Bayesian Statistical Model Checking with Application to Stateflow/Simulink Verification

Paolo Zuliani, André Platzer, Edmund M. Clarke

January 13, 2010
CMU-CS-10-100

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research was sponsored by the GSRC (University of California) under contract no. SA423679952, National Science Foundation under contracts no. CCF0429120, no. CNS0926181, no. CCF0541245, and no. CNS0931985, Semiconductor Research Corporation under contract no. 2005TJ1366, General Motors under contract no. GMCMU-CRLNV301, Air Force (University of Vanderbilt) under contract no. 18727S3, International Collaboration for Advanced Security Technology of the National Science Council, Taiwan, under contract no. 1010717, and by the Office of Naval Research under award no. N000141010188.

Keywords: Probabilistic model checking, hybrid systems, fault tolerant, stochastic systems, Bayesian statistics, statistical model checking

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 novel Statistical Model Checking (SMC) approach based on Bayesian statistics. We show that our approach is feasible for 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 or estimation. 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, while retaining the same error bounds. We emphasize that Bayesian SMC is by no means restricted to Stateflow/Simulink models: we have in fact successfully applied it to very large stochastic models from Systems Biology.

1 Introduction

Stochastic effects arise naturally in hybrid control 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 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 certain 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 1ms 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 [3, 11, 1]. 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, s_0 is a starting state, ϕ is a formula in temporal logic, and $\theta \in (0,1)$ is a probability threshold. The PMC problem is: to decide algorithmically whether $\mathcal{M}, s_0 \models P_{\geq\theta}(\phi)$, i.e., to decide whether the model $\mathcal{M}$ starting from 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 [21] in which the temporal operators are equipped with time bounds. Alternatively, BLTL can be viewed as a sublogic of Koymans' Metric Temporal Logic [16, 20]. 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, 5, 6, 13]). Numerical methods are generally only suitable for finite-state systems of about $10^7 - 10^8$ states [17]. In real control systems, the number of states easily exceeds this limit or is infinite, 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 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. We model check each sample trace separately to determine whether the BLTL property ϕ holds, and the number of satisfying traces 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 [12, 23] (in statistics such estimates are known as *confidence intervals*). In the second case, the PMC problem is directly treated as a *hypothesis testing* problem (e.g., [27, 28]), i.e., deciding between the hypothesis $H_0 : \mathcal{M} \models P_{\geq\theta}(\phi)$ that the property holds versus the hypothesis $H_1 : \mathcal{M} \models P_{<\theta}(\phi)$ that it does not.

Hypothesis-testing based methods are more efficient than those based on estimation when 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 s_0) [26]. In this paper we show that estimation can be much faster for probabilities close to 1. Also note that Statistical Model Checking cannot guarantee a correct answer to the PMC problem. The most crucial question needed to obtain meaningful results is whether the probability that the algorithm gives a wrong answer can be bounded. We prove that this error probability can indeed be bounded arbitrarily by the user.

Our SMC approach 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, but our algorithms instead 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). While our sequential sampling has many practical advantages compared to fixed-size sampling, its theoretical analysis is significantly more challenging.

We apply our approach to a representative example of discrete-time stochastic hybrid system models in Stateflow/Simulink: a fault-tolerant fuel control (hybrid) system. We show that our approach enables faster verification than state-of-the-art techniques based on statistical methods.

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 give the first application of Bayesian sequential interval estimation to Statistical Model Checking.
- We prove analytic error bounds for our Bayesian sequential hypothesis testing and estimation algorithms.
- In a series of experiments with different parameterizations of a relevant Simulink/Stateflow 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 Statistical Model Checking is very challenging, a beneficial property of our algorithms is that they are easy to implement.

2 Background

Our algorithm can be applied to any stochastic model $\mathcal{M}$ for which it is possible to define a probability space over its traces. Several stochastic models like discrete/continuous Markov chains satisfy this property [28]. Here we use discrete-time hybrid systems a la Stateflow/Simulink with probabilistic transitions.

Discrete Time Hybrid Systems with Probabilistic Transitions As a system model, we consider discrete time hybrid systems with additional probabilistic transitions (our case study uses Stateflow/Simulink). Such a model $\mathcal{M}$ 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). For Stateflow/Simulink, a *state* assigns real values to all the state variables and identifies the current discrete state (or location) for Stateflow machines.

Formally, we start with a definition of a deterministic automaton. Then we augment it with probabilistic transitions.

