *Proceedings of the 7th International Conference on the Quantitative Evaluation of Systems (QEST'10)*. 199–208. DOI: http://dx.doi.org/10.1109/QEST.2010.35

[69] Marta Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV'11)*, Ganesh Gopalakrishnan and Shaz Qadeer (Eds.). Springer, Berlin, Germany, 585–591. DOI: http://dx.doi.org/10.1007/978-3-642-22110-1_47

[70] Youngmin Kwon and Gul Agha. 2011. Verifying the evolution of probability distributions governed by a DTMC. *IEEE Trans. Softw. Eng.* 37, 1 (Jan 2011), 126–141. DOI: http://dx.doi.org/10.1109/TSE.2010.80

[71] Youngmin Kwon and Gul Agha. 2013. Performance evaluation of sensor networks by statistical modeling and euclidean model checking. *ACM Trans. Sen. Netw.* 9, 4, Article 39 (July 2013), 38 pages. DOI: http://dx.doi.org/10.1145/2489253.2489256

[72] David Kyle, Jeffery Hansen, and Sagar Chaki. 2015. Statistical model checking of distributed adaptive real-time software. In *Proceedings of the 6th International Conference on Runtime Verification (RV'15)*, Ezio Bartocci and Rupak Majumdar (Eds.). Springer International Publishing, Cham, 269–274. DOI: http://dx.doi.org/10.1007/978-3-319-23820-3_17

[73] Tze Leung Lai. 2001. Sequential analysis: Some classical problems and new challenges. *Statistica Sinica* 11, 2 (2001), 303–351. DOI: http://dx.doi.org/10.2307/24306854

[74] Kim G. Larsen and Arne Skou. 1991. Bisimulation through probabilistic testing. *Info. Comput.* 94, 1 (1991), 1–28. DOI: http://dx.doi.org/10.1016/0890-5401(91)90030-6

[75] Richard Lassaigne and Sylvain Peyronnet. 2002. Approximate verification of probabilistic systems. In *Proceedings of the 2nd Joint International Workshop on Process Algebra and Probabilistic Methods: Performance Modeling and Verification (PAPM-PROBMIV'02)*, Holger Hermanns and Roberto Segala (Eds.). Springer, Berlin, Germany, 213–214. DOI: http://dx.doi.org/10.1007/3-540-45605-8_16

[76] Richard Lassaigne and Sylvain Peyronnet. 2008. Probabilistic verification and approximation. *Ann. Pure Appl. Logic* 152, 13 (2008), 122–131. DOI: http://dx.doi.org/10.1016/j.apal.2007.11.006; *Proceedings of the 12th Workshop on Logic, Language, Information and Computation.*

[77] Richard Lassaigne and Sylvain Peyronnet. 2015. Approximate planning and verification for large Markov decision processes. *Int. J. Softw. Tools Technol. Transfer* 17, 4 (2015), 457–467. DOI: http://dx.doi.org/10.1007/s10009-014-0344-z

[78] Pierre L'Ecuyer, Valérie Demers, and Bruno Tuffin. 2007. Rare events, splitting, and quasi-monte carlo. *ACM Trans. Model. Comput. Simul.* 17, 2, Article 9 (April 2007). DOI: http://dx.doi.org/10.1145/1225275.1225280

[79] Axel Legay, Sean Sedwards, and Louis-Marie Traonouez. 2015. Scalable verification of Markov decision processes. In *Proceedings of the Workshops: HOFM, SAFOME, OpenCert, MoKMaSD, and WS-FMDS*, Carlos Canal and Akram Idani (Eds.). Springer International Publishing, Cham, 350–362. DOI: http://dx.doi.org/10.1007/978-3-319-15201-1_23

[80] Axel Legay, Sean Sedwards, and Louis-Marie Traonouez. 2016. Rare events for statistical model checking an overview. In *Proceedings of the 10th International Workshop on Reachability Problems (RP'16)*, Kim Guldstrand Larsen, Igor Potapov, and Jiří Srba (Eds.). Springer International Publishing, Cham, 23–35. DOI: http://dx.doi.org/10.1007/978-3-319-45994-3_2

[81] Axel Legay and Louis-Marie Traonouez. 2016. Statistical model checking with change detection. In *Transactions on Foundations for Mastering Change I*, Bernhard Steffen (Ed.). Springer International Publishing, Cham, 157–179. DOI: http://dx.doi.org/10.1007/978-3-319-46508-1_9

[82] Xavier Leroy. 2009. Formal verification of a realistic compiler. *Commun. ACM* 52, 7 (2009), 107–115. DOI: http://dx.doi.org/10.1145/1538788.1538814

