Probabilistic Safety Verification of Distributed Systems: A Statistical Approach for Monitoring*

Bineet Ghosh[1] and Étienne André[2,3]

[1] The University of Alabama, AL, The United States of America
[2] Université Sorbonne Paris Nord, LIPN, CNRS UMR 7030, Villetaneuse, France
[3] Institut universitaire de France (IUF)

Abstract. With the increasing autonomous capabilities of distributed cyber-physical systems, the complexity of their models also increases significantly, thus continually posing challenges to existing formal methods for safety verification. In contrast to model checking, monitoring emerges as an effective lightweight, yet practical verification technique capable of delivering results of practical importance with better scalability. Monitoring involves analyzing logs from an actual system to determine whether a specification (such as a safety property) is violated. Monitoring techniques, such as those using reachability methods, may fail to produce results when dealing with complex models like Deep Neural Networks (DNNs). We propose here a novel statistical approach for monitoring that is able to generate results with high probabilistic guarantees. We evaluate our monitoring technique on three case studies.

Keywords: Autonomous Systems · Monitoring · Statistical Verification

1 Introduction

Over the past decade, there has been a swift increase in the deployment of distributed autonomous systems across various domains. While formal methods have proven successful in various safety-critical domains, several existing techniques have been unable to cope with the growing complexity of these models. Formal verification requires an accurate model, which may not often be available, because some components are black-box, or because the entire system has no formal model. Despite some success in verifying formal models from the industry (e.g., [3,9,20,22]), formal verification techniques for autonomous systems are often subject to state space explosion and conservative analysis.

Monitoring Contrary to model checking, monitoring is an effective lightweight, yet feasible verification technique, that can bring practical results for systems with relatively high complexity. Monitoring involves analyzing the log of a system to determine if a given specification has been violated [7]. Deterministic

*This work is partially supported by ANR BisoUS (ANR-22-CE48-0012). Artifacts available in https://doi.org/10.5281/zenodo.15190263.

monitoring—where safety guarantees that are not probabilistic but formally ensured—encounters challenges in dealing with complex models like black-box systems, DNN-based models, and Simulink models. These challenges include limited adaptability to complex models, difficulties in achieving black-box transparency, and scalability issues [7, 8, 15, 24, 25, 30].

Proposed Approach and Statistical Hypothesis Testing This work, on the other hand, aims to ensure that autonomous systems are *trustworthy*, which means they are safe for most practical purposes with a high level of confidence. This approach contrasts with approaches that strive for formal safety, which might not be practical for complex systems. As such, we propose a statistical monitoring approach that employs Bayesian hypothesis testing [13, 21, 36]. By trading-off formal safety guarantees with statistical guarantees, our technique has the following advantages: *i*) Our method is versatile and can handle various system types, including non-linear and DNNs models. This is achieved merely by leveraging the knowledge of the system's input/output execution behavior (with detailed explanations provided later). In simpler terms, our approach simply assumes that random executions (i.e., trajectories) of the system can be generated from a specified initial state. We refer to this representation as *I/O execution*. *ii*) The user can adjust the confidence level needed for the analysis based on the specific use case. For example, in a safety critical scenario, a high confidence level (e.g., 0.99) may be chosen, while in less critical situations, a lower confidence level can be selected. Our method requires less time for monitoring as the desired confidence level decreases. *iii*) If the I/O execution is computationally efficient for a given system model, this method scales well even when the system model is complex.

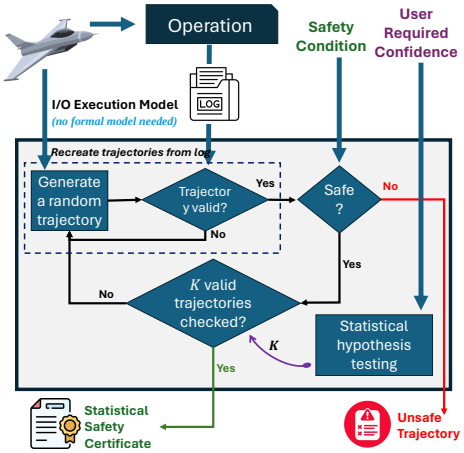


Fig. 1: **Proposed Monitoring Approach**

Our approach is illustrated in Fig. 1. After the system completes its real-world operation, it produces a log that comprises of both noisy and missing records. The noise in the log may result from sensor uncertainties, while missing records could be attributed to transmission losses over a shared network or the system's energy-saving policies (especially in intermittently powered devices). The task of monitoring is to analyze this log for possible safety violations. Once the inputs are received (I/O execution model, log, safety condition), the proposed approach performs a series of steps to analyze the log for safety violation. Before discussing the steps of the proposed approach, it is worth recalling that our approach

