Form Methods Syst Des (2013) 43:338–367
DOI 10.1007/s10703-013-0195-3

Bayesian statistical model checking with application to Stateflow/Simulink verification

Paolo Zuliani · André Platzer · Edmund M. Clarke

Published online: 27 August 2013

Abstract We address the problem of model checking stochastic systems, i.e., checking whether a stochastic system satisfies a certain temporal property with a probability greater (or smaller) than a fixed threshold. In particular, we present a Statistical Model Checking (SMC) approach based on Bayesian statistics. We show that our approach is feasible for a certain class of hybrid systems with stochastic transitions, a generalization of Simulink/Stateflow models. Standard approaches to stochastic discrete systems require numerical solutions for large optimization problems and quickly become infeasible with larger state spaces. Generalizations of these techniques to hybrid systems with stochastic effects are even more challenging. The SMC approach was pioneered by Younes and Simmons in the discrete and non-Bayesian case. It solves the verification problem by combining randomized sampling of system traces (which is very efficient for Simulink/Stateflow) with hypothesis testing (i.e., testing against a probability threshold) or estimation (i.e., computing with high probability a value close to the true probability). We believe SMC is essential for scaling up to large Stateflow/Simulink models. While the answer to the verification problem is not guaranteed to be correct, we prove that Bayesian SMC can make the probability of giving a wrong answer arbitrarily small. The advantage is that answers can usually be obtained much faster than with standard, exhaustive model checking techniques. We apply our Bayesian SMC approach to a representative example of stochastic discrete-time hybrid system models in Stateflow/Simulink: a fuel control system featuring hybrid behavior and fault tolerance. We show that our technique enables faster verification than state-of-the-art statistical techniques. We emphasize that Bayesian SMC is by no means restricted to State-

P. Zuliani (✉)
School of Computing Science, Newcastle University, Newcastle NE1 7RU, UK
e-mail: paolo.zuliani@ncl.ac.uk

A. Platzer · E.M. Clarke
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, USA

A. Platzer
e-mail: aplatzer@cs.cmu.edu

E.M. Clarke
e-mail: emc@cs.cmu.edu

flow/Simulink models. It is in principle applicable to a variety of stochastic models from other domains, e.g., systems biology.

Keywords Probabilistic verification · Hybrid systems · Stochastic systems · Statistical model checking · Hypothesis testing · Estimation

1 Introduction

Stochastic effects arise naturally in hybrid systems, for example, because of uncertainties present in the system environment (e.g., the reliability of sensor readings and actuator effects in control systems, the impact of timing inaccuracies, the reliability of communication links in a wireless sensor network, or the rate of message arrivals on an aircraft's communication bus). Uncertainty can often be modeled via a probability distribution, thereby resulting in a stochastic system, i.e., a system which exhibits probabilistic behavior. This raises the question of how to verify that a stochastic system satisfies a given probabilistic property. For example, we want to know whether the probability of an engine controller failing to provide optimal fuel/air ratio is smaller than 0.001; or whether the ignition succeeds within 1 ms with probability at least 0.99. In fact, several temporal logics have been developed in order to express these and other types of probabilistic properties [1, 3, 23]. The *Probabilistic Model Checking* (PMC) problem is to decide whether a stochastic model satisfies a temporal logic property with a probability greater than or equal to a certain threshold. More formally, suppose $\mathcal{M}$ is a stochastic model over a set of states S with the starting state s_0, ϕ is a formula in temporal logic, and $\theta \in (0, 1)$ is a probability threshold. The PMC problem is to decide algorithmically whether $\mathcal{M} \models P_{\geq\theta}(\phi)$, i.e., to decide whether the model $\mathcal{M}$ starting from its initial state s_0 satisfies the property ϕ with probability at least θ. In this paper, property ϕ is expressed in Bounded Linear Temporal Logic (BLTL), a variant of LTL [38] in which the temporal operators are equipped with upper time bounds. Alternatively, BLTL can be viewed as a sublogic of Koymans' Metric Temporal Logic [30]. As system models $\mathcal{M}$, we use a stochastic version of hybrid systems modeled in Stateflow/Simulink.

Existing algorithms for solving the PMC problem fall into one of two categories. The first category comprises numerical methods that can compute the probability that the property holds with high precision (e.g., [2, 3, 11, 13, 22, 31]). Numerical methods are generally only suitable for finite-state systems of about 10^7–10^8 states [32]. In real control systems, the number of states easily exceeds this limit, which motivates the need for algorithms for solving the PMC problem in a probabilistic fashion, such as Statistical Model Checking (SMC). These techniques heavily rely on simulation which, especially for large, complex systems, is generally easier and faster than a full symbolic study of the system. This can be an important factor for industrial systems designed using efficient simulation tools like Stateflow/Simulink. Since all we need for SMC are simulations of the system, we neither have to translate system models into separate verification tool languages, nor have to build symbolic models of the system (e.g., Markov chains) appropriate for numerical methods. This simplifies and speeds up the overall verification process. The most important question, however, is what information can be concluded from the behavior observed in simulations about the overall probability that ϕ holds for $\mathcal{M}$. The key for this are statistical techniques based on fair (iid = independent and identically distributed) sampling of system behavior.

Statistical Model Checking treats the PMC problem as a statistical inference problem, and solves it by randomized sampling of the *traces* (or simulations) from the model. Each sample trace is model checked to determine whether the BLTL property ϕ holds, and the

```
Input  : PBLTL property $P_{\geqslant\theta}(\phi)$, acceptance threshold $T \geqslant 1$, prior density $g$ for
         (unknown) probability $p$ that the system satisfies $\phi$
Output: "$H_0 : p \geqslant \theta$ accepted", or "$H_1 : p < \theta$ accepted"
1  $n := 0$;                                   {number of traces drawn so far}
2  $x := 0$;                           {number of traces satisfying $\phi$ so far}
3  loop
4      $\sigma :=$ draw a sample trace of the system (iid);              {Sect. 2}
5      $n := n + 1$;
6      if $\sigma \models \phi$ then                                     {Sect. 3}
7          $x := x + 1$
8      end
9      $\mathcal{B} :=$ BayesFactor$(n, x)$;                             {Sect. 5}
10     if $(\mathcal{B} > T)$ then return "$H_0$ accepted";
11     if $(\mathcal{B} < \frac{1}{T})$ then return "$H_1$ accepted"
12 end loop;
```

Algorithm 1: Statistical Model Checking by Bayesian Hypothesis Testing

number of satisfying traces that are found by randomized sampling is used to decide whether $\mathcal{M} \models P_{\geq\theta}(\phi)$. This decision is made by means of either estimation or hypothesis testing. In the first case one seeks to *estimate probabilistically* (i.e., compute with high probability a value close to) the probability that the property holds and then compare that estimate to θ [25, 41] (in statistics such estimates are known as *confidence inter*[illegible]. In the second case, the PMC problem is directly treated as a *hypothesis testing* problem [29, 41, 49], i.e., deciding between the null hypothesis $H_0 : \mathcal{M} \models P_{\geq\theta}(\phi)$ ($\mathcal{M}$ satisfies ϕ with probability greater than or equal to θ) versus the alternative hypothesis $H_1 : \mathcal{M} \models P_{<\theta}(\phi)$ ($\mathcal{M}$ satisfies ϕ with probability less than θ).

The algorithm for SMC by Bayesian hypothesis testing [29] (see Algorithm 1) is very simple but powerful. It is based exclusively on numerical simulations (called *traces*) of the system $\mathcal{M}$ to determine whether it accepts H_0 or H_1.

The SMC algorithm repeatedly simulates the system to draw sample traces from the system model $\mathcal{M}$ (line 4). For each sample trace σ, SMC checks whether or not σ satisfies the BLTL property ϕ (line 6) and increments the number of successes (x in line 7). At this point, the algorithm uses a statistical test, the Bayes factor test (line 9), to choose one of three things to do. The algorithm determines that it has seen enough data and accepts H_0 (line 10), or it determines that it has seen enough to accept the alternative hypothesis H_1 (line 11), or it decides that it has not yet seen statistically conclusive evidence and needs to repeat drawing sample traces (when $\frac{1}{T} \leq \mathcal{B} \leq T$). The value of the Bayes factor $\mathcal{B}$ gets larger when we see more statistical evidence in favor of H_0. It gets smaller when we see more statistical evidence in favor of H_1. If $\mathcal{B}$ exceeds a user-specified threshold T, the SMC algorithm accepts hypothesis H_0. If $\mathcal{B}$ becomes smaller than $\frac{1}{T}$, the SMC algorithm rejects H_0 and accepts H_1 instead. Otherwise $\mathcal{B}$ is inconclusive and the SMC algorithm repeats.

Algorithm 1 is very simple and generic. In order to be able to use it, we do, however, need to provide system models, a way of sampling from them efficiently, and a way to check properties for a given trace. In particular, for line 4, SMC needs a class of system models that is suitable for the application domain along with a way to sample traces from the system model. For correctness, it is imperative that the sample traces be drawn from the model in a fair (iid) way and that the resulting probabilities are well-defined. We investigate these questions in Sect. 2. For line 6, SMC needs a way to check a BLTL property along a given trace σ of the system. We investigate how this can be done and why it is well-defined for simulations, which can only have finite length, in Sect. 3. For line 9, we need an efficient way to compute the statistic employed, i.e., the Bayes factor—this is developed in Sect. 5.

Note that Statistical Model Checking cannot guarantee a correct answer of the PMC problem. The most crucial question needed to obtain meaningful results from SMC is whether the probability that the algorithm gives a wrong answer can be bounded. In Sect. 6 we prove that this error probability can indeed be bounded arbitrarily by the user. In Sect. 4, we also introduce a new form of SMC that uses Bayesian estimation instead of Bayesian hypothesis testing. Hypothesis-testing based methods are more efficient than those based on estimation when the threshold probability θ (which is specified by the user) is significantly different from the true probability that the property holds (which is determined by $\mathcal{M}$ and its initial state s_0) [48]. In this paper we show that our Bayesian estimation algorithm can be very efficient for probabilities close to 1 or close to 0.

Our SMC approach thus encompasses both hypothesis testing and estimation, and it is based on Bayes' theorem and sequential sampling. Bayes' theorem enables us to incorporate prior information about the model being verified. Sequential sampling means that the number of sampled traces is not fixed a priori. Instead, sequential algorithms determine the sample size at "run-time", depending on the evidence gathered by the samples seen so far. Because conclusive information from the samples can be used to stop our SMC algorithms as early as possible, this often leads to significantly smaller number of sampled traces (simulations).

We apply our approach to a representative example of discrete-time stochastic hybrid systems modeled in Stateflow/Simulink: a fault-tolerant fuel control system.

The contributions of this paper are as follows:

- We show how Statistical Model Checking can be used for Stateflow/Simulink-style hybrid systems with probabilistic transitions.
- We introduce Bayesian sequential interval estimation and prove almost sure termination.
- We prove analytic error bounds for the Bayesian sequential hypothesis testing and estimation algorithms.
- In a series of experiments with a relevant Stateflow/Simulink model, we empirically show that our sequential estimation method performs better than other estimation-based Statistical Model Checking approaches. In some cases our algorithm is faster by several orders of magnitudes.

While the theoretical analysis of our Statistical Model Checking approach is complicated by its sequential nature, a beneficial property of our algorithms is that they are easy to implement and often more efficient than working with a fixed number of samples.

2 Model

Our Statistical Model Checking algorithms can be applied to any stochastic model for which it is possible to define a probability space over its traces, draw sample traces in a fair way, and check BLTL properties for traces. Several stochastic models like discrete/continuous Markov chains satisfy this property [50]. Here we use discrete-time hybrid systems a la Stateflow/Simulink with probabilistic transitions.

Stateflow/Simulink[1] (SF/SL) is a model-based design and simulation tool developed by The Mathworks. It provides a graphical environment for designing data-flow programming architectures. It is heavily used in the automotive and defense industries for developing embedded systems. Models in SF/SL are recursively defined by *blocks*, which may contain specifications (in fact, blocks themselves) of communication, control, signal processing, and of many other types of computation. The blocks of a model are interconnected by *signals*. In Simulink, blocks operate with a continuous-time semantics, which means that the model's signals are thought to be continuous-time functions. (For example, the output of an integrator block is a continuous-time signal.) Stateflow introduces discrete-time computations in the form of discrete-time finite-state automata which can be freely embedded in Simulink blocks. The resulting models have thus a hybrid continuous/discrete-time semantics. The model simulation is accomplished by a solver (a numerical integration procedure) which calculates the temporal dynamics of the model. Finally, SF/SL provides for automated code generation (e.g., C) from the models. This feature is used when the model is ready to be deployed on the target hardware architecture.