Definition 1. *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 $(q_0, x_0) \in Q \times \mathbb{R}^n$;*
- *flows $\varphi_q(t;x) \in \mathbb{R}^n$, representing the 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 $e \in E$. We assume $jump_e$ to be measurable (preimages of measurable sets under $jump_e$ are measurable).*

Definition 2. *The* transition relation *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 discrete or continuous transition to execute from the respective state (q,x). For Stateflow/Simulink, Δ satisfies several properties, including that the first edge e (in clockwise orientation in the graphical notation) 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.

Each execution of a DTHA is obtained by following the transition relation 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 $i \in \mathbb{N}$, $s_i \rightarrow_{\Delta(s_i)} s_{i+1}$ 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-zeno*). We denote $\sum_{0 \leq l < i} t_l$ by $\tau(x_i)$, because we can assume there is one state variable tracking global time.

A *probabilistic* DTHA is obtained from a DTHA by means of a probabilistic simulation function instead of Δ. Unlike Δ, it selects discrete and continuous transitions according to a probability density. The *state* of a probabilistic DTHA is a probability density function on $Q \times \mathbb{R}^n$. We denote the set of these functions by $D(Q \times \mathbb{R}^n)$.

Definition 3. *The* transition function *for a* probabilistic *DTHA, which we denote by $\rightarrow$, maps a (probabilistic) state $p \in D(Q \times \mathbb{R}^n)$ to $\tilde{p} \in D(Q \times \mathbb{R}^n)$ with $\tilde{p}(\tilde{q},\tilde{x})$ defined as:*

$$\int_{\mathbb{R}_{\geq 0} \cup E} \int_{Q \times \mathbb{R}^n} p(q,x)\Pi(q,x)(\alpha) I_{\rightarrow_{\alpha}(\tilde{q},\tilde{x})}(q,x)\, d(q,x)\, d\alpha$$

where

- $\Pi : Q \times \mathbb{R}^n \rightarrow D(\mathbb{R}_{\geq 0} \cup E)$ *is the (measurable)* probabilistic simulation *function;*
- $I_{\rightarrow_{\alpha}(\tilde{q},\tilde{x})}$ *is the indicator function of the preimage of $\rightarrow_\alpha$ at $(\tilde{q},\tilde{x})$, i.e., $I_{\rightarrow_{\alpha}(\tilde{q},\tilde{x})}(q,x) = 1$ iff $(q,x) \rightarrow_\alpha (\tilde{q},\tilde{x})$, and 0 otherwise; $\rightarrow_\alpha$ is as per Definition 2.*

Well-definedness of the integral in Def. 3 follows directly from measurability of Π and the jump functions, plus the fact that integration over time can be restricted to a bounded interval from 0 to the current time $\tau(\tilde{x})$. Note that initial distributions on the initial state can be obtained easily by prefixing the system with a probabilistic transition from x_0. Sample traces of a probabilistic DTHA can be obtained by sampling from the traces generated by Π.

Specifying Properties in Temporal Logic Our algorithm verifies properties of $\mathcal{M}$ expressed as formulas in *Probabilistic Bounded Linear Temporal Logic* PBLTL). We first define the syntax and semantics of *Bounded Linear Temporal Logic* (BLTL), which we can check on a single trace, and then extend that logic to PBLTL. Finkbeiner and Sipma [8] 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:

$$\phi ::= y{\sim}v \,|\, (\phi_1 \vee \phi_2) \,|\, (\phi_1 \wedge \phi_2) \,|\, \neg\phi_1 \,|\, (\phi_1 \mathbf{U^t} \phi_\mathbf{2}),$$

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 $\mathbf{F^t}\psi = \mathbf{True}\,\mathbf{U^t}\,\psi$, or $\mathbf{G^t}\psi = \neg\mathbf{F^t}\neg\psi$ by bounded untils $\mathbf{U^t}$.

We define the semantics of BLTL with respect to executions of $\mathcal{M}$. 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 execution σ). We denote the value of the state variable y in σ at step i by $V(\sigma, i, y)$.

Definition 4. *The* semantics *of BLTL for a trace σ^k starting at the k^{th} 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) for each* $0 \leq j < i$*,* $\sigma^{k+j} \models \phi_1$.

Statistical Model Checking decides probabilistic Model Checking 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. If σ is too short, say of duration 2, the semantics of $\phi_1 \mathbf{U}^5 \phi_2$ may be unclear. 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 [19]. 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. It especially turns out that divergence of time ensures termination of SMC.