[83] Dennis V. Lindley. 1997. The choice of sample size. *J. Roy. Stat. Soc. Ser. D (The Statistician)* 46, 2 (1997), 129–138. DOI: http://dx.doi.org/10.1111/1467-9884.00068

[84] Si Liu, Son Nguyen, Jatin Ganhotra, Muntasir Raihan Rahman, Indranil Gupta, and José Meseguer. 2015. Quantitative analysis of consistency in NoSQL key-value stores. In *Proceedings of the 12th International Conference on Quantitative Evaluation of Systems (QEST'15)*, Javier Campos and R. Boudewijn Haverkort (Eds.). Springer International Publishing, Cham, 228–243. DOI: http://dx.doi.org/10.1007/978-3-319-22264-6_15

[85] Anna Lukina, Lukas Esterle, Christian Hirsch, Ezio Bartocci, Junxing Yang, Ashish Tiwari, Scott A. Smolka, and Radu Grosu. 2017. ARES: Adaptive receding-horizon synthesis of optimal plans. In *Proceedings of the 23rd International Conference on Quantitative Evaluation of Systems (TACAS'17)*, Axel Legay and Tiziana Margaria (Eds.). Springer, Berlin, Germany, 286–302. DOI: http://dx.doi.org/10.1007/978-3-662-54580-5_17

[86] Oded Maler. 2016. Some thoughts on runtime verification. In *Proceedings of the 16th International Conference, on Runtime Verification (RV'16)*, Yliès Falcone and César Sánchez (Eds.). Springer International Publishing, Cham, 3–14. DOI: http://dx.doi.org/10.1007/978-3-319-46982-9_1

[87] MARCIE. 2017. MARCIE tool. Retrieved from http://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/Marcie.

[88] João Martins, André Platzer, and João Leite. 2011. Statistical model checking for distributed probabilistic-control hybrid automata with smart grid applications. In *Proceedings of the 13th International Conference on Formal Engineering Methods and Software Engineering (ICFEM'11)*, Shengchao Qin and Zongyan Qiu (Eds.). Springer, Berlin, Germany, 131–146. DOI: http://dx.doi.org/10.1007/978-3-642-24559-6_11

[89] José Meseguer and Raman Sharykin. 2006. Specification and analysis of distributed object-based stochastic hybrid systems. In *Proceedings of the 9th International Workshop on Hybrid Systems: Computation and Control (HSCC'06)*, João P. Hespanha and Ashish Tiwari (Eds.). Springer, Berlin, Germany, 460–475. DOI: http://dx.doi.org/10.1007/11730637_35

[90] MODES. 2006. MODES tool. Retrieved from http://www.modestchecker.net.

[91] MRMC. 2011. MRMC tool. Retrieved from http://www.mrmc-tool.org.

[92] MultiVESTA. 2015. MultiVESTA tool. Retrieved from http://sysma.imtlucca.it/tools/multivesta/.

[93] NIST. 2017. NIST/SEMATECH e-Handbook of Statistical Methods. Retrieved from http://www.itl.nist.gov/div898/handbook/ppc/section3/ppc333.htm.

[94] Ayoub Nouri, Saddek Bensalem, Marius Bozga, Benoît Delahaye, Cyrille Jegourel, and Axel Legay. 2014. Statistical model checking QoS properties of systems with SBIP. *Int. J. Softw. Tools Technol. Transfer* 17, 2 (2014), 171–185. DOI: http://dx.doi.org/10.1007/s10009-014-0313-6

[95] Sucheendra K. Palaniappan, Benjamin M. Gyori, Bing Liu, David Hsu, and P. S. Thiagarajan. 2013. Statistical model checking based calibration and analysis of bio-pathway models. In *Proceedings of the 11th International Conference on Computational Methods in Systems Biology (CMSB'13)*, Ashutosh Gupta and Thomas A. Henzinger (Eds.). Springer, Berlin, Germany, 120–134. DOI: http://dx.doi.org/10.1007/978-3-642-40708-6_10

[96] Danhua Peng, Roland Ewald, and Adelinde M. Uhrmacher. 2014. Towards semantic model composition via experiments. In *Proceedings of the 2nd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM PADS'14)*. ACM, New York, NY, 151–162. DOI: http://dx.doi.org/10.1145/2601381.2601394

[97] A. G. Phatak and N. M. Bhatt. 1967. Estimation of the fraction defective in curtailed sampling plans by attributes. *Technometrics* 9, 2 (1967), 219–228. DOI: http://dx.doi.org/10.2307/1266419

[98] PLASMA. 2015. PLASMA Lab. Retrieved from http://project.inria.fr/plasma-lab/.

[99] Karl Popper. 1968. *Conjectures and Refutations: The Growth of Scientific Knowledge*. Harper Torchbooks.