Preliminaries We consider discrete-time stochastic processes over $\mathbb{R}^n$. Given a Borel set $S \subseteq \mathbb{R}^n$, we denote its Borel σ-algebra by $\mathcal{B}(S)$. We use the notion of stochastic kernel as a unifying concept for our particular model of stochastic hybrid system. Note that, unlike other models of stochastic hybrid systems [6, 8, 18, 35, 37], we do not consider continuous stochastic transitions, since those are less relevant for SF/SL applications.

Definition 1 A *stochastic kernel* on a measurable space $(S, \mathcal{B}(S))$ is a function $K : S \times \mathcal{B}(S) \to [0, 1]$ such that:

1. for each $x \in S$, $K(x, \cdot)$ is a probability measure on $\mathcal{B}(S)$; and
2. for each $B \in \mathcal{B}(S)$, $K(\cdot, B)$ is a (Borel) measurable function on S.

The sample space is the set $\Omega = S^\omega$ of (infinite) sequences of states, equipped with the usual product σ-algebra $\mathcal{F}$ of Ω (i.e., $\mathcal{F}$ is the smallest σ-algebra containing all the cylinder sets over $\mathcal{B}(S^n)$, for all $n \geqslant 1$.) Given a stochastic kernel K on $(\Omega, \mathcal{F})$ and an initial state $x \in S$, by Ionescu Tulcea's theorem [43, Theorem 2 in II.9] there exists a unique probability measure P defined on $(\Omega, \mathcal{F})$ and a Markov process $\{X_t : t \in \mathbb{N}\}$ such that for all $B \in \mathcal{B}(S)$ and for all $x_i \in S$:

- $\mathrm{P}(X_1 \in B) = \delta_x(B)$; and
- $\mathrm{P}(X_{t+1} \in B \mid (x_1, \ldots, x_t)) = \mathrm{P}(X_{t+1} \in B \mid x_t) = K(x_t, B)$

where δ_x is the Dirac measure (i.e., $\delta_x(B) = 1$ if $x \in B$, and 0 otherwise). Observe that we can formally assume all samples to be infinitely long by adding stuttering transitions for executions that have terminated.

Discrete-time hybrid systems As a system model, we consider discrete-time hybrid systems with additional probabilistic transitions (our case study uses SF/SL). Such a model gives rise to a transition system that allows for discrete transitions (e.g., from one Stateflow node to another), continuous transitions (when following differential equations underlying Simulink models), and probabilistic transitions (following a known probability distribution modeled in SF/SL). For SF/SL, a *state* assigns real values to all the state variables and

[1] http://www.mathworks.com/products/simulink/.

identifies the current location for Stateflow machines. We first define a deterministic hybrid automaton. Then we augment it with probabilistic transitions.

Definition 2 A *discrete-time hybrid automaton* (DTHA) consists of:

- a continuous state space $\mathbb{R}^n$;
- a directed graph with vertices Q (locations) and edges E (control switches);
- one initial state $(q_0, x_0) \in Q \times \mathbb{R}^n$;
- continuous flows $\varphi_q(t; x) \in \mathbb{R}^n$, representing the (continuous) state reached after staying in location q for time $t \geq 0$, starting from $x \in \mathbb{R}^n$;
- jump functions $jump_e : \mathbb{R}^n \to \mathbb{R}^n$ for edges $e \in E$.

Definition 3 The *transition function* for a *deterministic* DTHA is defined over $Q \times \mathbb{R}^n$ as

$$(q, x) \to_{\Delta(q,x)} (\tilde{q}, \tilde{x})$$

where

- for $t \in \mathbb{R}_{\geq 0}$, we have $(q, x) \to_t (q, \tilde{x})$ iff $\tilde{x} = \varphi_q(t; x)$;
- for $e \in E$, we have $(q, x) \to_e (\tilde{q}, \tilde{x})$ iff $\tilde{x} = jump_e(x)$ and e is an edge from q to $\tilde{q}$;
- $\Delta : Q \times \mathbb{R}^n \to \mathbb{R}_{\geq 0} \cup E$ is the *simulation* function.

The simulation function Δ makes system runs deterministic by selecting which particular discrete or continuous transition to execute from the current state. For Stateflow/Simulink, Δ satisfies several properties, including that the first edge that is enabled (i.e., where a jump is possible) will be chosen. Furthermore, if an edge is enabled, a discrete transition will be taken rather than a continuous transition. We note that Stateflow/Simulink's default ordering of outgoing edges is by clockwise orientation in the graphical notation, but allows user-specified precedence overrides as well.

Each execution of a DTHA is obtained by following the transition function repeatedly from state to state. A sequence $\sigma = (s_0, t_0), (s_1, t_1), \ldots$ of $s_i \in Q \times \mathbb{R}^n$ and $t_i \in \mathbb{R}_{\geq 0}$ is called *trace* iff, $s_0 = (q_0, x_0)$ and for each $i \in \mathbb{N}$, $s_i \to_{\Delta(s_i)} s_{i+1}$ and:

1. $t_i = \Delta(s_i)$ if $\Delta(s_i) \in \mathbb{R}_{\geq 0}$ (continuous transition), or
2. $t_i = 0$ if $\Delta(s_i) \in E$ (discrete transition).

Thus the system follows transitions from s_i to s_{i+1}. If this transition is a continuous transition, then t_i is its duration $\Delta(s_i)$, otherwise $t_i = 0$ for discrete transitions. In particular, the global time at state $s_i = (q_i, x_i)$ is $\sum_{0 \leq l < i} t_l$. We require that the sum $\sum_i^\infty t_i$ must diverge, that is, the system cannot make infinitely many state switches in finite time (*non-zero*).

Discrete time hybrid systems with probabilistic transitions A *probabilistic* DTHA is obtained from a DTHA by means of a *probabilistic* simulation function instead of (deterministic) simulation function Δ. Unlike Δ, it selects discrete and continuous transitions according to a probability density. We denote by $D(X)$ the set of probability density functions defined over set X.

Definition 4 The *probabilistic simulation* function Π for a DTHA is the map

$$\Pi = (\Pi_a, \Pi_c, \Pi_d) : Q \times \mathbb{R}^n \to D\big(\{0, 1\}\big) \times D(\mathbb{R}_{\geq 0}) \times D(E)$$

where for every $e = (a, b) \in E$ and $q \in Q$ it must be that $(\Pi_d(q))(e) = 0$ if $a \neq q$.

The condition on the edges ensures that only edges starting in the current location may have non-zero probability, because the others would not be enabled. For a given pair $(q,x) \in Q \times \mathbb{R}^n$, function Π enforces stochastic evolution by choosing the next transition—either a continuous or a discrete transition—according to a probability distribution (which in general depends on the current state (q,x)). In contrast, function Δ deterministically chooses the next transition by imposing $\Delta(q,x)$ as either a continuous or a discrete transition. Note that at every state $(q,x) \in Q \times \mathbb{R}^n$, Stateflow/Simulink's simulation engine deterministically chooses either an element of $D(\mathbb{R}_{\geq 0})$ or $D(E)$. Our probabilistic simulation function clearly generalizes this case—just choose either type of evolution with probability 1 (or 0), i.e., $\Pi_a(q,x)$ is a Dirac distribution at each (q,x).

We now define a stochastic kernel for a DTHA, using function Π.

Definition 5 Given a DTHA, let $S = Q \times \mathbb{R}^n$ be its state space. For a probabilistic simulation function Π we define the map $K : S \times \mathcal{B}(S) \to [0,1]$:

$$K\big((q,x),B\big) = p_a \cdot \Bigg(\sum_{\substack{\{e=(q,\tilde{q})\in E \mid \\ (q,jump_e(x))\in B\}}} \Pi_d(q,x)(e) \Bigg) + (1-p_a) \cdot \int_0^\infty \Pi_c(q,x)(t)\ I_B\big(q,\varphi_q(t,x)\big)dt$$

where I_B is the indicator function over set B and $p_a = \Pi_a(q,x)(0)$.

Note that the integrand, as for the case of discrete transition, contributes only if the flow stays in the given Borel set B. Function K thus probabilistically chooses between a discrete transition or a continuous evolution for any given $(q,x) \in S$. In particular, with probability $\Pi_a(q,x)(0)$ it will choose to perform a discrete transition, and a continuous evolution with probability $\Pi_a(q,x)(1) = 1 - \Pi_a(q,x)(0)$. After this choice, the actual action to be taken is sampled in the first case from a discrete distribution (over the edges $e \in E$) induced by Π_d, and in the second case from a continuous distribution (over time $t \geq 0$) induced by Π_c.

Lemma 1 *Function $K : S \times \mathcal{B}(S) \to [0,1]$ is a stochastic kernel.*

The proof of the lemma can be found in the Appendix. Again, by Ionescu Tulcea's theorem [43, Theorem 2 in II.9], this lemma shows that there is a unique probability measure and a Markov process defined by the stochastic kernel for probabilistic DTHA.

Note that initial distributions on the initial state can be obtained easily by prefixing the system with a probabilistic transition from the single initial state (q_0, x_0). Sample traces of a probabilistic DTHA can be obtained by sampling from the traces generated by the probabilistic simulation function Π.

Embedding Stateflow/Simulink SF/SL is a very sophisticated tool for embedded system design and simulation, and it is very challenging to define a comprehensive semantics for it. See the work of Tiwari [44] for a remarkable effort to define a formal semantics. Since Statistical Model Checking is based on numerical simulation, we do not need to know a full formal semantics of Stateflow/Simulink here. We can simply use its simulation engine (solver) to generate traces. What we assume, however, is that Stateflow/Simulink is well-behaved in terms of defining a probabilistic simulation function Π. Basically, Simulink primarily drives the continuous transitions (computed by numerical integration schemes) and interfaces with Stateflow blocks that determine the transition guards and discrete jumps. The continuous state space of the corresponding DTHA is some subset of $\mathbb{R}^n$ where n is the number of variables in the Stateflow/Simulink model. The locations Q and edges E correspond to the discrete transitions found in Stateflow/Simulink, also see [44]. For every state $(q,x) \in S$, Simulink/Stateflow, in fact, chooses deterministically whether a discrete transition $(\Pi_a(q,x)(0))$ or a continuous transition $(\Pi_a(q,x)(1))$ is performed next. That is, $\Pi_a(q,x)$ is deterministic. The discrete transition $\Pi_d(q,x)$ is also, for the most part, chosen deterministically based on the block placement or precedence numbers for enabled transitions. Randomness enters, however, in the outcome of random blocks introduced in the design by the user. The durations of continuous transitions are determined by the Simulink integration engine in a nontrivial way. What matters for us is not what exactly the distributions are that Stateflow/Simulink chooses for Π_a, Π_c, Π_d. Statistical model checking does not need whitebox access to the system model. It only needs a way to draw iid sample traces (cf. line 4 in Algorithm 1). What does matter, however, for Statistical Model Checking to work is the assumption that *there is* a well-defined probability measure over the trace space.

3 Specifying properties in temporal logic

Our algorithm verifies properties expressed as *Probabilistic Bounded Linear Temporal Logic* (PBLTL) formulas. We first define the syntax and semantics of (non-probabilistic) *Bounded Linear Temporal Logic* (BLTL), which we can check on a single trace, and then extend that logic to PBLTL. Finkbeiner and Sipma [16] have defined a variant of LTL on finite traces of discrete-event systems (where time is thus not considered).

For a stochastic model $\mathcal{M}$, let the set of state variables SV be a finite set of real-valued variables. A Boolean predicate over SV is a constraint of the form $y \sim v$, where $y \in SV$, $\sim \in \{\geq, \leq, =\}$, and $v \in \mathbb{R}$. A BLTL property is built on a finite set of Boolean predicates over SV using Boolean connectives and temporal operators. The syntax of the logic is given by the following grammar:

$$\phi ::= y \sim v \mid (\phi_1 \vee \phi_2) \mid (\phi_1 \wedge \phi_2) \mid \neg\phi_1 \mid \left(\phi_1 \mathbf{U}^t \phi_2\right),$$

where $\sim \in \{\geq, \leq, =\}$, $y \in SV$, $v \in \mathbb{Q}$, and $t \in \mathbb{Q}_{\geq 0}$. As usual, we can define additional temporal operators such as the operator "eventually within time t" which is defined as $\mathbf{F}^t \psi = \mathbf{True}\,\mathbf{U}^t \psi$, or the operator "always up to time t", which is defined as $\mathbf{G}^t \psi = \neg\mathbf{F}^t \neg\psi$.

