Statistical Model Checking : An Overview

Axel Legay[1], Benoît Delahaye[1], Saddek Bensalem[2]

[1] INRIA/IRISA, Rennes, France
[2] Verimag Laboratory, Université Joseph Fourier Grenoble, CNRS

Abstract. Quantitative properties of stochastic systems are usually specified in logics that allow one to compare the measure of executions satisfying certain temporal properties with thresholds. The model checking problem for stochastic systems with respect to such logics is typically solved by a numerical approach [31, 8, 35, 22, 21, 5] that iteratively computes (or approximates) the exact measure of paths satisfying relevant subformulas; the algorithms themselves depend on the class of systems being analyzed as well as the logic used for specifying the properties. Another approach to solve the model checking problem is to *simulate* the system for finitely many executions, and use *hypothesis testing* to infer whether the samples provide a *statistical* evidence for the satisfaction or violation of the specification. In this tutorial, we survey the statistical approach, and outline its main advantages in terms of efficiency, uniformity, and simplicity.

1 Introduction and Context

Quantitative properties of stochastic systems are usually specified in logics that allow one to compare the measure of executions satisfying certain temporal properties with thresholds. The model checking problem for stochastic systems with respect to such logics is typically solved by a numerical approach that iteratively computes (or approximates) the exact measure of paths satisfying relevant subformulas. The algorithm for computing such measures depends on the class of stochastic systems being considered as well as the logics used for specifying the correctness properties. Model checking algorithms for a variety of contexts have been discovered [2, 13, 8] and there are mature tools (see e.g. [25, 7]) that have been used to analyze a variety of systems in practice.

Despite the great strides made by numerical model checking algorithms, there are many challenges. Numerical algorithms work only for special systems that have certain structural properties. Further the algorithms require a lot of time and space, and thus scaling to large systems is a challenge. Finally, the logics for which model checking algorithms exist are extensions of classical temporal logics, which are often not the most popular among engineers.

Another way to verify quantitative properties is to use a simulation-based approach. The key idea is to deduce whether or not the system satisfies the property by observing some of its executions with a monitoring procedure [1, 19], and use *hypothesis testing* to infer whether the samples provide a *statistical*

evidence for the satisfaction or violation of the specification [42]. Of course, in contrast to a numerical approach, a simulation-based solution does not guarantee a correct result. However, it is possible to bound the probability of making an error. Simulation-based methods are known to be far less memory and time intensive than numerical ones, and are sometimes the only option [41].

The crux of the statistical model checking approach is that since sample executions of a stochastic system are drawn according to the distribution defined by the system, they can be used to get estimates of the probability measure on executions. Starting from time-bounded Probabilistic Computational Tree Logic properties [42], the technique has been extended to handle properties with unbounded until operators [33], as well as to black-box systems [32, 37]. Tools based on this idea have been built [34, 39], and they have been used to analyze many systems.

This approach enjoys many advantages. First, these algorithms only require that the system be executable (or rather, sample executions be drawn according to the measure space defined by the system). Thus, it can be applied to larger class of systems than numerical model checking algorithms including black-box systems and infinite state systems. Second the approach can be generalized to a larger class of properties, including Fourier transform based logics. Finally, the algorithm is easily parallelizable, which can help scale to large systems. However, the statistical approach also has some disadvantages when compared with the numerical approach. First, it only provides probabilistic guarantees about the correctness of the algorithms answer. Next, the sample size grows very large if the model checker's answer is required to be highly accurate. Finally, the statistical approach only works for purely probabilistic systems, i.e., those that do not have any nondeterminism. Furthermore, since statistical tests are used to determine the correctness of a system, the approach only works for systems that "robustly" satisfy a given property, i.e., the actual measure of paths satisfying a given subformula, is bounded away from the thresholds to which it is compared in the specification.

In this tutorial, we will overview existing statistical model checking algorithms and discuss their efficiency. We will also overview existing tools and case studies and discuss future work.

2 Our Objective

We consider a stochastic system $\mathcal{S}$ and a property ϕ. An *execution* of $\mathcal{S}$ is a possibly infinite sequence of states of $\mathcal{S}$. Our objective is to solve the *probabilistic model checking problem*, i.e., to decide whether $\mathcal{S}$ satisfies ϕ with a probability greater or equal to a certain threshold θ. The latter is denoted $S \models P_{\geq\theta}(\phi)$, where P is called a *probabilistic operator*. This paper will overview solutions to this problem. These solutions depend on the nature of $\mathcal{S}$ and ϕ. We consider three cases.