Lemma 1 (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 1 we need to derive bounds on when to stop simulation. Those bounds can be read off easily from the BLTL formula:

Definition 5. *We define the* sampling bound $\#(\phi) \in \mathbb{Q}_{\geq 0}$ *of a BLTL formula* ϕ *inductively as the maximum nested sum of time bounds:*

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

Unlike infinite traces, actual system simulations need to be finite in length. We prove that the semantics of BLTL formulas ϕ is well-defined on finite prefixes of traces with a duration that is bounded by $\#(\phi)$.

Lemma 2 (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*

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

we have that

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

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 (1) for $i = 0$.

2. If ϕ is of the form $\phi_1 \vee \phi_2$, then

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

   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 4 hold. Those conditions are equivalent, respectively, to the following conditions (a'),(b'),(c'):

   (a') $\sum_{0 \leq l < i} \tilde{t}_{k+l} \leq t$, because $\#(\phi_1 \mathbf{U}^t \phi_2) \geq t$ such that the durations of trace σ and $\tilde{\sigma}$ are $t_{k+l} = \tilde{t}_{k+l}$ for each index l with $0 \leq l < i$ by assumption (1).

(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

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

Thus

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

As $I \in \mathbb{N}$ was arbitrary, we conclude from this with assumption (1) 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 hold is equivalent to $\tilde{\sigma}^k \models \phi_1 \mathbf{U}^t \phi_2$. □

Now we prove that Lemma 1 holds using prefixes of traces according to the sampling bound $\#(\phi)$, which guarantees that finite simulations are sufficient for deciding ϕ.

Proof of Lemma 1. According to Lemma 2, the decision "$\sigma \models \phi$" is uniquely determined (and well-defined) by considering only a prefix of σ of duration $\#(\phi) \in \mathbb{Q}_{\ge 0}$. 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 2, 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. □

We now define Probabilistic Bounded Linear Temporal Logic.

Definition 6. *A Probabilistic Bounded LTL (PBLTL) formula is a formula of the form $P_{\ge \theta}(\phi)$, 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, because, by Lemma 1, each $\sigma \models \phi$ is decidable on a finite prefix of σ, finite iterations of the probabilistic transition function (Def. 3) gives a well-defined probability measure, and, thus, a corresponding probability measure can be associated to the set of all (non-zeno) executions of $\mathcal{M}$ that satisfy a BLTL formula [28]. 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 $P_{\geq\theta}(\phi)$.

3 Bayesian Interval Estimation

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

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:

$$H_0 : p \geqslant \theta \qquad H_1 : p < \theta. \tag{2}$$

For any trace σ_i of the system $\mathcal{M}$, we can 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 mass function associated with X_i is thus:

$$\forall u \in [0,1] \quad f(x_i|u) = u^{x_i}(1-u)^{1-x_i} \tag{3}$$

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), as each trace is given by an independent execution of the model. Since p is unknown, we may assume that it 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 [22, Section 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):

$$\forall u \in [0,1] \quad g(u,\alpha,\beta) \mathrel{\widehat{=}} \frac{1}{B(\alpha,\beta)} u^{\alpha-1}(1-u)^{\beta-1} \tag{4}$$

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

$$B(\alpha,\beta) \;\widehat{=}\; \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}dt \;. \tag{5}$$

By varying the parameters α and β, one can approximate 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:

$$F_{(\alpha,\beta)}(u) \;\widehat{=}\; \int_0^u g(t,\alpha,\beta)\; dt = \frac{1}{B(\alpha,\beta)}\int_0^u t^{\alpha-1}(1-t)^{\beta-1}\; dt \tag{6}$$

which is the usual 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).

In addition to their flexible shapes for various choices of α,β, the advantage of using Beta densities is that the Beta distribution is the *conjugate prior* to the Bernoulli distribution[1]. This relationship enables us to avoid numerical integration in the implementation of both the Bayesian estimation and hypothesis testing algorithms, as we next explain.

3.1 Bayesian Intervals

Bayes' theorem states that if we sample from a density $f(\cdot|u)$, where u (the unknown probability) 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:

$$f(u|x_1,\ldots,x_n) = \frac{f(x_1,\ldots,x_n|u)g(u)}{\int_0^1 f(x_1,\ldots,x_n|v)g(v)\; dv} \tag{7}$$

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 Bernoulli mass function (3) 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 *mean* of the posterior. 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 [7, Chapter 8]).