[100] PRISM. 2017. PRISM Statistical Model Checker. Retrieved from http://www.prismmodelchecker.org/manual/RunningPRISM/StatisticalModelChecking.

[101] James Gary Propp and David Bruce Wilson. 1996. Exact sampling with coupled markov chains and applications to statistical mechanics. *Rand. Struct. Algor.* 9, 1–2 (Aug. 1996), 223–252. DOI: http://dx.doi.org/10.1002/(SICI)1098-2418(199608/09)9:1/2⟨223::AID-RSA14⟩3.0.CO;2-O

[102] PVESTA. 2011. PVESTA tool. Retrieved from http://maude.cs.uiuc.edu/tools/pvesta/.

[103] Jean Quilbeuf, Everton Cavalcante, Louis-Marie Traonouez, Flavio Oquendo, Thais Batista, and Axel Legay. 2016. A logic for the statistical model checking of dynamic software architectures. In *Proceedings of the 7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques (ISoLA'16)*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer International Publishing, Cham, 806–820. DOI: http://dx.doi.org/10.1007/978-3-319-47166-2_56

[104] Daniël Reijsbergen, Pieter-Tjerk de Boer, Werner Scheinhardt, and Boudewijn Haverkort. 2010. Rare event simulation for highly dependable systems with fast repairs. In *Proceedings of the 2010 7th International Conference on the Quantitative Evaluation of Systems*. 251–260. DOI: http://dx.doi.org/10.1109/QEST.2010.39

[105] Nima Roohi and Mahesh Viswanathan. 2015. Statistical model checking for unbounded until formulas. *Int. J. Softw. Tools Technol. Transfer* 17, 4 (2015), 417–427. DOI: http://dx.doi.org/10.1007/s10009-015-0368-z

[106] SAM. 2010. SAM: Stochastic Analyser for Mobility. Retrieved from http://rap.dsi.unifi.it/SAM/.

[107] Sriram Sankaranarayanan and Georgios Fainekos. 2012. Simulating insulin infusion pump risks by in-silico modeling of the insulin-glucose regulatory system. In *Proceedings of the 10th International Conference on Computational Methods in Systems Biology (CMSB'12)*, David Gilbert and Monika Heiner (Eds.). Springer, Berlin, Germany, 322–341. DOI: http://dx.doi.org/10.1007/978-3-642-33636-2_19

[108] SBIP. 2013. SBIP tool. Retrieved from http://www-verimag.imag.fr/Statistical-Model-Checking.html.

[109] Stefano Sebastio and Andrea Vandin. 2013. MultiVeStA: Statistical model checking for discrete event simulators. In *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools (ValueTools'13)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, 310–315. DOI: http://dx.doi.org/10.4108/icst.valuetools.2013.254377

[110] Koushik Sen, Abhay Vardhan, Gul Agha, and Grigore Roşu. 2004. Efficient decentralized monitoring of safety in distributed systems. In *Proceedings of the 26th International Conference on Software Engineering*. 418–427. DOI: http://dx.doi.org/10.1109/ICSE.2004.1317464

[111] Koushik Sen, Mahesh Viswanathan, and Gul Agha. 2004. Statistical model checking of black-box probabilistic systems. In *Computer Aided Verification*, Rajeev Alur and Doron A. Peled (Eds.). Lecture Notes in Computer Science, Vol. 3114. Springer, Berlin, Germany, 202–215. DOI: http://dx.doi.org/10.1007/978-3-540-27813-9_16

[112] Koushik Sen, Mahesh Viswanathan, and Gul Agha. 2005. On statistical model checking of stochastic systems. In *Computer Aided Verification*, Kousha Etessami and Sriram K. Rajamani (Eds.). Lecture Notes in Computer Science, Vol. 3576. Springer, Berlin, Germany, 266–280. DOI: http://dx.doi.org/10.1007/11513988_26

[113] Koushik Sen, Mahesh Viswanathan, and Gul Agha. 2005. VESTA: A statistical model-checker and analyzer for probabilistic systems. *Proceedings of the International Conference on Quantitative Evaluation of Systems*. 251–252. DOI: http://dx.doi.org/10.1109/QEST.2005.42

[114] Koushik Sen, Mahesh Viswanathan, and Gul Agha. 2006. Model-checking markov chains in the presence of uncertainties. In *Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, Holger Hermanns and Jens Palsberg (Eds.). Springer, Berlin, Germany, 394–410. DOI: http://dx.doi.org/10.1007/11691372_26

[115] Rajan Srinivasan. 2002. *Importance Sampling*. Springer, Berlin, Germany.

[116] Lisa J. Strug, Charles A. Rohde, and Paul N. Corey. 2007. An introduction to evidential sample size calculations. *Amer. Stat.* 61, 3 (2007), 207–212. Retrieved from http://www.jstor.org/stable/27643895.