1. We first assume that $\mathcal{S}$ is a *white-box system*, i.e., that one can generate as much executions of the system as we want. We also assume that ϕ does

not contain probabilistic operators. In Section 3, we recall basic statistical algorithms that can be used to verify bounded properties (i.e., properties that can be monitored on fixed-length execution) of white-box systems.
2. In Section 4, we discuss extensions to the full probabilistic computation tree logic[8]. There, we consider the case where ϕ can also contain probabilistic operators and the case where it has to be verified on infinite executions.
3. In Section 5, we briefly discuss the verification of *black-box systems*, i.e. systems for which a part of the probability distribution is unknown.

In Section 6, we will present various experiments that show that (1) statistical model checking algorithms can be more efficient than numerical ones, and (2) statistical model checking algorithms can be applied to solve problems that are beyond the scope of numerical methods. Finally, Section 7 discusses our vision of the future of statistical model checking.

Remark 1. The objective of the tutorial is not to feed the reader with technical details, but rather to introduce statistical model checking, and outline its main advantages in terms of efficiency, uniformity, and simplicity.

Remark 2. There are other techniques that allow to estimate the probability for $\mathcal{S}$ to satisfy ϕ. These approaches (that are based on Monte-Carlo techniques) will not be covered in this paper. The interested reader is redirected to [17, 20, 26] for more details.

Remark 3. Statistical Model Checking also applies to non stochastic systems [17]. This topic will not be covered in this tutorial.

3 Statistical Model Checking : The Beginning

In this section, we overview several statistical model checking techniques. We assume that $\mathcal{S}$ is a white-box system and that ϕ is a bounded property. By bounded properties, we mean properties that can be defined on finite executions of the system. In general, the length of such executions has to be precomputed.

Let B_i be a discrete random variable with a Bernoulli distribution of parameter p. Such a variable can only take two values 0 and 1 with $Pr[B_i = 1] = p$ and $Pr[B_i = 0] = 1 - p$. In our context, each variable B_i is associated with one simulation of the system. The outcome for B_i, denoted b_i, is 1 if the simulation satisfies ϕ and 0 otherwise. To make sure that the above approach works, one has to make sure that one can get the result of any experiment in a finite amount of time. In general, this means that we are considering bounded properties, i.e., properties that can be decided on finite executions.

Remark 4. All the results presented in this section are well-known mathematical results coming from the area of statistics. As we shall see, these results are sufficient to verify bounded properties of a large class of systems. As those properties are enough in many practical applications, one could wonder whether the contribution of the computer scientist should not be at the practical level rather than at the theoretical one.

Before going further one should answer one last question: *"What is the class of models that can be considered?"* In fact, statistical model checking can be applied to any stochastic system and logic on which one can define a *probability space* for the property under consideration. Hence, the approach provides a uniform way for the verification of a wide range of temporal logic properties over various stochastic models, including Markov Chains or Continuous Timed Markov Chains [35, 3, 2]. In general, it is not necessary to make the hypothesis that the system has the Markovian property[3], except when working with nested formulas (see Section 4). It is worth mentioning that the technique cannot be applied to systems that combine both nondeterministic and stochastic aspects (such as Markov Decision Processes). Indeed, the simulation-based approach could not distinguish between the probability distributions that are sampled.

3.1 Qualitative Answer using Statistical Model Checking

The main approaches [38, 32] proposed to answer the qualitative question are based on *hypothesis testing*. Let $p = Pr(\phi)$, to determine whether $p \geq \theta$, we can test $H : p \geq \theta$ against $K : p < \theta$. A test-based solution does not guarantee a correct result but it is possible to bound the probability of making an error. The *strength* (α, β) of a test is determined by two parameters, α and β, such that the probability of accepting K (respectively, H) when H (respectively, K) holds, called a Type-I error (respectively, a Type-II error) is less or equal to α (respectively, β).

A test has *ideal performance* if the probability of the Type-I error (respectively, Type-II error) is exactly α (respectively, β). However, these requirements make it impossible to ensure a low probability for both types of errors simultaneously (see [38] for details). A solution to this problem is to relax the test by working with an *indifference region* (p_1, p_0) with $p_0 \geq p_1$ ($p_0 - p_1$ is the *size of the region*). In this context, we test the hypothesis $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$ instead of H against K. If the value of p is between p_1 and p_0 (the indifference region), then we say that the probability is sufficiently close to θ so that we are indifferent with respect to which of the two hypotheses K or H is accepted. The thresholds p_0 and p_1 are generally defined in term of the single threshold θ, e.g., $p_1 = \theta - \delta$ and $p_0 = \theta + \delta$. We now need to provide a test procedure that satisfies the requirements above. In the next two subsections, we recall two solutions proposed by Younes in [38, 43].