We define the semantics of BLTL with respect to (infinite) executions of the model $\mathcal{M}$. The fact that an execution σ satisfies property ϕ is denoted by $\sigma \models \phi$. We denote the trace suffix starting at step i by σ^i (in particular, σ^0 denotes the original trace σ). We denote the value of the state variable y in σ at step i by $V(\sigma, i, y)$.

Definition 6 The *semantics* of BLTL for a trace σ^k starting at the kth state ($k \in \mathbb{N}$) is defined as follows:

- $\sigma^k \models y \sim v$ if and only if $V(\sigma, k, y) \sim v$;
- $\sigma^k \models \phi_1 \vee \phi_2$ if and only if $\sigma^k \models \phi_1$ or $\sigma^k \models \phi_2$;
- $\sigma^k \models \phi_1 \wedge \phi_2$ if and only if $\sigma^k \models \phi_1$ and $\sigma^k \models \phi_2$;
- $\sigma^k \models \neg\phi_1$ if and only if $\sigma^k \models \phi_1$ does not hold (written $\sigma^k \not\models \phi_1$);
- $\sigma^k \models \phi_1 \mathbf{U}^t \phi_2$ if and only if there exists $i \in \mathbb{N}$ such that

(a) $\sum_{0\leq l<i} t_{k+l} \leq t$,
(b) $\sigma^{k+i} \models \phi_2$, and
(c) $\sigma^{k+j} \models \phi_1$ for each $0 \leq j < i$.

Statistical Model Checking decides the probabilistic Model Checking problem by repeatedly checking whether $\sigma \models \phi$ holds on sample simulations σ of the system. In practice, sample simulations only have a finite duration. The question is how long these simulations have to be for the formula ϕ to have a well-defined semantics such that $\sigma \models \phi$ can be checked (line 6 of Algorithm 1). If σ is too short, say of duration 2, the semantics of $\phi_1 \mathbf{U}^{5.3} \phi_2$ may be unclear if ϕ_2 is false for duration 2. But at what duration of the simulation can we stop because we know that the truth-value for $\sigma \models \phi$ will never change by continuing the simulation? Is the number of required simulation steps expected to be finite at all?

For a class of finite length continuous-time boolean signals, well-definedness of checking bounded MITL properties has been conjectured in [34]. Here we generalize to infinite, hybrid traces with real-valued signals. We prove well-definedness and the fact that a finite prefix of the discrete time hybrid signal is sufficient for BLTL model checking, which is crucial for termination.

Lemma 2 (Bounded sampling) *The problem "$\sigma \models \phi$" is well-defined and can be checked for BLTL formulas ϕ and traces σ based on only a* finite prefix *of σ of bounded duration.*

For proving Lemma 2 we need to derive bounds on when to stop simulation. Those time bounds can be read off easily from the BLTL formula:

Definition 7 [34] The *sampling bound* $\#(\phi) \in \mathbb{Q}_{\geq 0}$ of a BLTL formula ϕ is the maximum nested sum of time bounds:

$$\begin{aligned} \#(y \sim v) &:= 0 \\ \#(\neg\phi_1) &:= \#(\phi_1) \\ \#(\phi_1 \vee \phi_2) &:= \max\bigl(\#(\phi_1), \#(\phi_2)\bigr) \\ \#(\phi_1 \wedge \phi_2) &:= \max\bigl(\#(\phi_1), \#(\phi_2)\bigr) \\ \#\bigl(\phi_1 \mathbf{U}^t \phi_2\bigr) &:= t + \max\bigl(\#(\phi_1), \#(\phi_2)\bigr) \end{aligned}$$

The next lemma (proof in the Appendix) shows that the semantics of BLTL formulas ϕ is well-defined on finite prefixes of traces with a duration that is bounded by $\#(\phi)$.

Lemma 3 (BLTL on bounded simulation traces) *Let ϕ be a BLTL formula, $k \in \mathbb{N}$. Then for any two infinite traces $\sigma = (s_0, t_0), (s_1, t_1), \ldots$ and $\tilde{\sigma} = (\tilde{s_0}, \tilde{t_0}), (\tilde{s_1}, \tilde{t_1}), \ldots$ with*

$$s_{k+I} = \tilde{s}_{k+I} \quad \textit{and} \quad t_{k+I} = \tilde{t}_{k+I} \quad \forall I \in \mathbb{N} \quad \textit{with} \sum_{0\leq l<I} t_{k+l} \leq \#(\phi) \tag{1}$$

we have that

$$\sigma^k \models \phi \quad \textit{iff} \quad \tilde{\sigma}^k \models \phi.$$

As a consequence, for checking $\sigma \models \phi$ during the Statistical Model Checking Algorithm 1, we can stop simulation of the sample σ when the duration exceeds $\#(\phi)$, because, according to Lemma 3, all possible extensions of the trace agree on whether ϕ holds or not. In the Appendix we prove that Lemma 2 holds using prefixes of traces according to the sampling bound $\#(\phi)$, which guarantees that finite simulations are sufficient for deciding whether ϕ holds on a trace. Note that $\#(\phi)$ is not the maximal number of transitions until formula ϕ can be decided along a trace, but only an upper bound for the amount of time that passes in the model $\mathcal{M}$ until ϕ can be decided. In particular, divergence of time on samples is required to ensure that the number of transitions is finite as well and that SMC terminates. Observe that ad-hoc ways of ensuring divergence of time by discarding samples with too slow a progress of time may bias the outcome of SMC.

We now define Probabilistic Bounded Linear Temporal Logic.

Definition 8 A *Probabilistic Bounded LTL* (PBLTL) formula is a formula of the form $P_{\geq\theta}(\phi)$, where ϕ is a BLTL formula and $\theta \in (0, 1)$ is a probability.

We say that $\mathcal{M}$ satisfies PBLTL property $P_{\geq\theta}(\phi)$, denoted by $\mathcal{M} \models P_{\geq\theta}(\phi)$, if and only if the probability that an execution trace of $\mathcal{M}$ satisfies BLTL property ϕ is greater than or equal to θ. This problem is well-defined, since by Lemma 2, each $\sigma \models \phi$ is decidable on a finite prefix of σ, and in the previous Section we have proved existence of a probability measure over the traces of $\mathcal{M}$. Thus, the set of all (non-zero) executions of $\mathcal{M}$ that satisfy a given BLTL formula is measurable [50]. Note that counterexamples to the BLTL property ϕ are *not* counterexamples to the PBLTL property $P_{\geq\theta}(\phi)$, because the truth of $P_{\geq\theta}(\phi)$ depends on the likelihood of all counterexamples to ϕ. This makes PMC more difficult than standard Model Checking, because one counterexample to ϕ is not enough to decide $P_{\geq\theta}(\phi)$. We refer to the excellent overview [9] for a discussion of counterexamples in Probabilistic Model Checking.

4 Bayesian interval estimation

We present our new Bayesian statistical estimation algorithm. In this approach we are interested in *estimating* p, the (unknown) probability that a random execution trace of $\mathcal{M}$ satisfies a fixed BLTL property ϕ. The estimate will be in the form of a confidence interval, i.e., an interval which will contain p with arbitrarily high probability.

For any trace σ_i of the system $\mathcal{M}$, we can, according to Lemma 2, deterministically decide whether σ_i satisfies BLTL formula ϕ. Therefore, we can define a Bernoulli random variable X_i denoting the outcome of $\sigma_i \models \phi$. The conditional probability density function associated with X_i is thus:

$$f(x_i|u) = u^{x_i}(1-u)^{1-x_i} \tag{2}$$

where $x_i = 1$ iff $\sigma_i \models \phi$, otherwise $x_i = 0$. Note that the X_i are (conditionally) independent and identically distributed (iid) random variables, as each trace is given by an independent execution of the model. Since p (the probability that property ϕ holds) is unknown, in the Bayesian approach one assumes that p is given by a random variable, whose density $g(\cdot)$ is called the *prior* density. The prior is usually based on our previous experiences and beliefs about the system. A lack of information about the probability of the system satisfying the formula is usually summarized by a *non-informative* or *objective* prior (see [39, Sect. 3.5] for an in-depth treatment).

Since p lies in $[0, 1]$, we need prior densities defined over this interval. In this paper we focus on Beta priors which are defined by the following probability density (for real parameters $\alpha, \beta > 0$ that give various shapes):

$$\forall u \in (0,1) \quad g(u, \alpha, \beta) \mathrel{\widehat{=}} \frac{1}{B(\alpha, \beta)} u^{\alpha-1}(1-u)^{\beta-1} \tag{3}$$

where the Beta function $B(\alpha, \beta)$ is defined as:

$$B(\alpha, \beta) \mathrel{\widehat{=}} \int_0^1 t^{\alpha-1}(1-t)^{\beta-1} dt. \tag{4}$$

By varying the parameters α and β, one can approximate several other smooth unimodal densities on $(0, 1)$ by a Beta density (e.g., the uniform density over $(0, 1)$ is a Beta with $\alpha = \beta = 1$). For all $u \in [0, 1]$ the Beta distribution function $F_{(\alpha,\beta)}(u)$ is defined:

$$F_{(\alpha,\beta)}(u) \mathrel{\widehat{=}} \int_0^u g(t, \alpha, \beta) dt = \frac{1}{B(\alpha, \beta)} \int_0^u t^{\alpha-1}(1-t)^{\beta-1} dt \tag{5}$$

which is the distribution function for a Beta random variable of parameters α, β (i.e., the probability that it takes values less than or equal to u).

An advantage of using Beta densities is that the Beta distribution is the *conjugate prior* to the Bernoulli distribution.[2] This relationship enables us to avoid numerical integration in the implementation of our Bayesian estimation and hypothesis testing algorithms, as we explain next. Furthermore, conjugate priors do not limit the type of priors usable in practice. It is known that any prior distribution (even those without a density) can be well approximated by a *finite* mixture of conjugate priors [15].

4.1 Bayesian intervals

Bayes' theorem states that if we sample from a density $f(\cdot|u)$, where the unknown probability u is given by a random variable U over $(0, 1)$ whose density is $g(\cdot)$, then the posterior density of U given the data $x_1, \ldots, x_n$ is:

$$f(u|x_1, \ldots, x_n) = \frac{f(x_1, \ldots, x_n|u)g(u)}{\int_0^1 f(x_1, \ldots, x_n|v)g(v)dv} \tag{6}$$

and in our case $f(x_1, \ldots, x_n|u)$ factorizes as $\prod_{i=1}^n f(x_i|u)$, where $f(x_i|u)$ is the conditional density function (2) associated with the i-th sample (remember that we assume conditionally independent, identically distributed—iid—samples). Since the posterior is an actual distribution (note the normalization constant), we can estimate p by the posterior *mean*. In fact, the posterior mean is a *posterior Bayes estimator* of p, i.e., it minimizes the risk over the whole parameter space of p (under a quadratic loss function, see [14, Chap. 8]).

For a *coverage* goal $c \in (\frac{1}{2}, 1)$, any interval (t_0, t_1) such that

$$\int_{t_0}^{t_1} f(u|x_1, \ldots, x_n) du = c \tag{7}$$

[2] A distribution $P(\theta)$ is said to be a conjugate prior for a likelihood function, $P(d|\theta)$, if the posterior, $P(\theta|d)$ is in the same family of distributions.

is called a $100c$ percent *Bayesian interval estimate* of p. Naturally, one would choose t_0 and t_1 that minimize $t_1 - t_0$ and satisfy (7), thus determining an optimal interval. (Note that t_0 and t_1 are in fact functions of the sample $x_1, \dots, x_n$.) Optimal interval estimates can be found, for example, for the mean of a normal distribution with normal prior, where the resulting posterior is normal. In general, however, it is difficult to find optimal interval estimates. For unimodal posterior densities like Beta densities, we can use the posterior's mean as the "center" of an interval estimate.

Here, we do not pursue the computation of an optimal interval, which may be numerically infeasible. Instead, we fix a desired half-interval width δ and then sample until the posterior probability of an interval of width 2δ containing the posterior mean exceeds c. When sampling from a Bernoulli distribution and with a Beta prior of parameters α, β, it is known that the mean $\hat{p}$ of the posterior is:

$$\hat{p} = \frac{x + \alpha}{n + \alpha + \beta} \tag{8}$$

where $x = \sum_{i=1}^{n} x_i$ is the number of successes in the sampled data $x_1, \dots, x_n$. The integral in (7) can be computed easily in terms of the Beta distribution function.