[117] Alexander Tartakovsky, Igor Nikiforov, and Michèle Basseville. 2014. *Sequential Analysis: Hypothesis Testing and Changepoint Detection*. Chapman and Hall/CRC, Boca Raton, FL.

[118] Philip S. Thomas and Emma Brunskill. 2017. Importance sampling with unequal support. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*. AAAI Press, Palo Alto, CA. Retrieved from https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14957/14457.

[119] Kishor S. Trivedi. 2008. *Probability & Statistics With Reliability, Queuing And Computer Science Applications* (2nd ed.). Wiley, Hoboken, NJ.

[120] UPPAAL 2015. UPPAAL. Retrieved from http://www.uppaal.org.

[121] Abraham Wald. 1945. Sequential tests of statistical hypotheses. *Ann. Math. Stat.* 16, 2 (1945), 117–186. DOI: http://dx.doi.org/10.1214/aoms/1177731118

[122] Abraham Wald. 1950. *Statistical Decision Functions*. Wiley, New York, NY.

[123] Abraham Wald. 1992. *Breakthroughs in Statistics: Foundations and Basic Theory*. Springer New York, New York, NY, Chapter Sequential Tests of Statistical Hypotheses, 256–298. DOI: http://dx.doi.org/10.1007/978-1-4612-0919-5_18

[124] D. J. White. 1993. A survey of applications of markov decision processes. *J. Oper. Res. Soc.* 44, 11 (1993), 1073–1096. DOI: http://dx.doi.org/10.2307/2583870

[125] Ymer. 2015. Ymer tool. Retrieved from http://www.tempastic.org/ymer/.

[126] Håkan L. S. Younes. 2005. Probabilistic verification for "black-box" systems. In *Computer Aided Verification*, Kousha Etessami and Sriram K. Rajamani (Eds.). Lecture Notes in Computer Science, Vol. 3576. Springer, Berlin, Germany, 253–265. DOI: http://dx.doi.org/10.1007/11513988_25

[127] Håkan L. S. Younes. 2005. Ymer: A statistical model checker. In *Computer Aided Verification*, Kousha Etessami and Sriram K. Rajamani (Eds.). Lecture Notes in Computer Science, Vol. 3576. Springer, Berlin, Germany, 429–433. DOI: http://dx.doi.org/10.1007/11513988_43

[128] Håkan L. S. Younes. 2006. Error control for probabilistic model checking. In *Proceedings of the 7th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'06)*, E. Allen Emerson and Kedar S. Namjoshi (Eds.). Springer, Berlin, Germany, 142–156. DOI: http://dx.doi.org/10.1007/11609773_10

[129] Håkan L. S. Younes, Edmund M. Clarke, and Paolo Zuliani. 2011. Statistical verification of probabilistic properties with unbounded until. In *Proceedings of the 13th Brazilian Symposium on Formal Methods: Foundations and Applications (SBMF'10)*. Springer, Berlin, Germany, 144–160. DOI: http://dx.doi.org/10.1007/978-3-642-19829-8_10

[130] Håkan L. S. Younes, Marta Kwiatkowska, Gethin Norman, and David Parker. 2006. Numerical vs. statistical probabilistic model checking. *Int. J. Softw. Tools Technol. Transfer* 8, 3 (2006), 216–228. DOI: http://dx.doi.org/10.1007/s10009-005-0187-8

[131] Håkan L. S. Younes and Reid G. Simmons. 2002. Probabilistic verification of discrete event systems using acceptance sampling. In *Computer Aided Verification*, Ed Brinksma and Kim Guldstrand Larsen (Eds.). Lecture Notes in Computer Science, Vol. 2404. Springer, Berlin, Germany, 223–235. DOI: http://dx.doi.org/10.1007/3-540-45657-0_17

[132] Håkan L. S. Younes and Reid G. Simmons. 2006. Statistical probabilistic model checking with a focus on time-bounded properties. *Info. Comput.* 204, 9 (2006), 1368–1409. DOI: http://dx.doi.org/10.1016/j.ic.2006.05.002

[133] Paolo Zuliani. 2015. Statistical model checking for biological applications. *Int. J. Softw. Tools Technol. Transfer* 17, 4 (2015), 527–536. DOI: http://dx.doi.org/10.1007/s10009-014-0343-0

[134] Paolo Zuliani, André Platzer, and Edmund M. Clarke. 2013. Bayesian statistical model checking with application to stateflow/simulink verification. *Formal Methods Syst. Design* 43, 2 (2013), 338–367. DOI: http://dx.doi.org/10.1007/s10703-013-0195-3

Received February 2017; revised September 2017; accepted November 2017