Single Sampling Plan. To test H_0 against H_1, we specify a constant c. If $\sum_{i=1}^{n} b_i$ is larger than c, then H_0 is accepted, else H_1 is accepted. The difficult part in this approach is to find values for the pair (n, c), called a *single sampling plan (SSP in short)*, such that the two error bounds α and β are respected. In practice, one tries to work with the smallest value of n possible so as to minimize the number of simulations performed. Clearly, this number has to be greater if α and β are

[3] The Markovian property ensures that the probability to go from a state s to a next state only depends on s, not on the states that have been visited before reaching s.

smaller but also if the size of the indifference region is smaller. This results in an optimization problem, which generally does not have a closed-form solution except for a few special cases [38]. In his thesis [38], Younes proposes a binary search based algorithm that, given p_0, p_1, α, β, computes an approximation of the minimal value for c and n.

Remark 5. There are many variants of this algorithm. As an example, in [33], Sen et al. proposes to accept H_0 if $\frac{(\sum_{i=1}^{n} b_i)}{n} \geq p$. Here, the difficulty is to find a value for n such that the error bounds are valid.

Sequential probability ratio test. The sample size for a single sampling plan is fixed in advance and independent of the observations that are made. However, taking those observations into account can increase the performance of the test. As an example, if we use a single plan (n, c) and the $m > c$ first simulations satisfy the property, then we could (depending on the error bounds) accept H_0 without observing the $n - m$ other simulations. To overcome this problem, one can use the *sequential probability ratio test (SPRT in short)* proposed by Wald [36]. The approach is briefly described below.

In SPRT, one has to choose two values A and B ($A > B$) that ensure that the strength of the test is respected. Let m be the number of observations that have been made so far. The test is based on the following quotient:

$$\frac{p_{1m}}{p_{0m}} = \prod_{i=1}^{m} \frac{Pr(B_i = b_i \mid p = p_1)}{Pr(B_i = b_i \mid p = p_0)} = \frac{p_1^{d_m}(1-p_1)^{m-d_m}}{p_0^{d_m}(1-p_0)^{m-d_m}}, \tag{1}$$

where $d_m = \sum_{i=1}^{m} b_i$. The idea behind the test is to accept H_0 if $\frac{p_{1m}}{p_{0m}} \geq A$, and H_1 if $\frac{p_{1m}}{p_{0m}} \leq B$. The SPRT algorithm computes $\frac{p_{1m}}{p_{0m}}$ for successive values of m until either H_0 or H_1 is satisfied; the algorithm terminates with probability 1[36]. This has the advantage of minimizing the number of simulations. In his thesis [38], Younes proposed a logarithmic based algorithm SPRT that given p_0, p_1, α and β implements the sequential ratio testing procedure.

Discussion. Computing ideal values A_{id} and B_{id} for A and B in order to make sure that we are working with a test of strength (α, β) is a laborious procedure (see Section 3.4 of [36]). In his seminal paper [36], Wald showed that if one defines $A_{id} \geq A = \frac{(1-\beta)}{\alpha}$ and $B_{id} \leq B = \frac{\beta}{(1-\alpha)}$, then we obtain a new test whose strength is (α', β'), but such that $\alpha' + \beta' \leq \alpha + \beta$, meaning that either $\alpha' \leq \alpha$ or $\beta' \leq \beta$. In practice, we often find that both inequalities hold. This is illustrated with the following example taken from [38].

Example 1. Let $p_0 = 0.5$, $p_1 = 0.3$, $\alpha = 0.2$ and $\beta = 0.1$. If we use $A_{id} \geq A = \frac{(1-\beta)}{\alpha}$ and $B_{id} \leq B = \frac{\beta}{(1-\alpha)}$, then we are guaranteed that $\alpha' \leq 0.222$ and $\beta' \leq 0.125$. Through computer simulation (reproducing the same experiments 10000 of time), we observe that $\alpha' \leq 0.175$ and $\beta' \leq 0.082$. So the strength of the test is in reality better than the theoretical assumption.