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

$$\int_{t_0}^{t_1} f(u|x_1,\ldots,x_n)\; du = c \tag{8}$$

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 (8), thus determining an optimal interval. Note that t_0 and t_1 are in fact functions of the sample $x_1,\ldots,x_n$.

[1]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.

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, 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 probability mass of an interval estimate 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:

$$\hat{p} = \frac{x+\alpha}{n+\alpha+\beta} \tag{9}$$

where $x = \sum_{i=1}^{n} x_i$ is the number of successes in the sampled data $x_1, \ldots, x_n$. The integral in (8) can further 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 $(x_1, \ldots, x_n)$ and Beta prior of parameters α, β can be calculated as:*

$$\int_{t_0}^{t_1} f(u|x_1, \ldots, x_n)\, du = F_{(x+\alpha, n-x+\beta)}(t_1) - F_{(x+\alpha, n-x+\beta)}(t_0) \tag{10}$$

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

Proof. Direct from definition of Beta distribution function (6) 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, the Beta distribution is the appropriate choice for summarizing the prior distribution in Statistical Model Checking.

3.2 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 Section 2 that p is well-defined. Fix the half-size $\delta \in (0, \frac{1}{2})$ of the desired interval estimate for p, the coefficient $c \in (\frac{1}{2}, 1)$ to be used in (8), and the coefficients α, β of the Beta prior.

Our algorithm iteratively draws iid sample traces $\sigma_1, \sigma_2, \ldots$, and checks whether they satisfy ϕ. At stage n, the algorithm computes $\hat{p}$, the Bayes estimator for p (i.e., the posterior mean) according to (9). Next, using $t_0 = \hat{p} - \delta$, $t_1 = \hat{p} + \delta$ it computes

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

If $\gamma \geqslant c$ it 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 Algorithm 1.

Algorithm 1 Statistical Model Checking by Bayesian Interval Estimates

Require: BLTL Property ϕ, half-interval size $\delta \in (0, \frac{1}{2})$, interval coefficient $c \in (\frac{1}{2}, 1)$, Prior Beta distribution with parameters α, β

```
$n := 0$                                   {number of traces drawn so far}
$x := 0$                                   {number of traces satisfying $\phi$ so far}
repeat
    $\sigma$ := draw a sample trace of the system (iid)
    $n := n + 1$
    if $\sigma \models \phi$ then
        $x := x + 1$
    end if
    $\hat{p} := (x + \alpha)/(n + \alpha + \beta)$          {compute posterior mean}
    $(t_0, t_1) := (\hat{p} - \delta, \hat{p} + \delta)$          {compute interval estimate}
    if $t_1 > 1$ then
        $(t_0, t_1) := (1 - 2 \cdot \delta, 1)$
    else if $t_0 < 0$ then
        $(t_0, t_1) := (0, 2 \cdot \delta)$
    end if
    $\gamma$ := PosteriorProb$(t_0, t_1)$          {compute posterior probability of $p \in (t_0, t_1)$, by (10)}
until $(\gamma \geqslant c)$
return $(t_0, t_1), \hat{p}$
```

4 Bayesian Hypothesis Testing

In this section we briefly present our sequential Bayesian hypothesis test, which was introduced in [15]. Let $X_1, \ldots, X_n$ be a sequence of Bernoulli random variables defined as for the PMC problem in Sect. 3, 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 (2). 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

$$P(H_0|d) = \frac{P(d|H_0)P(H_0)}{P(d)} \qquad P(H_1|d) = \frac{P(d|H_1)P(H_1)}{P(d)} \tag{11}$$

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).

4.1 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 7. *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 [14]. 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 7, we have to calculate the ratio of the probabilities of the observed sample $d = (x_1, \ldots, x_n)$ 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 (7) by integrating the joint density $f(x_1|\cdot)\cdots f(x_n|\cdot)$ with respect to the prior $g(\cdot)$:

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

Thus, the Bayes factor is:

$$\mathcal{B} = \frac{\pi_1}{\pi_0} \cdot \frac{P(H_0|x_1,\ldots,x_n)}{P(H_1|x_1,\ldots,x_n)} = \frac{\pi_1}{\pi_0} \cdot \frac{\int_\theta^1 f(x_1|u)\cdots f(x_n|u)\cdot g(u)\,du}{\int_0^\theta f(x_1|u)\cdots f(x_n|u)\cdot g(u)\,du} \tag{13}$$

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 $x_1, \ldots, x_n$, 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 $(x_1, \ldots, x_n)$ 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 $(x_1, \ldots, x_n)$ and $F_{(s,t)}(\cdot)$ is the Beta distribution function of parameters s, t.