Proposition 1 *Let* (t_0, t_1) *be an interval in* $[0, 1]$. *The posterior probability of Bernoulli iid samples* $(x_1, \dots, x_n)$ *and Beta prior of parameters* $\alpha, \beta > 0$ *can be calculated as*:

$$\int_{t_0}^{t_1} f(u|x_1, \dots, x_n)du = F_{(x+\alpha, n-x+\beta)}(t_1) - F_{(x+\alpha, n-x+\beta)}(t_0) \tag{9}$$

where $x = \sum_{i=1}^{n} x_i$ *is the number of successes in* $(x_1, \dots, x_n)$ *and* $F(\cdot)$ *is the Beta distribution function.*

Proof Direct from definition of Beta distribution function (5) and the fact that the posterior density is a Beta of parameters $x + \alpha$ and $n - x + \beta$. □

The Beta distribution function can be computed with high accuracy by standard mathematical libraries (e.g. the GNU Scientific Library) or software (e.g. Matlab). Hence, as also argued in the previous section, the Beta distribution is the appropriate choice for summarizing the prior distribution in Statistical Model Checking.

4.2 Bayesian estimation algorithm

We want to compute an interval estimate of $p = \text{Prob}(\mathcal{M} \models \phi)$, where ϕ is a BLTL formula and $\mathcal{M}$ a stochastic hybrid system model—remember from our discussion in Sects. 2 and 3 that p is well-defined (but unknown). Fix the half-size $\delta \in (0, \frac{1}{2})$ of the desired interval estimate for p, the interval coverage coefficient $c \in (\frac{1}{2}, 1)$ to be used in (7), and the coefficients α, β of the Beta prior.

Our algorithm (shown in Algorithm 2) iteratively draws iid sample traces $\sigma_1, \sigma_2, \dots,$ and checks whether they satisfy ϕ. At stage n, the algorithm computes the posterior mean $\hat{p}$, which is the Bayes estimator for p, according to (8). Next, using $t_0 = \hat{p} - \delta$, $t_1 = \hat{p} + \delta$ it computes the posterior probability of the interval (t_0, t_1) as

$$\gamma = \int_{t_0}^{t_1} f(u|x_1, \dots, x_n)du.$$

```
Input  : BLTL Property $\phi$, half-interval size $\delta \in (0, \frac{1}{2})$, interval coverage coefficient
         $c \in (\frac{1}{2}, 1)$, Prior Beta distribution with parameters $\alpha, \beta$ for the (unknown)
         probability $p$ that the system satisfies $\phi$
Output: An interval $(t_0, t_1)$ of width $2\delta$ with posterior probability at least $c$, estimate
         $\hat{p}$ for the true probability $p$
1  $n := 0$;                                    {number of traces drawn so far}
2  $x := 0$;                              {number of traces satisfying $\phi$ so far}
3  repeat
4      $\sigma :=$ draw a sample trace of the system (iid);
5      $n := n + 1$;
6      if $\sigma \models \phi$ then $x := x + 1$;
7      $\hat{p} := (x + \alpha)/(n + \alpha + \beta)$;            {compute posterior mean}
8      $(t_0, t_1) := (\hat{p} - \delta, \hat{p} + \delta)$;           {compute interval estimate}
9      if $t_1 > 1$ then $(t_0, t_1) := (1 - 2 \cdot \delta, 1)$
10     else if $t_0 < 0$ then $(t_0, t_1) := (0, 2 \cdot \delta)$;
       {compute posterior probability of $p \in (t_0, t_1)$, by (9)}
11     $\gamma :=$ PosteriorProb$(t_0, t_1)$
12 until $(\gamma \geqslant c)$;
13 return $(t_0, t_1)$, $\hat{p}$
```

Algorithm 2: Statistical Model Checking by Bayesian Interval Estimates

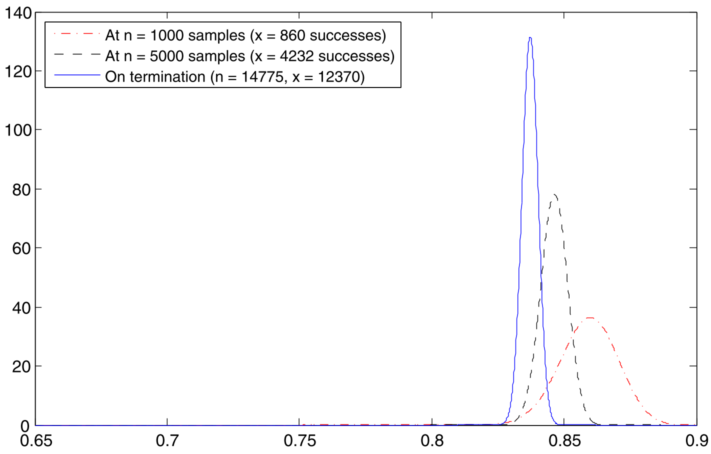

Fig. 1 Posterior density for sequential Bayesian estimation (Algorithm 2) at several sample sizes; the unknown probability is 0.84, $\delta = 0.01$, $c = 0.999$, uniform prior ($\alpha = \beta = 1$). For readability the graph is restricted to [0.65, 0.9]

If $\gamma \geqslant c$ the algorithm stops and returns t_0, t_1 and $\hat{p}$; otherwise it samples another trace and repeats. One should pay attention at the extreme points of the (0, 1) interval, but those are easily taken care of, as shown in lines 9 and 10 of Algorithm 2. Note that the algorithm always returns an interval of width 2δ.

In Fig. 1 we give three snapshots of a typical execution of the algorithm. We have plotted the posterior density (6) after 1000 and 5000 samples, and on termination of the algorithm (which occurred after 14775 samples for this specific run.) In this experiment, we have replaced system simulation and trace checking by tossing a coin with a fixed bias. Therefore, the Bayesian estimation algorithm will compute both interval and point estimates for such (unknown) bias. The true coin bias (i.e., the probability of success) used was 0.84, while half-interval size was $\delta = 0.01$, coverage probability was $c = 0.999$, and the prior was uniform ($\alpha = \beta = 1$). From Fig. 1 we can see that after 1000 samples the posterior density is almost entirely within the $[0.825, 0.9]$ interval. However, as the algorithm progresses to 5000 samples, the posterior density gets "leaner" and "taller", i.e., its mass is concentrated over a shorter interval, thereby giving better (tighter) estimates. On termination, after 14775 samples for this run, the posterior density becomes even more thin, and satisfies the required coverage probability (c) and half-interval size (δ) specified by the user.

Finally, we note that the posterior mean $\hat{p}$ is a *biased* estimator of the true probability p—the expected value of $\hat{p}$ is not p, as it can be easily seen from $\hat{p}$'s definition (8). However, this is not a problem, since unbiasedness is a rather weak property of estimators for the mean. (Unbiased estimators are easy to get: from a sample of iid variables estimate their mean by using any one sample. This estimator is unbiased by definition, but of course not useful.) More importantly, the posterior mean $\hat{p}$ is a *consistent* estimator, i.e., it converges in probability to p as the sample size tends to infinity (this is immediate from (8) and the Law of Large Numbers.) Therefore, larger sample sizes will generally yields more accurate estimates, as it is witnessed by Fig. 1, and as one would expect from a "good" estimator. We remark that the consistency of $\hat{p}$ is independent from the choice of the prior.

5 Bayesian hypothesis testing

In this section we briefly present the sequential Bayesian hypothesis test, which was introduced in [29]. Recall that the PMC problem is to decide whether $\mathcal{M} \models P_{\geq\theta}(\phi)$, where $\theta \in (0, 1)$ and ϕ is a BLTL formula. Let p be the (unknown but fixed) probability of the model satisfying ϕ: thus, the PMC problem can now be stated as deciding between two hypotheses:

$$H_0 : p \geqslant \theta \qquad H_1 : p < \theta. \tag{10}$$

Let $X_1, \ldots, X_n$ be a sequence of Bernoulli random variables defined as for the PMC problem in Sect. 4, and let $d = (x_1, \ldots, x_n)$ denote a sample of those variables. Let H_0 and H_1 be mutually exclusive hypotheses over the random variable's parameter space according to (10). Suppose the *prior probabilities* $P(H_0)$ and $P(H_1)$ are strictly positive and satisfy $P(H_0) + P(H_1) = 1$. Bayes' theorem states that the *posterior probabilities* are:

$$P(H_i|d) = \frac{P(d|H_i)P(H_i)}{P(d)} \quad (i = 0, 1) \tag{11}$$

for every d with $P(d) = P(d|H_0)P(H_0) + P(d|H_1)P(H_1) > 0$. In our case $P(d)$ is always non-zero (there are no impossible *finite* sequences of outcomes).

5.1 Bayes factor

By Bayes' theorem, the posterior odds for hypothesis H_0 is

$$\frac{P(H_0|d)}{P(H_1|d)} = \frac{P(d|H_0)}{P(d|H_1)} \cdot \frac{P(H_0)}{P(H_1)}. \tag{12}$$

Definition 9 The Bayes factor $\mathcal{B}$ of sample d and hypotheses H_0 and H_1 is

$$\mathcal{B} = \frac{P(d|H_0)}{P(d|H_1)}.$$

For fixed priors in a given example, the Bayes factor is directly proportional to the posterior odds by (12). Thus, it may be used as a measure of relative confidence in H_0 vs. H_1, as proposed by Jeffreys [28]. To test H_0 vs. H_1, we compute the Bayes factor $\mathcal{B}$ of the available data d and then compare it against a fixed threshold $T \geqslant 1$: we shall accept H_0 iff $\mathcal{B} > T$. Jeffreys interprets the value of the Bayes factor as a measure of the evidence in favor of H_0 (dually, $\frac{1}{\mathcal{B}}$ is the evidence in favor of H_1). Classically, a fixed number of samples was suggested for deciding H_0 vs. H_1. We develop an algorithm that chooses the number of samples adaptively.

We now show how to compute the Bayes factor. According to Definition 9, we have to calculate the ratio of the probabilities of the observed sample $d = (x_1, \dots, x_n)$ given H_0 and H_1. By (12), this ratio is proportional to the ratio of the posterior probabilities, which can be computed from Bayes' theorem (6) by integrating the joint density $f(x_1|\cdot) \cdots f(x_n|\cdot)$ with respect to the prior $g(\cdot)$:

$$\frac{P(H_0|x_1, \dots, x_n)}{P(H_1|x_1, \dots, x_n)} = \frac{\int_\theta^1 f(x_1|u) \cdots f(x_n|u) \cdot g(u)du}{\int_0^\theta f(x_1|u) \cdots f(x_n|u) \cdot g(u)du}.$$

Thus, the Bayes factor is:

$$\mathcal{B} = \frac{\pi_1}{\pi_0} \cdot \frac{\int_\theta^1 f(x_1|u) \cdots f(x_n|u) \cdot g(u)du}{\int_0^\theta f(x_1|u) \cdots f(x_n|u) \cdot g(u)du} \tag{13}$$

where $\pi_0 = P(H_0) = \int_\theta^1 g(u)du$, and $\pi_1 = P(H_1) = 1 - \pi_0$. We observe that the Bayes factor depends on the data d and on the prior g, so it may be considered a measure of confidence in H_0 vs. H_1 provided by the data $x_1, \dots, x_n$, and "weighted" by the prior g. When using Beta priors, the calculation of the Bayes factor can be much simplified.

Proposition 2 *The Bayes factor of $H_0 : p \geqslant \theta$ vs. $H_1 : p < \theta$ with Bernoulli samples $(x_1, \dots, x_n)$ and Beta prior of parameters α, β is:*

$$\mathcal{B}_n = \frac{\pi_1}{\pi_0} \cdot \left(\frac{1}{F_{(x+\alpha, n-x+\beta)}(\theta)} - 1 \right).$$

where $x = \sum_{i=1}^n x_i$ is the number of successes in $(x_1, \dots, x_n)$ and $F_{(s,t)}(\cdot)$ is the Beta distribution function of parameters s, t.

5.2 Bayesian hypothesis testing algorithm

Our Bayesian hypothesis testing algorithm generalizes Jeffreys' test to a sequential version, and it is shown in Algorithm 1. Remember we want to establish whether $\mathcal{M} \models P_{\geqslant\theta}(\phi)$, where $\theta \in (0, 1)$ and ϕ is a BLTL formula. The algorithm iteratively draws independent and identically distributed sample traces $\sigma_1, \sigma_2, \dots$ (line 4), and checks whether they satisfy ϕ (line 6). Again, we can model this procedure as independent sampling from a Bernoulli distribution X of unknown parameter p—the actual probability of the model satisfying ϕ. At stage n the algorithm has drawn samples $x_1, \ldots, x_n$ iid like X. In line 9, it then computes the Bayes factor $\mathcal{B}$ according to Proposition 2, to check if it has obtained conclusive evidence. The algorithm accepts H_0 iff $\mathcal{B} > T$ (line 10), and accepts H_1 iff $\mathcal{B} < \frac{1}{T}$ (line 11). Otherwise ($\frac{1}{T} \leqslant \mathcal{B} \leqslant T$) it repeats the loop and continues drawing iid samples.