3.2 Some Generalities Regarding Efficiency

The efficiency of the above algorithms is characterized by the number of simulations needed to obtain an answer as well as the time it costs to compute a simulation. The latter often depends on the property under verification. Both numbers are *expected numbers* as they change from executions to executions and can only be estimated (see [38] for an explanation). However, some generalities are known. For example, it is known that, except for some situations, SPRT is always faster than SSP. When $\theta = 1$ (resp. $\theta = 0$) SPRT degenerates to SSP; it is not a problem since SSP is known to be optimal for such values. Observe that the time complexity of statistical model checking is independent from the state-space and that the space complexity is of the order of the state space. Also, the expected number of simulations for SSP is logarithmic with respect to α and β and linear with respect to the indifference region; for SPRT, the number depends on the probability distribution p.

Remark 6. A very relevant discussion on complexity of statistical model checking can be found in Section 5.4 of [38].

4 Statistical Model Checking: Next Step

In the previous section, we have proposed statistical model checking algorithms for verifying bounded properties of white-box systems. In this section, we go one step further and consider three nontrivial extensions that are:

1. The nested case, i.e., the case where ϕ can also contain probabilistic operators. As an example, we can write the following property $P_{\geq\theta_1}(q \Rightarrow P_{\geq\theta_2}(\phi_2))$
2. The unbounded case, i.e., the case where ϕ cannot be decide on a finite execution. Here we will restrict ourselves to the until property. Given two formulas ϕ_1 and ϕ_2, the until operator ensures that ϕ_1 is true until ϕ_2 has been seen.
3. Boolean combinations of formulae, i.e., formulae of the form: $P_{\geq\theta_1}(\phi_1) \wedge P_{\geq\theta_2}(\phi_2)$.

We will only survey existing results and give pointers to the relevant papers.

4.1 The Unbounded Case: Until

We are now concerned with the verification of the *until property*. This property requires that a property ϕ_1 remains valid until a property ϕ_2 has been seen. The problem is that we do not know a priori the moment when ϕ_2 will be satisfied. Hence, one has to reason on infinite execution. There are two works on this topics, one by Sen et al.[33] and one more recent work by Pekergin et al. [30]. We will not give details on these works, but the reader should know that Sen works by extending the model with extra probabilities, which makes the solution extremely slow. Pekergin uses a new technique that is based on

perfect simulation. According to [30], this technique is not only faster than Sen's one, but also more general as it allows to study the steady-state operator for continuous timed Markov Chains.

Remark 7. Contrary to the numerical results [35, 5] the above results are not sufficient to verify properties of the form $P_{\geq\theta}(\phi)$, where ϕ is a property expressed in Linear Temporal Logic [29]. Incomplete results regarding the verification of these properties with simulation-based techniques can be found in [20, 17].

4.2 Nested Probability Operators

We consider the problem of checking whether $\mathcal{S}$ satisfies ϕ with a probability greater or equal to θ. However, contrary to what we have been doing so far, we will now assume that ϕ cannot be decided on a single execution, i.e., we will assume that ϕ is of the form $P_{\geq\theta_1}\phi_1$. So, where is the difficulty? The difficulty is that ϕ cannot be model checked on a single execution, but rather depends on another test. Hence, we have to provide a way to nest tests. In his thesis, Younes proposed the following theorem.

Theorem 1. *Let $\psi = P_{\geq\theta}(\phi)$ be a property and assume that ϕ can be verified with Type-I error α' and Type-II error β', then ψ can be verified with Type-I error α and Type-II error β, assuming that the indifference region is of size at least $((\theta+\delta)(1-\alpha'), (1-(1-(\theta-\delta)))(1-\beta')$.*

Hence one has to find a compromise between the size of the indifference region of the inner test and the outer one. There are two interesting facts to know about nested operators:

1. Even for bounded properties, the above result (and in fact, any result in the literature [33, 38, 37, 39]) only works for systems that have the Markovian property.
2. In practice, the complexity (in term of number of sampling) becomes exponential in the number of tests.

Remark 8. An interesting research direction would be to study the link with probabilistic testing [27].

4.3 Boolean Combinations

We have to consider two operations, namely conjunction and negation (as it is known that any Boolean combination reduces to combinations of these two operators). We recall some results provided by Younes. We start with conjunction.

Theorem 2. *Let ψ be the conjunction of n properties $\phi_1, \ldots, \phi_2$. Assume that each ϕ_i can be decided with Type-I error α_i and Type-II error β_I. Then ϕ can be decided with Type-I error $min_i(\alpha_i)$ and Type-II error $max_i(\beta_i)$.*

The idea behind the proof of the theorem is that

1. If we claim that the conjunction is not satisfied, this means that we have deduced that one of the operands is not.
2. If we claim that the conjunction is satisfied, this means that we have concluded that all the operands are satisfied. As we may have made mistakes in each individual verification, we get $\max_i(\beta_i)$.