4.2 Bayesian Hypothesis Testing Algorithm

Our algorithm generalizes Jeffreys' test to a sequential version. 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 $\sigma_1, \sigma_2, ...$, and checks whether they satisfy ϕ. 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. 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 $\mathcal{B} > T$, and accepts H_1 iff $\mathcal{B} < \frac{1}{T}$. Otherwise ($\frac{1}{T} \leqslant \mathcal{B} \leqslant T$) it continues drawing iid samples. This algorithm is shown in Algorithm 2.

Algorithm 2 Statistical Model Checking by Bayesian Hypothesis Testing

Require: PBLTL Property $P_{\geqslant\theta}(\phi)$, Threshold $T \geqslant 1$, Prior density g for unknown parameter p

```
$n := 0$                                   {number of traces drawn so far}
$x := 0$                                   {number of traces satisfying $\phi$ so far}
loop
   $\sigma$ := draw a sample trace of the system (iid)
   $n := n+1$
   if $\sigma \models \phi$ then
      $x := x+1$
   end if
   $\mathcal{B}$ := BayesFactor$(n,x)$          {compute as in Proposition 2}
   if $(\mathcal{B} > T)$ then
      return $H_0$ accepted
   else if $(\mathcal{B} < \frac{1}{T})$ then
      return $H_1$ accepted
   end if
end loop
```

5 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 outside the sampled traces, unless the probability of making an error during the PMC decision can be bounded.

As our main contribution, we prove error bounds for Statistical Model Checking by Bayesian sequential hypothesis testing and by Bayesian interval estimation. In particular, we show that the (Bayesian) Type I-II error probabilities for the algorithms in Sect. 3–4 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 1 (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 2 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 2 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 $d = (x_1, \ldots, x_n)$ is the data sample, and the Bayes Factor is

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

The event $\{\text{reject } H_0\}$ is formally defined as

$$\{\text{reject } H_0\} = \bigcup_{d \in \Omega} \{\mathcal{B}(d) < \frac{1}{T} \wedge D = d\} \tag{14}$$

where D is the random variable denoting a sequence of n 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:

$P(\text{reject } H_0 \mid H_0)$

= (14)

$P(\bigcup_{d\in\Omega}\{\mathcal{B}(d) < \frac{1}{T} \wedge D = d\} \mid H_0)$

= additivity

$\sum_{d\in\Omega} P(\{\mathcal{B}(d) < \frac{1}{T} \wedge D = d\} \mid H_0)$

= independent events

$\sum_{d\in\Omega} P(\mathcal{B}(d) < \frac{1}{T}) \cdot P(D = d \mid H_0)$

¡ $\mathcal{B}(d) < \frac{1}{T}$ iff $P(D = d \mid H_0) < \frac{1}{T} P(D = d \mid H_1)$

$\sum_{d\in\Omega} \frac{1}{T} \cdot P(D = d \mid H_1)$

= additivity and independence

$\frac{1}{T} \cdot P(\bigcup_{d\in\Omega} D = d \mid H_1)$

= universal event

$\frac{1}{T} \cdot P(\Omega \mid H_1) = \frac{1}{T}$

□

Note that the bound $\frac{1}{T}$ is independent from the prior used.

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

Theorem 2 (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 1 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 1 terminates (with coverage c). From the hypothesis

$$H_0 : p \in (t_0,t_1) \tag{15}$$

we compute the Bayes factor for H_0 vs. the alternate hypothesis $H_1 : p \notin (t_0,t_1)$. Then we use Theorem 1 to derive the bounds on the Type I and II error. If the estimation algorithm 1 terminates at step n with output t_0,t_1, we have that:

$$\int_{H_0} f(u|x_1,\ldots,x_n)\,du = \int_{t_0}^{t_1} f(u|x_1,\ldots,x_n)\,du \geqslant c \tag{16}$$

and therefore (since the posterior is a distribution):

$$\int_{H_1} f(u|x_1,\ldots,x_n)\,du \leqslant 1-c. \tag{17}$$

The Bayes factor of H_0 vs H_1 is, by (13):

$$\begin{aligned}
&\frac{(1-\pi_0)}{\pi_0}\cdot\frac{\int_{H_0} f(u|x_1,\ldots,x_n)\,du}{\int_{H_1} f(u|x_1,\ldots,x_n)\,du} \\
&\geqslant && \text{by (16) and (17)} \\
&\frac{(1-\pi_0)}{\pi_0}\cdot\frac{c}{1-c}
\end{aligned}$$

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

6 Application

We study an example that is part of the Stateflow/Simulink package. The model[2] describes a fuel controller system for a gasoline engine. It detects sensor failures, and dynamically changes 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 - Exahust Gas Oxygen (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 2.4GHz Pentium 4, 1GB RAM desktop computer running Matlab R2008b on Windows XP.