6 Analysis

Statistical Model Checking algorithms are easy to implement and—because they are based on selective system simulation—enjoy promising scalability properties. Yet, for the same reason, their output would be useless, unless the probability of making an error during the PMC decision can be bounded.

As our main contribution, we prove (almost sure) termination for the Bayesian interval estimation algorithm, and we prove error bounds for Statistical Model Checking by Bayesian sequential hypothesis testing and by Bayesian interval estimation. The proofs of these results can be found in the Appendix. Termination of the Bayesian sequential hypothesis testing has been shown in [52]. In the Appendix we now prove termination of the sequential Bayesian estimation algorithm.

Theorem 1 (Termination of Bayesian estimation) *The Sequential Bayesian interval estimation Algorithm 2 terminates with probability one.*

Next, we show that the (Bayesian) Type I-II error probabilities for the algorithms in Sects. 4–5 can be bounded arbitrarily. We recall that a Type I (II) error occurs when we reject (accept) the null hypothesis although it is true (false).

Theorem 2 (Error bound for hypothesis testing) *For any discrete random variable and prior, the probability of a Type I-II error for the Bayesian hypothesis testing Algorithm 1 is bounded above by $\frac{1}{T}$, where T is the Bayes Factor threshold given as input.*

Note that the bound $\frac{1}{T}$ is independent from the prior used. Also, in practice a slightly smaller Type I-II error can be read off from the actual Bayes factor at the end of the algorithm. By construction of the algorithm, we know that this Bayes factor is bounded above by $\frac{1}{T}$, but it may be much smaller than that when the algorithm terminates.

Finally, we lift the error bounds found in Theorem 2 for Algorithm 1 to Algorithm 2 by representing the output of the Bayesian interval estimation Algorithm 2 as a hypothesis testing problem. We use the output interval (t_0, t_1) of Algorithm 2 to define the (null) hypothesis $H_0 : p \in (t_0, t_1)$. Now H_0 represents the hypothesis that the output of Algorithm 2 is correct. Then, we can test H_0 and determine bounds on Type I and II errors by Theorem 2.

Theorem 3 (Error bound for estimation) *For any discrete random variable and prior, the Type I and II errors for the output interval (t_0, t_1) of the Bayesian estimation Algorithm 2 are bounded above by $\frac{(1-c)\pi_0}{c(1-\pi_0)}$, where c is the coverage coefficient given as input and π_0 is the prior probability of the hypothesis $H_0 : p \in (t_0, t_1)$.*

7 Application

We study an example that is part of the Stateflow/Simulink package. The model[3] describes a fuel control system for a gasoline engine. It detects sensor failures, and dynamically changes

[3] *Modeling a Fault-Tolerant Fuel Control System.* http://www.mathworks.com/help/simulink/examples/modeling-a-fault-tolerant-fuel-control-system.html.

the control law to provide seamless operation. A key quantity in the model is the ratio between the air mass flow rate (from the intake manifold) and the fuel mass flow rate (as pumped by the injectors). The system aims at keeping the air-fuel ratio close to the *stoichiometric* ratio of 14.6, which represents an acceptable compromise between performance and fuel consumption. The system estimates the "correct" fuel rate giving the target stoichiometric ratio by taking into account sensor readings for the amount of oxygen present in the exhaust gas (EGO), for the engine speed, throttle command and manifold absolute pressure. In the event of a single sensor fault, the system detects the situation and operates the engine with a higher fuel rate to compensate. If two or more sensors fail, the engine is shut down, since the system cannot reliably control the air-fuel ratio.

The Stateflow control logic of the system has a total of 24 locations, grouped in 6 parallel (i.e., simultaneously active) states. The Simulink part of the system is described by several nonlinear equations and a linear differential equation with a switching condition. Overall, this model provides a representative summary of the important features of hybrid systems. Our stochastic system is obtained by introducing random faults in the EGO, speed and manifold pressure sensors. We model the faults by three independent Poisson processes with different arrival rates. When a fault happens, it is "repaired" with a fixed service time of one second (i.e., the sensor remains in fault condition for one second, then it resumes normal operation). Note that the system has no free inputs, since the throttle command provides a periodic triangular input, and the nominal speed is never changed. This ensures that, once we set the three fault rates, for any given temporal logic property ϕ the probability that the model satisfies ϕ is well-defined. All our experiments have been performed on a 2.4 GHz Pentium 4, 1 GB RAM desktop computer running Matlab R2008b on Windows XP.

7.1 Experimental results in application

For our experiments we model check the following formula (null hypothesis)

$$H_0 : \mathcal{M} \models P_{\geq\theta}\big(\neg \mathbf{F}^{100}\mathbf{G}^{1}(\mathit{FuelFlowRate} = 0)\big) \tag{14}$$

for different values of threshold θ and sensors fault rates. We test whether with probability greater than θ it is not the case that within 100 seconds the fuel flow rate stays zero for one second. The fault rates are expressed in seconds and represent the mean interarrival time between two faults (in a given sensor). In experiment 1, we use uniform priors over $(0, 1)$, with null and alternate hypotheses equally likely a priori. In experiment 2, we use *informative* priors, i.e., Beta priors highly concentrated around the true probability that the model satisfies the BLTL formula (this was obtained simply by using our Bayesian estimation algorithm—see below.) The Bayes Factor threshold is $T = 1000$, so by Theorem 2 both Type I and II errors are bounded by 0.001.

Table 1 Number of samples/verification time when testing (14) with uniform, equally likely priors and $T = 1000$: ✗ = 'H_0 rejected', ✓ = 'H_0 accepted'

		Probability threshold θ	
		0.9	0.99
Fault rates	(3 7 8)	✗ (8/21 s)	✗ (2/5 s)
	(10 8 9)	✗ (710/1738 s)	✗ (8/21 s)
	(20 10 20)	✓ (44/100 s)	✗ (1626/3995 s)
	(30 30 30)	✓ (44/107 s)	✓ (239/589 s)

Table 2 Number of samples/verification time when testing (14) with informative priors and $T = 1000$: ✗ = 'H_0 rejected', ✓ = 'H_0 accepted'

		Probability threshold θ	
		0.9	0.99
Fault rates	(3 7 8)	✗ (8/21 s)	✗ (2/5 s)
	(10 8 9)	✗ (255/632 s)	✗ (8/21 s)
	(20 10 20)	✓ (39/88 s)	✗ (1463/3613 s)
	(30 30 30)	✓ (33/80 s)	✓ (201/502 s)

Table 3 Posterior mean/number of samples for estimating probability of (15) with uniform prior and $\delta = 0.05$, and sample size required by the Chernoff-Hoeffding bound [27]

		Interval coverage c	
		0.99	0.999
Fault rates	(3 7 8)	0.3569/606	0.3429/972
	(10 8 9)	0.8785/286	0.8429/590
	(20 10 20)	0.9561/112	0.9625/158
	(30 30 30)	0.9778/43	0.9851/65
	C-H bound	922	1382

Table 4 Posterior mean/number of samples when estimating probability of (15) with uniform prior and $\delta = 0.01$, and sample size required by the Chernoff-Hoeffding bound [27]

		Interval coverage c	
		0.99	0.999
Fault rates	(3 7 8)	0.3558/15205	0.3563/24830
	(10 8 9)	0.8528/8331	0.8534/13569
	(20 10 20)	0.9840/1121	0.9779/2583
	(30 30 30)	0.9956/227	0.9971/341
	C-H bound	23026	34539

In Tables 1 and 2 we report our results. Even in the longest run (for $\theta = .99$ and fault rates (20 10 20) in Table 1), Bayesian SMC terminates after 3995 s already. This is very good performance for a test with such a small (0.001) error probability run on a desktop computer. We note that the total time spent for this case on actually computing the statistical test, i.e., Bayes factor computation, was just about 1 s. The dominant computation cost is system simulation. Also, by comparing the sample sizes of Table 1 and 2 we note that the use of an informative prior generally helps the algorithm—i.e., fewer samples are required to decide.

Next, we estimate the probability that $\mathcal{M}$ satisfies the following property, using our Bayesian estimation algorithm:

$$\mathcal{M} \models \big(\neg \mathbf{F}^{100}\mathbf{G}^{1}(\mathit{FuelFlowRate} = 0)\big). \tag{15}$$

In particular, we ran two sets of tests, one with half-interval size $\delta = 0.05$ and another with $\delta = 0.01$. In each set we used different values for the interval coefficient c and different sensor fault rates, as before. Experimental results are in Tables 3 and 4. We used uniform priors in both cases.

Finally, since both Bayesian estimation and hypothesis testing are general techniques, they can be applied to a variety of models. In fact, we have recently applied both approaches to verify computational models of biological signaling pathways [20] and analog circuits [47]. In the systems biology application, models are inherently probabilistic because of the stochastic nature of the (quantum) physics underlying chemical reactions. In particular, the pioneering work of Gillespie [19] showed that, under some reasonable assumptions, one can use continuous-time Markov chains to approximate the temporal evolution of chemical reaction networks. In our work [20] we used the BioNetGen rule-based language [26] to succinctly describe the reactions of an important signaling pathway in cancer. The model (i.e., the resulting continuous-time Markov chain) is then efficiently simulated by the BioNetGen simulator and verified against known behavioral properties expressed in BLTL. The size of the model—to the order of 10^{40} states—is currently out of the reach of (standard) probabilistic model checking techniques.

In analog circuits, low-voltage operations and process variability in the fabrication process can affect the nominal performance of a circuit, effectively turning a deterministic system into a stochastic one. In [47] we studied a gate-level model of an operational amplifier. We used the tool SPICE as the model specification language and simulator. (SPICE is a popular tool used by analog designers and validation engineers.) Stochastic behavior was introduced to model process variation. In particular, we assumed that four parameters of each transistor in the model were subject to normally-distributed noise. Transient properties of the circuit model were described using BLTL, which proved to be a useful specification language in this domain, too. Again, the combination of efficient statistical techniques and simulation has made verification accessible for analog circuit designs.

7.2 Discussion

A general trend shown by our experimental results and additional simulations is that our Bayesian estimation model checking algorithm is generally faster at the extremes, i.e., when the unknown probability p is close to 0 or close to 1. Performance is worse when p is closer to 0.5. In contrast, the performance of our Bayesian hypothesis testing model checking algorithm is faster when the unknown true probability p is far from the threshold probability θ.

We note the remarkable performance of our estimation approach compared to the technique based on the Chernoff-Hoeffding bound [25]. From Tables 3, 4, and 5 we see that when the unknown probability is close to 1, our algorithm can be up to two orders of magnitude faster. (The same argument holds when the true probability is close to 0.) Chernoff-Hoeffding bounds hold for any random variable with bounded variance. Our Bayesian approach, instead, explicitly constructs the posterior distribution on the basis of the Bernoulli sampling distribution and the prior.

We chose the Chernoff-Hoeffding technique as a comparison because it provides *bounded* confidence intervals, i.e., the interval's coverage probability is bounded below by a desired value. Similarly, our Bayesian estimation technique provides bounded confidence intervals (for given a prior distribution.) There are other, standard confidence interval techniques (see for example [40, Chap. 4]) based on the Central Limit Theorem (CLT). However, they are based on finite-sample *approximations* of the CLT. As a result, the coverage probability of the computed interval is not guaranteed to be equal to or larger than the desired value. One only has an *approximate* coverage probability, depending on the chosen sample size. The coverage probability will of course converge to desired value as the sample size tends to infinity.

Finally, we point out that the choice of a prior is obviously subjective. The user may employ previous knowledge, empirical observations, and other information sources to decide

Table 5 Number of samples for Chernoff-Hoeffding bound and Bayesian estimation via Monte Carlo simulation. The number of samples for Bayesian estimation is the average of 10 repetitions for the $\delta = 0.001$ case, and of 100 repetitions for $\delta = 0.01$ (numbers rounded to the nearest integer)

	Bayesian estimation			Chernoff-Hoeffding bound
	$p = 0.9999$	$p = 0.999$	$p = 0.5$	
$\delta = 0.01$ $c = 0.99$	230	258	16582	23026
$\delta = 0.001$ $c = 0.99999$	6662	23385	4877844	5756463

Table 6 Number of samples at the extreme points of the Monte Carlo simulations for Bayesian estimation reported in Fig. 2. The number of samples is the average of 100 repetitions, rounded to the nearest integer

	Bayesian estimation					
	$p = 0.0001$	$p = 0.001$	$p = 0.01$	$p = 0.99$	$p = 0.999$	$p = 0.9999$
$\delta = 0.05$ $c = 0.99999$	109	113	144	140	113	109
$\delta = 0.01$ $c = 0.99$	228	240	738	660	258	230

on a particular prior distribution. However, in statistical model checking the cost of sampling is relatively low, so the choice of an adequate prior is not much of a problem. For all practical purposes, we have found that a "simple" prior such as the uniform distribution is fine. If the user has an informative prior (i.e., highly concentrated around a specific value), then this can certainly be used, and it will in general lead to a reduction in the number of samples needed. Also, even with reasonably "wrong" priors, the evidence coming from the samples will eventually overcome the prior. A prior with zero probability on some regions would, however, rule out such regions from any further Bayesian inference. In fact, the posterior probability of those regions would always be zero.