For negation, the result is provided by the following theorem.

Theorem 3. *To verify a formula $\neg\psi$ with Type-I error α and Type-II error β, it is sufficient to verify ψ with Type-I error β and Type-II error α.*

5 Black-box Systems: a Note

Black-box Systems is an interesting class of stochastic systems whose treatment is beyond the scope of numerical techniques. Roughly speaking, a black-box systems is simply a system whose probability distribution (i.e., set of behaviors) is not totally known and cannot be observed. Hence, one can view a black-box system as a finite set of executions pre-computed and for which no information is available.

In the context of such systems, Type errors and indifference region cannot play a role. Indeed, those parameters influence the number of simulations that can be computed, but here the simulations are given and you cannot compute more!

A solution to this problem is to conduct a SSP test assuming that the parameter n is fixed to the number of simulations that are given in advance. The difficulty is to chose the constant c in such a way that it becomes roughly equal to accept H_0 or H_1 if $\theta = p$. In his thesis [38] and in [40], Younes proposed a solution to the problem. He also shown that a previous solution proposed by Sen [32] is not correct.

There are techniques to verify nested formulas over black-box systems. However, a technique for the verification of unbounded properties is still needed.

6 Tools and Experiments

Any statistical model checking toolset is build by combining 1) a monitoring procedure to decide whether a finite execution satisfies the property under consideration, 2) a statistical model checking algorithm, and 3) a tool that allows to describe a system and generate sets of executions.

The two firsts tools that implemented statistical model checking algorithms are *Ymer*[39] and *Vesta*[34]. *Vesta* implements a variation of the single sampling plan algorithm. The choice of implementing the SSP algorithm is motivated by the fact that it is easier to parallelize as the number of simulations to perform is known in advance. However, in his thesis, Younes showed that sequential algorithms are also easily parallelizable. *Ymer* is limited to bounded properties while *Vesta* also incorporate the unbounded until. In [22], the authors conducted several experiments that tend to show that (1) *Ymer* is faster than *Vesta* and (2)

Vesta makes more false positive (selecting the bad hypothesis) than *Ymer*. Regarding the unbounded case, it seems that *Vesta* is not very efficient and can make a lot of false positive. Both *Vesta* and *Ymer* have been applied to huge case studies. A comparison of *Ymer* and *Vesta* to established tools such as *PRISM* [25] can be found in [22].

Both *Ymer* and *Vesta* as well as their successors [23, 24] focus on the verification of classical stochastic extension of temporal logics. In a series of recent work, we have shown that statistical model checking can also be used in other contexts that are clearly beyond the scope of existing tools. This topic is the subject of the next subsections.

6.1 Verifying Circuits

In [9, 10], we applied SPRT to verifying properties of *mixed-signal circuits*, i.e., circuits for which there is an interaction between analog (continuous) and digital (discrete) values. Our first contribution was to propose a version of stochastic discrete-time event systems that fits into the framework introduced by Younes with the additional advantage that it explicitly handles analog and digital signals. We also introduced *probabilistic signal linear temporal logic*, a logic adapted to the specification of properties for mixed-signal circuits in the *temporal* domain and in the *frequency* domain. Our second contribution was the analysis of a $\Delta-\Sigma$ modulator. A $\Delta-\Sigma$ modulator is an efficient *Analog-to-Digital Converter circuit*, i.e., a device that converts analog signals into digital signals. A common critical issue in this domain is the analysis of the *stability* of the internal state variables of the circuit. The concern is that the values that are stored by these variables can grow out of control until reaching a maximum value, at which point we say that the circuit *saturates*. Saturation is commonly assumed to compromise the quality of the analog-to-digital conversion. In [14] and [18] reachability techniques developed in the area of hybrid systems are used to analyze the stability of a third-order modulator. Their idea is to use such techniques to guarantee that for *every* input signal in a given range, the states of the system remain stable. While this reachability-based approach is sound, it has important drawbacks such as (1) signals with long duration cannot be practically analyzed, and (2) properties that are commonly specified in the frequency domain rather than in the time domain cannot be checked. Our results show that a simulation-based approach makes it possible to handle properties and signals that are beyond the scope of the reachability-based approach. As an example, in our experiments, we analyze discrete-time signals with 24000 sampling points in seconds, while the approach in [14] takes hours to analyze signals with up to 31 sampling points. We are also able to provide insight into a question left open in [14] by observing that saturation does not always imply an improper signal conversion. This can be done by comparing the Fourier transform of each of the input analog signals with the Fourier transform of its corresponding digital signal. Such a property can easily be expressed in our logic and Model Checked with our simulation-based approach. We are unaware of other formal verification techniques that can solve

this problem. Indeed, numerical techniques cannot reason on an execution at a time.