6.1 Experimental Results in Application

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

$$H_0 : \mathcal{M} \models P_{\geq\theta}(\neg \mathbf{F}^{100}\mathbf{G}^{1}(FuelFlowRate = 0)) \tag{18}$$

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 highly concentrated around the true probability of the model satisfying the BLTL formula. The Bayes Factor threshold is $T = 1000$, so by Theorem 1 both Type I and II errors are bounded by .001.

[2]More information on the model is available at `http://mathworks.com/products/simulink/demos.html?file=/products/demos/shipping/simulink/sldemo_fuelsys.html`.

		Probability threshold θ				
		.5	.7	.8	.9	.99
Fault rates	(3 7 8)	✗ (58/124s)	✗ (17/40s)	✗ (10/25s)	✗ (8/21s)	✗ (2/5s)
	(10 8 9)	✓ (32/78s)	✓ (95/225s)	✓ (394/1013s)	✗ (710/1738s)	✗ (8/21s)
	(20 10 20)	✓ (9/21s)	✓ (16/36s)	✓ (24/54s)	✓ (44/100s)	✗ (1626/3995s)
	(30 30 30)	✓ (9/24s)	✓ (16/41s)	✓ (24/59s)	✓ (44/107s)	✓ (239/589s)

Table 1: Number of samples / verification time when testing (18) with uniform, equally likely priors and $T = 1000$: ✗ = 'H_0 rejected', ✓ = 'H_0 accepted'.

		Probability threshold θ				
		.5	.7	.8	.9	.99
Fault rates	(3 7 8)	✗ (55/117s)	✗ (12/28s)	✗ (10/25s)	✗ (8/21s)	✗ (2/5s)
	(10 8 9)	✓ (28/69s)	✓ (64/150s)	✓ (347/876s)	✗ (255/632s)	✗ (8/21s)
	(20 10 20)	✓ (8/18s)	✓ (13/30s)	✓ (20/45s)	✓ (39/88s)	✗ (1463/3613s)
	(30 30 30)	✓ (7/18s)	✓ (13/34s)	✓ (18/45s)	✓ (33/80s)	✓ (201/502s)

Table 2: Number of samples / verification time when testing (18) with informative priors and $T = 1000$: ✗ = 'H_0 rejected', ✓ = 'H_0 accepted'.

In Table 1 and Table 2 we report our results. Even the longest test (for $\theta = .99$ and fault rates $(20\ 10\ 20)$ in Table 1) Bayesian SMC terminates after 3995s already. This is very good performance for a test with such a small (.001) error probability run on a standard desktop computer. We note the total time spent for this case on actually computing the statistical test i.e., Bayes factor computation, was just about 1s. Also, by comparing the numbers 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:

$$\mathcal{M} \models (\neg \mathbf{F}^{100}\mathbf{G}^{1}(FuelFlowRate = 0)) \,. \qquad (19)$$

In particular, we ran two sets of tests, one with half-interval size $\delta = .05$ and another with $\delta = .01$. In each set we used different values for the interval coefficient c and different sensor fault rates, as before. Experimental results are in Table 3 and 4. We used uniform priors in both cases.

6.2 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

		Interval coverage c			
		.9	.95	.99	.999
Fault rates	(3 7 8)	.4 / 258	.376 / 357	.3569 / 606	.3429 / 972
	(10 8 9)	.8857 / 103	.8904 / 144	.8785 / 286	.8429 / 590
	(20 10 20)	.9565 / 21	.9667 / 28	.9561 / 112	.9625 / 158
	(30 30 30)	.9565 / 21	.9667 / 28	.9778 / 43	.9851 / 65
	samples needed in [12]	4793	5902	8477	12161

Table 3: Posterior mean / number of samples for estimating probability of (19) with uniform prior and $\delta = .05$, and comparison with the samples needed by the Chernoff-Hoeffding bound.