7.3 Performance evaluation

We have conducted a series of Monte Carlo simulations to analyze the performance (measured as number of samples) of our sequential Bayesian estimation algorithm with respect to the unknown probability p. In particular, we have run simulations for values of p ranging from 0.0001 to 0.9999, with coverage (c) of 0.99 and 0.99999, interval half-size (δ) of 0.05, 0.01 and 0.001, and uniform prior.

Our experiments (see Fig. 2) show that Bayesian estimation is very fast when p is close to either 0 or 1, while a larger number of samples is needed when p is close to $\frac{1}{2}$. In a sense, our algorithm can decide easier PMC instances faster: if the probability p of a formula being true is very small or very large, we need fewer samples. This is another advantage of our approach that it is not currently matched by other SMC estimation techniques (e.g., [25]). Our findings are consistent with those of Yu et al. [51] in the VLSI testing domain.

Our simulations also indicate that the performance of the algorithm depends more strongly on the half-size δ of the interval than on the coverage c of the interval. It is much faster to estimate an interval of half-size $\delta = 0.05$ with coverage $c = 0.99999$ than it is to estimate an interval of $\delta = 0.01$ with $c = 0.99$, as we can see from Fig. 2 and Table 6. More theoretical work is needed, however, to characterize fully the behavior of the Bayesian sequential estimation algorithm. Our initial findings suggest that the algorithm scales very well.

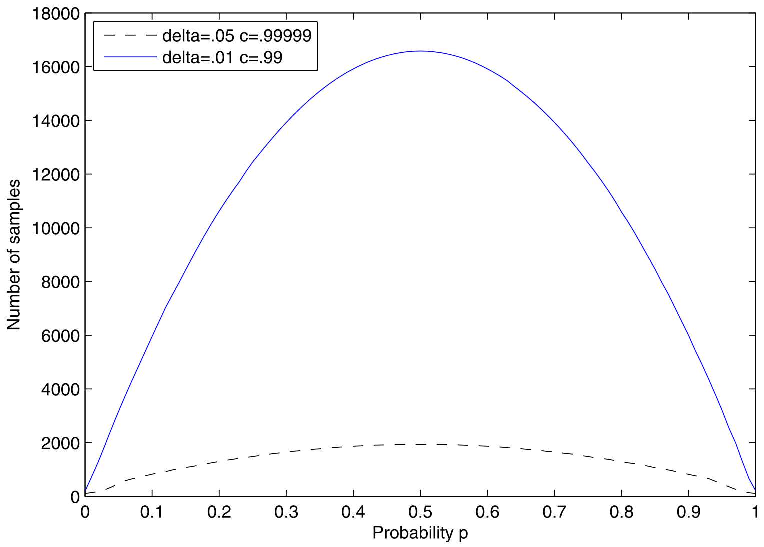

Fig. 2 Monte Carlo simulation for analyzing the performance of Bayesian estimation in two different settings. The number of samples is the average of 100 repetitions. The probability p runs from 0.01 to 0.99 with 0.01 increment, plus the points 0.0001, 0.001, 0.999, and 0.9999. Uniform prior

A study of the relationship between Bayesian intervals and (classical or frequentist) confidence intervals is beyond the scope of this paper, so we will just provide a brief discussion. We start by recalling the difference between Bayesian intervals and confidence intervals. The coverage probability of a Bayesian interval is an actual probability statement about the unknown parameter belonging to that interval, given the sampled data and the prior. Instead, the coverage probability carried by a confidence interval procedure tells us that, e.g., 99% of the time the procedure will output an interval containing the unknown parameter. Now, recall the *asymptotic normality* of the posterior distribution, i.e., when the sample size tends to infinity the posterior distribution converges to a normal distribution. (The result holds also when the actual distribution of the data does not belong to the same family of the posterior—see [17, Sect. 4.2] for more details.) This means that for large sample sizes our posterior distribution converges to a normal distribution of mean p, the unknown probability to estimate, and variance proportional to $\frac{1}{n}$, where n is the sample size. It follows that, *asymptotically*, the coverage probability of Bayesian intervals centered on the posterior mean will be equal to that of confidence intervals computed via techniques based on the normal distribution.

For finite (and small) sample sizes, Bayesian techniques using informative priors can have different properties than frequentist techniques. The reason is because one can construct arbitrarily strong priors that are quite different from the actual distribution of the data. However, it is known that by using weak (vague or non-informative) priors, Bayesian estimates typically enjoy good frequentist properties such as coverage probability, even for small sample sizes. In particular, this holds for the normal and Bernoulli distributions, for which Bayesian estimation techniques can actually produce shorter intervals than frequentist techniques [7, Sects. 5.6 and 5.7]. We remind that in our experiments we have always used uniform priors, except for the experiments of Table 2, where we have used informative priors weakly centered around the unknown probability (estimated through our Bayesian technique).

Finally, we note that the sequential estimation technique presented in [10] is *asymptotically* "consistent" (the technique is based on an approximation of the normal distribution). That is, the coverage probability of the returned interval is guaranteed to be the desired value only in the limit $\delta \to 0$ (interval width approaching 0). Therefore, given a finite $\delta > 0$ the actual coverage probability may be less than the desired value.

8 Related work

Younes, Musliner and Simmons introduced the first algorithm for Statistical Model Checking [49, 50]. Their work uses the SPRT [46], which is designed for *simple* hypothesis testing.[4] Specifically, the SPRT decides between the simple null hypothesis $H_0' : \mathcal{M} \models P_{=\theta_0}(\phi)$ against the simple alternate hypothesis $H_1' : \mathcal{M} \models P_{=\theta_1}(\phi)$, where $\theta_0 < \theta_1$. The SPRT is optimal for simple hypothesis testing, since it minimizes the expected number of samples among all the tests satisfying the same Type I and II errors, when either H_0' or H_1' is true [46]. The PMC problem is instead a choice between two *composite* hypotheses $H_0 : \mathcal{M} \models P_{\geq\theta}(\phi)$ versus $H_1 : \mathcal{M} \models P_{<\theta}(\phi)$. The SPRT is not defined unless $\theta_0 \neq \theta_1$, so Younes and Simmons overcome this problem by separating the two hypotheses by an *indifference region* $(\theta - \delta, \theta + \delta)$, inside which any answer is tolerated. Here $0 < \delta < 1$ is a user-specified parameter. It can be shown that the SPRT with indifference region can be used for testing composite hypotheses, while respecting the same Type I and II errors of a standard SPRT [46]. However, in this case the test is no longer optimal, and the maximum expected sample size may be much bigger than the optimal fixed-size sample test [5]. The Bayesian approach solves instead the composite hypothesis testing problem, with no indifference region.

The method of [25] uses a fixed number of samples and estimates the probability that the property holds as the number of satisfying traces divided by the number of sampled traces. Their algorithm guarantees the accuracy of the results using the Chernoff-Hoeffding bound. In particular, their algorithm can guarantee that the difference in the estimated and the true probability is less than ϵ, with probability ρ, where $\rho < 1$ and $\epsilon > 0$ are user-specified parameters. Our experimental results show a significant advantage of our Bayesian estimation algorithm in the sample size.

Grosu and Smolka use a standard acceptance sampling technique for verifying formulas in LTL [21]. Their algorithm randomly samples lassos (i.e., random walks ending in a cycle) from a Büchi automaton in an on-the-fly fashion. The algorithm terminates if it finds a counterexample. Otherwise, the algorithm guarantees that the probability of finding a counterexample is less than δ, under the assumption that the true probability that the LTL formula is true is greater than ϵ (δ and ϵ are user-specified parameters).

Sen et al. [41] used the *p-value* for the null hypothesis as a statistic for hypothesis testing. The p-value is defined as the probability of obtaining observations at least as extreme as the one that was actually seen, given that the null hypothesis is true. It is important to realize that a p-value is *not* the probability that the null hypothesis is true. Sen et al.'s method does not have a way to control the Type I and II errors. Sen et al. [42] have started investigating the extension of SMC to unbounded (i.e., standard) LTL properties. Langmead [33] has applied Bayesian point estimation and SMC for querying Dynamic Bayesian Networks.

[4] A simple hypothesis completely specifies a distribution. For example, a Bernoulli distribution of parameter p is fully specified by the hypothesis $p = 0.3$ (or some other numerical value). A composite hypothesis, instead, still leaves the free parameter p in the distribution. This results, e.g., in a family of Bernoulli distributions with parameter $p < 0.3$.

It should be noted that with the use of abstraction, efficient tools such as Prism [31] can verify large instances of probabilistic models using numerical (non-statistical) techniques. Also, Prism can perform statistical model checking using the SPRT and the Chernoff-Hoeffding bound as in [25]. Abstraction has been proposed for hybrid systems [45], as well. However, automated abstraction is still difficult to perform in general, and in particular for Stateflow/Simulink models.

With respect to the temporal logic used in this work, we note that our BLTL is a sublogic of Metric Temporal Logic (MTL) [30]. In particular, MTL extends LTL by endowing the Until operator with an *interval* of the positive real line with natural endpoints (or infinite). Such intervals can also be singleton, i.e., MTL can specify that an event happens at a particular time. For an overview of MTL, its complexity and decidability see [36]. BLTL formulae can thus be embedded in MTL simply by assuming that the left endpoint of any interval is 0 and the right endpoint is positive, and by disallowing infinite endpoints.

The probabilistic logic we have defined, PBLTL, is a simple extension of BLTL in which formulae have a single outer probabilistic quantifier. The logic PCTL [23] is another popular probabilistic temporal formalism, which modifies (standard) CTL by replacing in state formulae the **A** and **E** path quantifiers with probabilistic quantifiers. In summary, PBLTL offers a more flexible logic for the temporal part (nesting of temporal operators), while PCTL is richer on the probabilistic extension (nesting of probabilistic operators). Also, PCTL allows unbounded Until operators.

Finally, our logics do not currently have support for rewards in the model. The main reason is that we wanted to define a logic as general as possible, so that it could be used on a variety of computational models. However, certain classes of models (e.g., Markov Decision Processes) are often associated with reward structures. Therefore, in a statistical model checking approach to Markov Decision Processes [24], it might be useful to extend our logic with reward operators.

9 Conclusions

Extending our Statistical Model Checking (SMC) algorithm that uses Bayesian Sequential Hypothesis Testing, we have introduced the first SMC algorithm based on Bayesian Interval Estimation. For both algorithms, we have proven analytic bounds on the probability of returning an incorrect answer, which are crucial for understanding the outcome of Statistical Model Checking. We have used SMC for Stateflow/Simulink models of a fuel control system featuring fault-tolerance and hybrid behavior. Because verification is fast in most cases, we expect SMC methods to enjoy good scalability properties for larger Stateflow/Simulink models. Our Bayesian estimation is orders of magnitudes faster than previous estimation-based model checking algorithms.

Acknowledgements This research was sponsored in part by the GigaScale Research Center under contract no. 1041377 (Princeton University), National Science Foundation under contracts no. CNS0926181, CNS0931985, and no. CNS1054246, Semiconductor Research Corporation under contract no. 2005TJ1366, General Motors under contract no. GMCMUCRLNV301, by the US DOT award DTRT12GUTC11, and the Office of Naval Research under award no. N000141010188. This work was carried out while P. Zuliani was at Carnegie Mellon University.

Appendix

A.1 Proofs

In this Section we report proofs for some of the results presented in the paper.

Lemma 1 *Function $K : S \times \mathcal{B}(S) \to [0,1]$ is a stochastic kernel.*

Proof We show that K is in fact a convex combination of two stochastic kernels. It is easy to see that stochastic kernels are closed with respect to convex combinations.