6.2 Systems Biology

In [11], we considered the verification of complex biological systems. we introduced a new tool, called BioLab, for *formally* reasoning about the behavior of stochastic dynamic models by integrating SPRT into the BioNetGen [15, 16] framework for rule-based modeling. We then used BioLab to verify the stochastic bistability of T-cell signalling. Our results have recently been extended to take prior knowledge on the model into account [6].

Remark 9. Statistical model checking techniques recently received a lot of attention in the area of systems biology. As an example, Carnegie Mellon University was awarded a $10.000.000 grant for applying such techniques in the medical area [12].

6.3 Heterogeneous applications

In [4], we have proposed to apply statistical model checking techniques to the verification of *heterogeneous applications*. Systems integrating multiple heterogeneous distributed applications communicating over a shared network are typical in various sensitive domains such as aeronautic or automotive embedded systems. Verifying the correctness of a particular application inside such a system is known to be a challenging task, which is often beyond the scope of existing exhaustive validation techniques.

In our paper, we proposed to exploit the structure of the system in order to increase the efficiency of the verification process. The idea is conceptually simple: instead of performing an analysis of the entire system, we proposed to analyze each application separately, but under some particular context/execution environment. This context is a *stochastic abstraction* that represents the interactions with other applications running within the system and sharing the computation and communication resources. The idea is to build such a context automatically by simulating the system and learning the probability distributions of key characteristics impacting the functionality of the given application. The abstraction can easily be analyzed with statistical model checking techniques.

The overall contribution of our study is an application of the above method on an industrial case study, the *heterogeneous communication system* (HCS for short) deployed for cabin communication in a civil airplane. HCS is an heterogeneous system providing entertainment services (ex : audio/video on passengers demand) as well as administrative services (ex: cabin illumination, control, audio announcements), which are implemented as distributed applications running in parallel, across various devices within the plane and communicating through a common Ethernet-based network. The HCS system has to guarantee stringent requirements, such as reliable data transmission, fault tolerance, timing and synchronization constraints. An important requirement is the *accuracy of clock*

synchronization between different devices. This latter property states that the difference between the clocks of any two devices should be bounded by a small constant, which is provided by the user and depends on his needs (for example, to guarantee the fiability of another service). Hence, one must be capable to compute the smallest bound for which synchronization occurs and compare it with the bound expected by the user. Unfortunately, due to the large number of heterogeneous components that constitute the system, deriving such a bound manually from the textual specification is an unfeasible task. In this paper, we propose a formal approach that consists in building a formal model of the HCS, then we apply simulation-based algorithms to this model in order to deduce the smallest value of the bound for which synchronization occurs. We start with a fixed value of the bound and check whether synchronization occurs. If yes, then we make sure that this is the best one. If no, we restart the experiment with a new value.

We have been able to derive precise bounds that guarantee proper synchronization for all the devices of the system. We also computed the probability to satisfy the property for smaller values of the bound, i.e., bounds that do not satisfy the synchronization property with probability 1. Being able to provide such an information is of clear importance, especially when the best bound is too high with respect to user's requirements. We have observed that the values we obtained strongly depend on the position of the device in the network. We also estimated the average and worst proportion of failures per simulation for bounds that are smaller than the one that guarantees synchronization. Checking this latter property has been made easy because statistical model checking allows us to reason on one execution at a time. Finally, we have also considered the influence of clock drift on the synchronisation results. The experiments highlight the generality of our technique, which could be applied to other versions of the HCS as well as to other heterogeneous applications.

7 The Future of Statistical Model Checking

There are various directions for future research in the statistical model checking area. Here is a list of possible topics.