		Interval coverage c			
		.9	.95	.99	.999
Fault rates	(3 7 8)	.3603/6234	.3559/8802	.3558/15205	.3563/24830
	(10 8 9)	.8534/3381	.8518/4844	.8528/8331	.8534/13569
	(20 10 20)	.9764/592	.9784/786	.9840/1121	.9779/2583
	(30 30 30)	.9913/113	.9933/148	.9956/227	.9971/341
	samples needed in [12]	119829	147555	211933	304036

Table 4: Posterior mean / number of samples when estimating probability of (19) with uniform prior and $\delta = .01$, and comparison with the samples needed by the Chernoff-Hoeffding bound.

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 [12]. From Table 3 and 4 we see that when the unknown probability is close to 1, our algorithm can be between two and three 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 builds the posterior distribution on the basis of the Bernoulli sampling distribution and the prior.

6.3 Performance Evaluation

We have conducted a series of 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 .01 to .99, with coverage (c) of .9999 and .99999, interval half-size (δ) of .001 and .005, and uniform prior. We present details of our simulations in Figure 1.

Our Simulink experiments show that Bayesian estimation is very fast when p is close to either 0 or 1, while the algorithm needs a larger number of samples 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

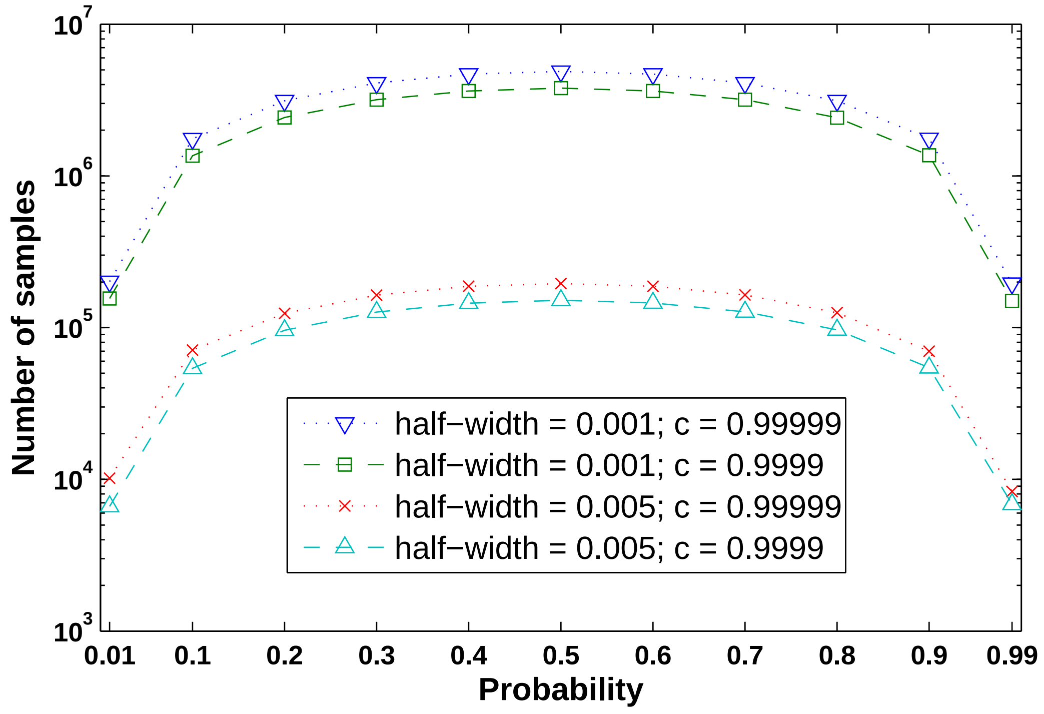

Figure 1: Performance of Bayesian estimation: number of samples vs probability

that it is not currently matched by other SMC estimation techniques (e.g., [12]). Our findings are consistent with those of Yu *et al.* for the VLSI testing domain [29].

Our simulations also indicate that the performance of the algorithm depends more strongly on the half-size δ of the estimated interval than on the coverage c of the interval itself. It is much faster to estimate an interval of half-size $\delta = .005$ with coverage $c = .99999$ than it is to estimate an interval of $\delta = .001$ with $c = .9999$. More theoretical work is needed, however, to understand fully the behavior of the Bayesian sequential estimation algorithm. Our initial findings suggest that the algorithm scales very well.

7 Related Work

Younes and Simmons introduced the first algorithm for Statistical Model Checking [27, 28]. Their work uses the SPRT [25], which is designed for *simple* hypothesis testing[3]. 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,

[3]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 $p < 0.3$.