We have already shown above that the discrete part of K, i.e., the summation term, is a stochastic kernel. Because *jump* is a function and x is fixed, $jump_e(x)$ is uniquely determined from x and e, so the argument from above still applies. For continuous transitions, consider any $(q,x) \in S$. Then the following integral

$$\int_0^\infty \Pi_c(q,x)(t) I_B\big(\varphi_q(t,x)\big) dt$$

defines a probability measure over $\mathcal{B}(S)$. Note that $\Pi_c(q,x)$ is a probability density over time, φ_q is a continuous (thus measurable) function of time, and I_B is measurable since $B \in \mathcal{B}(S)$ is a measurable set. Thus, the integral is well-defined and satisfies the usual probability requirements for $B = \emptyset$ and $B = S$. Countable additivity follows easily from Lebesgue's dominated convergence theorem. It remains to prove that, for any $B \in \mathcal{B}(S)$

$$\mathcal{I}(x) = \int_0^\infty \Pi_c(q,x)(t)\ I_B\big(\varphi_q(t,x)\big) dt$$

is a measurable function defined over $\mathbb{R}^n$. Again, note that $\mathcal{I}(x)$ is finite for all x, because the integrand functions are all measurable and integrate up to 1. We recall that the Lebesgue integral of a (non-negative, real) measurable function f with respect to some measure μ, is defined as

$$\int f d\mu = \sup\left\{\int s d\mu : s \text{ simple}, 0 \leqslant s \leqslant f\right\} \tag{16}$$

where a *simple* function takes only finitely many values (piecewise constant). A measurable simple function s can be written as a finite sum $s = \sum_{i=1}^{l} c_i I_{C_i}$, where the c_i's are non-negative reals, and the C_i's are disjoint measurable sets. The integral of s with respect to μ is defined to be the finite sum

$$\int s d\mu = \sum_{i=1}^{l} c_i \mu(C_i)$$

and it is easy to check that the integral does not depend on the particular representation of s.

It can be shown (see for example [12, Proposition 2.3.3]) that the Lebesgue integral (16) is equivalently defined as $\lim_{i\to\infty} \int f_i d\mu$, where $\{f_i\}$ is any non-decreasing sequence of measurable simple functions that converges pointwise to f. Such a sequence can always be found [12, Proposition 2.1.7]. Finally, for any sequence $\{g_i\}$ of (non-negative, real) measurable functions, the function $\lim_{i\to\infty} g_i$ (with domain $\{x \mid \liminf_{i\to\infty} g_i(x) = \limsup_{i\to\infty} g_i(x)\}$), is measurable [12, Proposition 2.1.4]. Therefore, $\mathcal{I}(\cdot)$ is a measurable function. □

Lemma 2 (Bounded sampling) *The problem "$\sigma \models \phi$" is well-defined and can be checked for BLTL formulas ϕ and traces σ based on only a finite prefix of σ of bounded duration.*

Proof According to Lemma 3, the decision "$\sigma \models \phi$" is uniquely determined (and well-defined) by considering only a prefix of σ of duration $\#(\phi) \in \mathbb{Q}_{\geq 0}$. By divergence of time, σ reaches or exceeds this duration $\#(\phi)$ in some finite number of steps n. Let σ' denote a finite prefix of σ of length n such that $\sum_{0\le l<n} t_l \ge \#(\phi)$. Again by Lemma 3, the semantics of $\sigma' \models \phi$ is well-defined because any extension σ'' of σ' satisfies $\sigma'' \models \phi$ if and only if $\sigma' \models \phi$. Consequently the semantics of $\sigma' \models \phi$ coincides with the semantics of $\sigma \models \phi$. On the finite trace σ', it is easy to see that BLTL is decidable by evaluating the atomic formulas $x \sim v$ at each state s_i of the system simulation. □

Lemma 3 (BLTL on bounded simulation traces) *Let ϕ be a BLTL formula, $k \in \mathbb{N}$. Then for any two infinite traces $\sigma = (s_0, t_0), (s_1, t_1), \dots$ and $\tilde{\sigma} = (\tilde{s_0}, \tilde{t_0}), (\tilde{s_1}, \tilde{t_1}), \dots$ with*

$$s_{k+I} = \tilde{s}_{k+I} \quad \textit{and} \quad t_{k+I} = \tilde{t}_{k+I} \quad \forall I \in \mathbb{N} \quad \textit{with} \sum_{0\le l<I} t_{k+l} \le \#(\phi) \tag{17}$$

we have that

$$\sigma^k \models \phi \quad \textit{iff} \quad \tilde{\sigma}^k \models \phi.$$

Proof The proof is by induction on the structure of the BLTL formula ϕ. IH is short for induction hypothesis.

1. If ϕ is of the form $y \sim v$, then $\sigma^k \models y \sim v$ iff $\tilde{\sigma}^k \models y \sim v$, because $s_k = \tilde{s}_k$ by using (17) for $i = 0$.
2. If ϕ is of the form $\phi_1 \vee \phi_2$, then

$$\begin{aligned}
&\sigma^k \models \phi_1 \vee \phi_2\\
&\quad \text{iff} \quad \sigma^k \models \phi_1 \quad \text{or} \quad \sigma^k \models \phi_2\\
&\quad \text{iff} \quad \tilde{\sigma}^k \models \phi_1 \quad \text{or} \quad \tilde{\sigma}^k \models \phi_2 \quad \text{by IH as } \#(\phi_1 \vee \phi_2) \ge \#(\phi_1) \quad \text{and} \quad \#(\phi_1 \vee \phi_2) \ge \#(\phi_2)\\
&\quad \text{iff} \quad \tilde{\sigma}^k \models \phi_1 \vee \phi_2
\end{aligned}$$

   The proof is similar for $\neg\phi_1$ and $\phi_1 \wedge \phi_2$.
3. If ϕ is of the form $\phi_1 \mathbf{U}^t \phi_2$, then $\sigma^k \models \phi_1 \mathbf{U}^t \phi_2$ iff conditions (a), (b), (c) of Definition 6 hold. Those conditions are equivalent, respectively, to the following conditions (a′), (b′), (c′):

   (a′) $\sum_{0\le l<i} \tilde{t}_{k+l} \le t$, because $\#(\phi_1 \mathbf{U}^t \phi_2) \ge t$ such that the durations of trace σ and $\tilde{\sigma}$ are $t_{k+l} = \tilde{t}_{k+l}$ for each index l with $0 \le l < i$ by assumption (17).

   (b′) $\tilde{\sigma}^{k+i} \models \phi_2$ by induction hypothesis as follows: We know that the traces σ and $\tilde{\sigma}$ match at k for duration $\#(\phi_1 \mathbf{U}^t \phi_2)$ and need to show that the semantics of $\phi_1 \mathbf{U}^t \phi_2$ matches at k. By IH we know that ϕ_2 has the same semantics at $k+i$ (that is $\tilde{\sigma}^{k+i} \models \phi_2$ iff $\sigma^{k+i} \models \phi_2$) provided that we can show that the traces σ and $\tilde{\sigma}$ match at $k+i$ for duration $\#(\phi_2)$. For this, consider any $I \in \mathbb{N}$ with $\sum_{0\le l<I} t_{k+i+l} \le \#(\phi_2)$. Then

$$\#(\phi_2) \ge \sum_{0\le l<I} t_{k+i+l} = \sum_{0\le l<i+I} t_{k+l} - \sum_{0\le l<i} t_{k+l} \overset{(a)}{\ge} \sum_{0\le l<i+I} t_{k+l} - t$$

   Thus

$$\sum_{0\le l<i+I} t_{k+l} \le t + \#(\phi_2) \le t + \max\bigl(\#(\phi_1), \#(\phi_2)\bigr) = \#(\phi_1 \mathbf{U}^t \phi_2)$$

As $I \in \mathbb{N}$ was arbitrary, we conclude from this with assumption (17) that, indeed $s_I = \tilde{s}_I$ and $t_I = \tilde{t}_I$ for all $I \in \mathbb{N}$ with

$$\sum_{0 \le l < I} t_{k+i+l} \le \#(\phi_2)$$

Thus the IH for ϕ_2 yields the equivalence of $\sigma^{k+i} \models \phi_2$ and $\tilde{\sigma}^{k+i} \models \phi_2$ when using the equivalence of (a) and (a′).

(c′) for each $0 \le j < i$, $\tilde{\sigma}^{k+j} \models \phi_1$. The proof of equivalence to (c) is similar to that for (b′) using $j < i$.

The existence of an $i \in \mathbb{N}$ for which these conditions (a′), (b′), (c′) hold is equivalent to $\tilde{\sigma}^k \models \phi_1 \mathbf{U}^t \phi_2$. □

Theorem 1 (Termination of Bayesian estimation) *The Sequential Bayesian interval estimation Algorithm 2 terminates with probability one.*

Proof We follow an argument by DeGroot [14, Sect. 10.5]. Let p be the actual probability that the BLTL formula ϕ holds, and let $x_1, \dots, x_n$ a sample given by model checking ϕ over n simulation traces (i.e., iid as the random variable X defined by (2)). Recall that the estimation algorithm returns an interval of width 2δ which contains p with posterior probability at least c, where $\delta \in (0, \frac{1}{2})$ and $c \in (\frac{1}{2}, 1)$ are user-specified parameters. We shall show that the posterior probability of any open interval containing p must converge almost surely to 1, as $n \to \infty$.

Let $\alpha, \beta > 0$ be the parameters of the Beta prior. We know that the posterior X_n given the sample $x_1, \dots, x_n$ has a Beta distribution (5) with parameters $x + \alpha$ and $n - x + \beta$, where $x = \sum_{i=1}^n x_i$. Recall that the posterior mean (8) is:

$$E[X_n] = \hat{p}_n = \frac{x + \alpha}{n + \alpha + \beta}$$

and the posterior variance is (see Appendix A.2):

$$E\big[(X_n - \hat{p}_n)^2\big] = \hat{\sigma}_n^2 = \frac{(x + \alpha)(n - x + \beta)}{(n + \alpha + \beta)^2 (n + \alpha + \beta + 1)}.$$

Because $x \leqslant n$ and $\alpha, \beta > 0$, we have that

$$\hat{\sigma}_n^2 \leqslant \frac{1}{n + \alpha + \beta}$$

and thus $\lim_{n \to \infty} \hat{\sigma}_n^2 = 0$, i.e., the posterior variance tends to 0 as we increase the sample size. (Intuitively, X_n will be arbitrarily close to its mean, as $n \to \infty$.) Also, note that this is everywhere convergence, and not just almost sure convergence.

Now, since α and β are fixed, from the law of large numbers it follows that

$$\lim_{n \to \infty} \hat{p}_n = p \quad \text{(almost surely)} \tag{18}$$

But $\lim_{n \to \infty} \hat{\sigma}_n^2 = \lim_{n \to \infty} E[(X_n - \hat{p}_n)^2] = 0$, so X_n converges almost surely to the *constant* random variable p. Therefore, the posterior probability $P(X_n \in I)$ of any open interval I

containing p must converge almost surely to 1, as $n \to \infty$. Finally, note that the interval returned by the algorithm is always of fixed size $2\delta > 0$. □

Theorem 2 (Error bound for hypothesis testing) *For any discrete random variable and prior, the probability of a Type I-II error for the Bayesian hypothesis testing Algorithm 1 is bounded above by $\frac{1}{T}$, where T is the Bayes Factor threshold given as input.*

Proof We present the proof for Type I error only—for Type II it is very similar. A Type I error occurs when the null hypothesis H_0 is true, but we reject it. We then want to bound $P(\text{reject } H_0 \mid H_0)$. If the Bayesian Algorithm 1 stops at step n, then it will accept H_0 if $\mathcal{B}(d) > T$, and reject H_0 if $\mathcal{B}(d) < \frac{1}{T}$, where $d = (x_1, \dots, x_n)$ is the data sample, and the Bayes Factor is

$$\mathcal{B}(d) = \frac{P(d|H_0)}{P(d|H_1)}.$$

The event {reject H_0} is formally defined as

$$\{\text{reject } H_0\} = \bigcup_{d \in \Omega} \left\{ \mathcal{B}(d) < \frac{1}{T} \wedge D = d \right\} \tag{19}$$

where D is the random variable denoting a sequence of n (finite) discrete random variables, and Ω is the sample space of D—i.e., the (countable) set of all the possible realizations of D (in our case D is clearly finite). We now reason:

$$\begin{aligned}
& P(\text{reject } H_0 \mid H_0) \\
= \quad & && (19) \\
& P\left(\bigcup_{d \in \Omega} \left\{ \mathcal{B}(d) < \frac{1}{T} \wedge D = d \right\} \mid H_0 \right) \\
= \quad & && \text{additivity} \\
& \sum_{d \in \Omega} P\left(\left\{ \mathcal{B}(d) < \frac{1}{T} \wedge D = d \right\} \mid H_0 \right) \\
= \quad & && \text{independent events} \\
& \sum_{d \in \Omega} P\left(\mathcal{B}(d) < \frac{1}{T} \right) \cdot P\big(D = d \mid H_0\big) \\
< \quad & && \mathcal{B}(d) < \frac{1}{T} \quad \text{iff} \quad P\big(D = d \mid H_0\big) < \frac{1}{T} P\big(D = d \mid H_1\big) \\
& \sum_{d \in \Omega} \frac{1}{T} \cdot P\big(D = d \mid H_1\big) \\
= \quad & && \text{additivity and independence} \\
& \frac{1}{T} \cdot P\left(\bigcup_{d \in \Omega} D = d \mid H_1 \right) \\
= \quad & && \text{universal event} \\
& \frac{1}{T} \cdot P\big(\Omega \mid H_1\big) = \frac{1}{T}
\end{aligned}$$

□

Theorem 3 (Error bound for estimation) *For any discrete random variable and prior, the Type I and II errors for the output interval (t_0, t_1) of the Bayesian estimation Algorithm 2* *are bounded above by* $\frac{(1-c)\pi_0}{c(1-\pi_0)}$, *where* c *is the coverage coefficient given as input and* π_0 *is the prior probability of the hypothesis* $H_0 : p \in (t_0, t_1)$.