- Using efficient techniques for performing simulation is crucial to guarantee good performances for any statistical model checking algorithm. Unfortunately, the existing algorithms do not exploit efficient simulation techniques. It would thus be worth combining statistical model checking algorithms with such techniques (example : rare-event simulations, , ...). This is a huge implementation effort which also requires to define a methodology to select the good simulation technique to be applied.
- Statistical model checking algorithms have not yet been applied to the verification of multi-core systems, this area should be investigated.
- Statistical model checking algorithms do not apply to systems that combine both stochastic and non deterministic aspects. Extending the results to

such systems is however crucial to perform verification of security protocols, networking protocols, and performance protocols.
- Statistical model checking algorithms reduce to decide between two hypothesis. In many areas, especially systems biology, we may have a prior knowledge on the probability to satisfy each hypothesis. Incorporating this prior knowledge in the verification process may considerably reduce the number of simulations needed for the algorithm to terminate.
- Statistical model checking algorithms suppose that the property ϕ can be checked on finite executions of the system. There are however many situations where ϕ cannot be checked in a finite amount of time. This is for example the case when ϕ is a long-run average or a steady state property. In systems biology, we are clearly interested in the study of such properties.
- Verifying applications running within a huge heterogeneous system without is a challenging problem. In a recent work [4], the authors have proposed a new simulation-based technique for solving such problem. The technique starts by performing simulations of the system in order to learn the context in where the application is used. Then, it creates a stochastic abstraction for the application, which takes the context information into account. Up to know, there is no automatic way to learn the context and derive the stochastic context. However, what we have observed so far is that it often takes the form of properties that cannot be expressed in classical temporal logic. Hence, statistical model checking may be our last resort to analyze the resulting abstraction.
- Statistical model checking may help testers. In [28], Cavalli et al. proposed to use statistical techniques for conformance testing of timed stochastic systems. The technique should be automated. This could lead to new algorithms for verifying the so-called black-box systems.

Acknowledgments

We would like to thanks our collaborators on the statistical model checking project: Sumit Jha, and Marius Bozga.

References

1. A. B. 0002, M. Leucker, and C. Schallhart. Monitoring of real-time properties. In *FSTTCS*, volume 4337 of *LNCS 4337*, pages 260–272. Springer, 2006.
2. C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE Trans. Software Eng.*, 29(6):524–541, 2003.
3. C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
4. A. Basu, S. Bensalem, M. Bozga, B. Caillaud, B. Delahaye, and A. Legay. Statistical abstraction and model-checking of large heterogeneous systems. Technical report, INRIA, 2010.

5. D. Bustan, S. Rubin, and M. Y. Vardi. Verifying omega-regular properties of markov chains. In *Proc. 16th Int. Conference on Computer Aided Verification (CAV)*, volume 3114 of *Lecture Notes in Computer Science*, pages 189–201. Springer, 2004.
6. Carnegie Mellon University. *A Bayesian Approach to Model Checking Biological Systems*, 2009. under submission.
7. F. Ciesinski and C. Baier. Liquor: A tool for qualitative and quantitative linear time analysis of reactive systems. In *QEST*, pages 131–132. IEEE, 2006.
8. F. Ciesinski and M. Größer. On probabilistic computation tree logic. In *Validation of Stochastic Systems*, LNCS, 2925, pages 147–188. Springer, 2004.
9. E. M. Clarke, A. Donzé, and A. Legay. Statistical model checking of mixed-analog circuits with an application to a third order delta-sigma modulator. In *HVC*, volume 5394 of *LNCS*, pages 149–163. Springer, 2008.
10. E. M. Clarke, A. Donzé, and A. Legay. On simulation-based probabilistic model checking of mixed-analog circuits. *Formal Methods in System Design*, 2009. to appear.
11. E. M. Clarke, J. R. Faeder, C. J. Langmead, L. A. Harris, S. K. Jha, and A. Legay. Statistical model checking in biolab: Applications to the automated analysis of t-cell receptor signaling pathway. In *CMSB*, volume 5307 of *Lecture Notes in Computer Science*, pages 231–250. Springer, 2008.
12. Cmacs. http://cmacs.cs.cmu.edu/.
13. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.
14. T. Dang, A. Donze, and O. Maler. Verification of analog and mixed-signal circuits using hybrid systems techniques. In A. J. Hu and A. K. Martin, editors, *FMCAD'04 - Formal Methods for Computer Aided Design*, LNCS 3312, pages 21–36. Springer-Verlag, 2004.
15. J. R. Faeder, M. L. Blinov, and W. S. Hlavacek. Graphical rule-based representation of signal-transduction networks. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 133–140, New York, NY, USA, 2005. ACM.
16. J. R. Faeder, M. L. Blinov, and W. S. Hlavacek. Rule-based modeling of biochemical systems with BioNetGen. In I. V. Maly, editor, *Systems Biology*, Methods in Molecular Biology. Humana Press, Totowa, NJ, 2008.
17. R. Grosu and S. A. Smolka. Monte carlo model checking. In *TACAS*, volume 3440 of *Lecture Notes in Computer Science*, pages 271–286. Springer, 2005.
18. S. Gupta, B. H. Krogh, and R. A. Rutenbar. Towards formal verification of analog designs. In *ICCAD*, pages 210–217, 2004.
19. K. Havelund and G. Rosu. Synthesizing monitors for safety properties. In *Proc. of 11th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 2280 of *Lecture Notes in Computer Science*, pages 342–356. Springer, 2002.
20. T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. In *VMCAI*, volume 2937 of *Lecture Notes in Computer Science*, pages 73–84. Springer, 2004.
21. H. Hermanns, B. Wachter, and L. Zhang. Probabilistic cegar. In *CAV*, volume 5123 of *Lecture Notes in Computer Science*, pages 162–175. Springer, 2008.
22. D. N. Jansen, J. Katoen, M. Oldenkamp, M. Stoelinga, and I. S. Zapreev. How fast and fat is your probabilistic model checker? an experimental performance comparison. In *Haifa Verification Conference*, volume 4899 of *Lecture Notes in Computer Science*, pages 69–85. Springer, 2008.