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 [25]. 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 [9, Section 3.4]. 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 - see [4] and [9, Section 3.6]. Our approach solves instead the composite hypothesis testing problem, with no indifference region.

The method of [12] 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 Chernoff-Hoeffding bounds. 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 $\varepsilon > 0$ 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 [10]. 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.* [23] 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.* [24] have started investigating the extension of SMC to unbounded (i.e., standard) LTL properties. Finally, Langmead [18] has applied Bayesian point estimation and SMC for querying Dynamic Bayesian Networks.

8 Conclusions and Future Work

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.

References

[1] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for probabilistic real-time systems. In *ICALP*, volume 510 of *LNCS*, pages 115–126, 1991.

[2] C. Baier, E. M. Clarke, V. Hartonas-Garmhausen, M. Z. Kwiatkowska, and M. Ryan. Symbolic model checking for probabilistic processes. In *ICALP*, volume 1256 of *LNCS*, pages 430–440, 1997.

[3] C. Baier, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Trans. Software Eng.*, 29(6):524–541, 2003.

[4] R. Bechhofer. A note on the limiting relative efficiency of the Wald sequential probability ratio test. *J. Amer. Statist. Assoc.*, 55:660–663, 1960.

[5] F. Ciesinski and M. Größer. On probabilistic computation tree logic. In *Validation of Stochastic Systems*, LNCS, 2925, pages 147–188. Springer, 2004.

[6] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.

[7] M. H. DeGroot. *Optimal Statistical Decisions*. Wiley, 2004.

[8] B. Finkbeiner and H. Sipma. Checking finite traces using alternating automata. In *Runtime Verification (RV '01)*, volume 55(2) of *ENTCS*, pages 44–60, 2001.

[9] B. Ghosh and P. Sen, editors. *Handbook of sequential analysis*. Dekker, 1991.

[10] R. Grosu and S. Smolka. Monte Carlo Model Checking. In *TACAS*, volume 3440 of *LNCS*, pages 271–286, 2005.

[11] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.*, 6(5):512–535, 1994.

[12] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. In *VMCAI*, volume 2937 of *LNCS*, pages 73–84, 2004.

[13] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In *TACAS*, volume 3920 of *LNCS*, pages 441–444, 2006.

[14] H. Jeffreys. *Theory of Probability*. Clarendon, 1961.

[15] S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani. A Bayesian approach to Model Checking biological systems. In *CMSB*, volume 5688 of *LNCS*, pages 218–234, 2009.

[16] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-time Systems*, 2(4):255–299, 1990.

[17] M. Z. Kwiatkowska, G. Norman, and D. Parker. Symmetry reduction for probabilistic model checking. In *CAV*, volume 4144 of *LNCS*, pages 234–248, 2006.

[18] C. J. Langmead. Generalized queries and Bayesian statistical model checking in dynamic Bayesian networks: Application to personalized medicine. In *Computational Systems Bioinformatics (CSB)*, pages 201–212, 2009.

[19] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *FORMATS*, volume 3253 of *LNCS*, pages 152–166, 2004.

[20] J. Ouaknine and J. Worrell. Some recent results in metric temporal logic. In *Proc. of FORMATS*, volume 5215 of *LNCS*, pages 1–13, 2008.

[21] A. Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57. IEEE, 1977.

[22] C. P. Robert. *The Bayesian Choice*. Springer, 2001.

[23] K. Sen, M. Viswanathan, and G. Agha. Statistical model checking of black-box probabilistic systems. In *CAV*, volume 3114 of *LNCS*, pages 202–215, 2004.

[24] K. Sen, M. Viswanathan, and G. Agha. On statistical model checking of stochastic systems. In *CAV*, volume 3576 of *LNCS*, pages 266–280, 2005.

[25] A. Wald. Sequential tests of statistical hypotheses. *Ann. Math. Statist.*, 16(2):117–186, 1945.

[26] H. L. S. Younes, M. Z. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking. *STTT*, 8(3):216–228, 2006.

[27] H. L. S. Younes and R. G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *CAV*, volume 2404 of *LNCS*, pages 223–235, 2002.

[28] H. L. S. Younes and R. G. Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Inf. Comput.*, 204(9):1368–1409, 2006.

[29] P. S. Yu, C. M. Krishna, and Y.-H. Lee. Optimal design and sequential analysis of VLSI testing strategy. *IEEE T. Comput.*, 37(3):339–347, 1988.