Proof Let (t_0, t_1) be the interval estimate when the estimation Algorithm 2 terminates (with coverage c). From the hypothesis

$$H_0 : p \in (t_0, t_1) \tag{20}$$

we compute the Bayes factor for H_0 vs. the alternate hypothesis $H_1 : p \notin (t_0, t_1)$. Then we use Theorem 2 to derive the bounds on the Type I and II error. If the estimation Algorithm 2 terminates at step n with output t_0, t_1, we have that:

$$\int_{H_0} f(u|x_1, \dots, x_n)du = \int_{t_0}^{t_1} f(u|x_1, \dots, x_n)du \geqslant c \tag{21}$$

and therefore (since the posterior is a distribution):

$$\int_{H_1} f(u|x_1, \dots, x_n)du \leqslant 1 - c. \tag{22}$$

By (13) we get the Bayes factor of H_0 vs. H_1, which can then be bounded by (21) and (22) as follows

$$\frac{(1-\pi_0)}{\pi_0} \cdot \frac{\int_{H_0} f(u|x_1, \dots, x_n)du}{\int_{H_1} f(u|x_1, \dots, x_n)du} \geqslant \frac{(1-\pi_0)}{\pi_0} \cdot \frac{c}{1-c}.$$

Therefore, by Theorem 2 the error is bounded above by $(\frac{c(1-\pi_0)}{(1-c)\pi_0})^{-1} = \frac{(1-c)\pi_0}{c(1-\pi_0)}$. □

A.2 The Beta Distribution

For the reader's convenience, we calculate the mean and variance of a random variable Y with Beta density of parameters $u, v > 0$. What we outline below can be found in most textbooks on Bayesian statistics and special functions, e.g., [39] and [4].

Recall that the Beta density is

$$\forall t \in (0, 1) \quad g(t, u, v) \mathrel{\widehat{=}} \frac{1}{B(u, v)} t^{u-1}(1-t)^{v-1}$$

where the Beta function $B(u, v)$ is defined as:

$$B(u, v) \mathrel{\widehat{=}} \int_0^1 t^{u-1}(1-t)^{v-1}dt.$$

It is well known that

$$B(u, v) = \frac{\Gamma(u)\Gamma(v)}{\Gamma(u+v)}$$

where $\Gamma(\cdot)$ is Euler's gamma function defined for $z \in \mathbb{C}$ with $\Re(z) > 0$ as:

$$\Gamma(z) \mathrel{\widehat{=}} \int_0^\infty t^{z-1}e^{-t}dt.$$

Also, the gamma function satisfies the equation $\Gamma(z+1) = z\Gamma(z)$, for $\Re(z) > 0$. By means of a few algebraic steps, we are now able to compute the mean of Y:

$$\begin{aligned} E[Y] &= \frac{1}{B(u,v)} \int_0^1 t^u (1-t)^{v-1} dt = \frac{B(u+1,v)}{B(u,v)} \\ &= \frac{\Gamma(u+1)\Gamma(v)}{\Gamma(u+1+v)} \cdot \frac{\Gamma(u+v)}{\Gamma(u)\Gamma(v)} = \frac{u\Gamma(u)\Gamma(u+v)}{(u+v)\Gamma(u+v)\Gamma(u)} = \frac{u}{u+v}. \end{aligned}$$

For the variance, we proceed analogously to show that

$$E\big[Y^2\big] = \frac{1}{B(u,v)} \int_0^1 t^{u+1} (1-t)^{v-1} dt = \frac{(u+1)u}{(u+v+1)(u+v)}$$

and therefore

$$Var[Y] = E\big[Y^2\big] - \big(E[Y]\big)^2 = \frac{uv}{(u+v)^2(u+v+1)}.$$

References

1. Alur R, Courcoubetis C, Dill D (1991) Model-checking for probabilistic real-time systems. In: ICALP. LNCS, vol 510, pp 115–126
2. Baier C, Clarke EM, Hartonas-Garmhausen V, Kwiatkowska MZ, Ryan M (1997) Symbolic model checking for probabilistic processes. In: ICALP. LNCS, vol 1256, pp 430–440
3. Baier C, Haverkort BR, Hermanns H, Katoen J-P (2003) Model-checking algorithms for continuous-time Markov chains. IEEE Trans Softw Eng 29(6):524–541
4. Beals R, Wong R (2010) Special functions. Cambridge University Press, Cambridge
5. Bechhofer R (1960) A note on the limiting relative efficiency of the Wald sequential probability ratio test. J Am Stat Assoc 55:660–663
6. Bujorianu ML, Lygeros J (2006) Towards a general theory of stochastic hybrid systems. In: Blom HAP, Lygeros J (eds) Stochastic hybrid systems: theory and safety critical applications. Lecture notes contr inf, vol 337. Springer, Berlin, pp 3–30
7. Carlin BP, Louis TA (2009) Bayesian methods for data analysis, 3rd edn. CRC Press, Boca Raton
8. Cassandras CG, Lygeros J (eds) (2006) Stochastic hybrid systems. CRC Press, Boca Raton
9. Chadha R, Viswanathan M (2010) A counterexample-guided abstraction-refinement framework for Markov decision processes. ACM Trans Comput Log 12(1):1
10. Chow YS, Robbins H (1965) On the asymptotic theory of fixed-width sequential confidence intervals for the mean. Ann Math Stat 36(2):457–462
11. Ciesinski F, Größer M (2004) On probabilistic computation tree logic. In: Validation of stochastic systems. LNCS, vol 2925. Springer, Berlin, pp 147–188
12. Cohn DL (1994) Measure theory. Birkhäuser, Basel
13. Courcoubetis C, Yannakakis M (1995) The complexity of probabilistic verification. J ACM 42(4):857–907
14. DeGroot MH (2004) Optimal statistical decisions. Wiley, New York
15. Diaconis P, Ylvisaker D (1985) Quantifying prior opinion. In: Bayesian statistics 2: 2nd Valencia international meeting. Elsevier, Amsterdam, pp 133–156
16. Finkbeiner B, Sipma H (2001) Checking finite traces using alternating automata. In: Runtime verification (RV'01). ENTCS, vol 55, pp 44–60
17. Gelman A, Carlin JB, Stern HS, Rubin DB (1997) Bayesian data analysis. Chapman & Hall, London
18. Ghosh MK, Arapostathis A, Marcus SI (1997) Ergodic control of switching diffusions. SIAM J Control Optim 35(6):1952–1988
19. Gillespie DT (1976) A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. J Comput Phys 22(4):403–434
20. Gong H, Zuliani P, Komuravelli A, Faeder JR, Clarke EM (2010) Analysis and verification of the HMGB1 signaling pathway. BMC Bioinform 11(S7):S10
21. Grosu R, Smolka S (2005) Monte Carlo model checking. In: TACAS. LNCS, vol 3440, pp 271–286

22. Hahn EM, Hermanns H, Wachter B, Zhang L (2009) INFAMY: an infinite-state Markov model checker. In: CAV, pp 641–647
23. Hansson H, Jonsson B (1994) A logic for reasoning about time and reliability. Form Asp Comput 6(5):512–535
24. Henriques D, Martins J, Zuliani P, Platzer A, Clarke EM (2012) Statistical model checking for Markov decision processes. In: QEST 2012: Proceedings of the 9th international conference on quantitative evaluation of systems. IEEE Press, New York, pp 84–93
25. Hérault T, Lassaigne R, Magniette F, Peyronnet S (2004) Approximate probabilistic model checking. In: VMCAI. LNCS, vol 2937, pp 73–84
26. Hlavacek WS, Faeder JR, Blinov ML, Posner RG, Hucka M, Fontana W (2006) Rules for modeling signal-transduction system. Sci STKE 18(344):re6
27. Hoeffding W (1963) Probability inequalities for sums of bounded random variables. J Am Stat Assoc 58(301):13–30
28. Jeffreys H (1961) Theory of probability. Clarendon, Oxford
29. Jha SK, Clarke EM, Langmead CJ, Legay A, Platzer A, Zuliani P (2009) A Bayesian approach to model checking biological systems. In: CMSB. LNCS, vol 5688, pp 218–234
30. Koymans R (1990) Specifying real-time properties with metric temporal logic. Real-Time Syst 2(4):255–299
31. Kwiatkowska M, Norman G, Parker D (2011) PRISM 4.0: verification of probabilistic real-time systems. In: CAV. LNCS, vol 6806, pp 585–591
32. Kwiatkowska MZ, Norman G, Parker D (2006) Symmetry reduction for probabilistic model checking. In: CAV. LNCS, vol 4144, pp 234–248
33. Langmead CJ (2009) Generalized queries and Bayesian statistical model checking in dynamic Bayesian networks: application to personalized medicine. In: CSB, pp 201–212
34. Maler O, Nickovic D (2004) Monitoring temporal properties of continuous signals. In: FORMATS. LNCS, vol 3253, pp 152–166
35. Meseguer J, Sharykin R (2006) Specification and analysis of distributed object-based stochastic hybrid systems. In: Hespanha JP, Tiwari A (eds) HSCC, vol 3927. Springer, Berlin, pp 460–475
36. Ouaknine J, Worrell J (2008) Some recent results in metric temporal logic. In: Proc of FORMATS. LNCS, vol 5215, pp 1–13
37. Platzer A (2011) Stochastic differential dynamic logic for stochastic hybrid programs. In: Bjørner N, Sofronie-Stokkermans V (eds) CADE. LNCS, vol 6803. Springer, Berlin, pp 431–445
38. Pnueli A (1977) The temporal logic of programs. In: FOCS. IEEE Press, New York, pp 46–57
39. Robert CP (2001) The Bayesian choice. Springer, Berlin
40. Rubinstein RY, Kroese DP (2008) Simulation and the Monte Carlo method. Wiley, New York
41. Sen K, Viswanathan M, Agha G (2004) Statistical model checking of black-box probabilistic systems. In: CAV. LNCS, vol 3114, pp 202–215
42. Sen K, Viswanathan M, Agha G (2005) On statistical model checking of stochastic systems. In: CAV. LNCS, vol 3576, pp 266–280
43. Shiryaev AN (1995) Probability. Springer, Berlin
44. Tiwari A (2002) Formal semantics and analysis methods for Simulink Stateflow models. Technical report, SRI International
45. Tiwari A (2008) Abstractions for hybrid systems. Form Methods Syst Des 32(1):57–83
46. Wald A (1945) Sequential tests of statistical hypotheses. Ann Math Stat 16(2):117–186
47. Wang Y-C, Komuravelli A, Zuliani P, Clarke EM (2011) Analog circuit verification by statistical model checking. In: ASP-DAC 2011: Proceedings of the 16th Asia and South Pacific design automation conference. IEEE Press, New York, pp 1–6
48. Younes HLS, Kwiatkowska MZ, Norman G, Parker D (2006) Numerical vs statistical probabilistic model checking. Int J Softw Tools Technol Transf 8(3):216–228
49. Younes HLS, Musliner DJ (2002) Probabilistic plan verification through acceptance sampling. In: AIPS workshop on planning via model checking, pp 81–88
50. Younes HLS, Simmons RG (2006) Statistical probabilistic model checking with a focus on time-bounded properties. Inf Comput 204(9):1368–1409
51. Yu PS, Krishna CM, Lee Y-H (1988) Optimal design and sequential analysis of VLSI testing strategy. IEEE Trans Comput 37(3):339–347
52. Zuliani P, Platzer A, Clarke EM (2010) Bayesian statistical model checking with application to Stateflow/Simulink verification. Technical report CMU-CS-10-100, Computer Science Department, Carnegie Mellon University