23. J.-P. Katoen and I. S. Zapreev. Simulation-based ctmc model checking: An empirical evaluation. In *Proc. of 6th Int. Conference on the Quantitative Evaluation of Systems (QEST)*, pages 31–40. IEEE Computer Society, 2009.
24. J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen. The ins and outs of the probabilistic model checker mrmc. In *Proc. of 6th Int. Conference on the Quantitative Evaluation of Systems (QEST)*, pages 167–176. IEEE Computer Society, 2009.
25. M. Z. Kwiatkowska, G. Norman, and D. Parker. Prism 2.0: A tool for probabilistic model checking. In *QEST*, pages 322–323. IEEE, 2004.
26. S. Laplante, R. Lassaigne, F. Magniez, S. Peyronnet, and M. de Rougemont. Probabilistic abstraction for model checking: An approach based on property testing. *ACM Trans. Comput. Log.*, 8(4), 2007.
27. K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Inf. Comput.*, 94(1):1–28, 1991.
28. M. G. Merayo, I. Hwang, M. Núñez, and A. R. Cavalli. A statistical approach to test stochastic and probabilistic systems. In *On Proc.11th International Conference on Formal Engineering Methods*, volume 5885 of *Lecture Notes in Computer Science*, pages 186–205. Springer, 2009.
29. A. Pnueli. The temporal logic of programs. In *Proc. 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 46–57, 1977.
30. D. E. Rabih and N. Pekergin. Statistical model checking using perfect simulation. In *Proc. 7th Int. Conference on Automated Technology for Verification and Analysis (ATVA)*, volume 5799 of *Lecture Notes in Computer Science*, pages 120–134. Springer, 2009.
31. J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems,* P. Panangaden and F. van Breugel (eds.), volume 23 of *CRM Monograph Series*. American Mathematical Society, 2004.
32. K. Sen, M. Viswanathan, and G. Agha. Statistical model checking of black-box probabilistic systems. In *CAV*, LNCS 3114, pages 202–215. Springer, 2004.
33. K. Sen, M. Viswanathan, and G. Agha. On statistical model checking of stochastic systems. In *CAV*, LNCS 3576, pages 266–280, 2005.
34. K. Sen, M. Viswanathan, and G. A. Agha. Vesta: A statistical model-checker and analyzer for probabilistic systems. In *QEST*, pages 251–252. IEEE Computer Society, 2005.
35. M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *FOCS*, pages 327–338, 1985.
36. A. Wald. sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, 1945.
37. H. L. S. Younes. Probabilistic verification for "black-box" systems. In *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 253–265. Springer, 2005.
38. H. L. S. Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events*. PhD thesis, Carnegie Mellon, 2005.
39. H. L. S. Younes. Ymer: A statistical model checker. In *CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 429–433. Springer, 2005.
40. H. L. S. Younes. Error control for probabilistic model checking. In *VMCAI*, LNCS 3855, pages 142–156. springer-verlag, 2006.
41. H. L. S. Younes, M. Z. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking. *STTT*, 8(3):216–228, 2006.

42. H. L. S. Younes and R. G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *CAV*, LNCS 2404, pages 223–235. Springer, 2002.
43. H. L. S. Younes and R. G. Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Information and Computation*, 204(9):1368–1409, 2006.