is statistical, and as such, the assurances it offers are inherently probabilistic. Specifically, if the approach identifies a trajectory breaching the safety condition, it is reported as a counterexample. Upon discovering such a counterexample, the intrinsic nature of our method ensures that it is concrete—meaning it is a real execution of the actual system, reconstructed from the available logs, that indeed violates the safety condition. Consequently, the system is confirmed to be unsafe. However, if no violating trajectory is found, it infers that the system is safe *with probabilistic guarantees*. The key challenge in monitoring a given uncertain system log with missing records lies in efficiently recreating the missing records from the available noisy records of the log. In essence, recreating missing records amounts to generating complete trajectories (i.e., records available at all time steps) of the system from the given log. These recreated trajectories are regarded as a valid recreation of the behavior of the system from the given log. As the available records of the given log are noisy (i.e., not reduced to a point), there could be an infinite number of potential trajectories that can be recreated from the available records of the given log. That is, any trajectory of the system that passes through (i.e., intersects) all the available noisy records of the log, is considered a valid recreation of the behavior of the system from the given log. We refer to such a trajectory as a valid sample trajectory (or simply a valid trajectory) w.r.t. the given log (formally defined in Definition 3). Recreating all possible valid trajectories from a given noisy log usually requires reachability analysis, which often results in conservative estimates, suffers from scalability problems, and may be completely unavailable for complex systems. In contrast, our method overcomes this issue by trading-off formal guarantees with statistical guarantees, and thus only requiring to check a finite number of valid sample trajectories generated from the log. To do so, the user configures the proposed monitoring approach by specifying the desired confidence level for the analysis. This is typically chosen to be high, e.g., $c = 0.99$, where c stands for the desired confidence. Based on the required pre-set user confidence, our method computes the value of K, which is the number of random valid trajectories (recreated from the records in the log) that must be checked for safety in order to infer the system to be safe with the required confidence c. It is worth noting, in our approach, to generate a valid trajectory (w.r.t. to the given log), we simply proceed to generate a random trajectory of the system (from the given initial set), and then check if it intersects with all the records in the given log (see Fig. 1). If the trajectory intersects with every record of the log, we classify it as one of the valid trajectories among the K valid trajectories to be checked for safety. Otherwise, we discard it and generate a new trajectory. While checking K random valid trajectories for safety, if a trajectory is found to be violating the safety condition, then it is returned as concrete counterexample, and the system is inferred as unsafe. If all the K random valid trajectories are safe, our method guarantees the system is safe with the required the probabilistic confidence c. An overview of the propsoed approach is given in Fig. 1.

Contributions The main contributions of this work are:

1. An SHT (statistical hypothesis testing)-based framework, using Jefferies Bayes factor, for monitoring (proposed in Section 4).
2. A monitoring algorithm—that leverages the SHT-based framework—to detect safety violations from a given input log with noisy and missing records (proposed in Section 5).
3. We implemented our method as prototype tool, named `Posto`. Further, using this tool, we evaluated the proposed algorithm on three case studies, with two non-linear models and one DNN-based model. Our tool successfully monitored all the systems under 250 seconds.

Outline Section 2 reviews related works. Section 3 recalls necessary preliminaries. Sections 4 and 5 introduce our approach, and Section 6 discusses implementation and experiments. Section 7 outlines future works.

2 Related Work

Deterministic Monitoring Signal Temporal Logic (STL) formalizes real-valued signal properties in dense-time contexts. In practical scenarios with continuous dynamics and numerical parameters, simple yes/no answers are insufficient, requiring quantitative satisfaction details for informed decision-making. Existing methods like [14, 18] handle complex safety specifications but often need formal models or direct signal access, which can be overly complex or unavailable. In addition to STL-based monitoring, recent attention has shifted towards monitoring using automata-based specifications. Notably, advancements have been made in the study of complex, quantitative extensions of automata. Techniques such as timed pattern matching on timed regular expressions [6, 29] have emerged, particularly in the context of deterministic offline monitoring. The concept of *model-bounded monitoring*, introduced in [30], deviates from the conventional approach of monitoring a black-box system solely against a specification. Instead, it incorporates a limited, over-approximated understanding of the system to mitigate false positives. The over-approximated knowledge takes the form of a *linear hybrid automaton*. Model-bounded monitoring was extended in the framework of uncertain linear systems in [15]. Unlike all these deterministic approaches, our proposed monitoring approach does not require any formal model—and can handle complex models such as non-linear models, DNNs etc.—of the system. It trades off formal safety guarantees with high confidence probabilistic guarantees.

Statistical Verification Statistical verification has found extensive application across various domains, such as [12, 19, 28, 34] (see a survey in [23]). The method proposed in [35] uses statistical model checking which relies on Clopper-Pearson confidence levels (see [27]). This approach is used for verification of sample specifications in a Neural Network-based controller, captured through STL formulas. The work in [13] also uses the *Jeffries Bayes factor* test but tackles a fundamentally different problem. While they analyze falsifying traces in cyber-physical systems to identify input regions responsible for property violations, we apply the

same statistical method for monitoring complex systems with noisy and missing logs. Our goal is to detect safety violations, rather than characterize counterexample neighborhoods. Further, privacy implications of statistical model checking algorithms through the lens of differential privacy has also been explored [31]. Specifically, it focuses on sequential algorithms that draw samples until a specified condition is satisfied. It shows that disclosing the number of samples drawn can compromise privacy and highlights the inadequacy of the standard exponential mechanism in achieving differential privacy for sequential algorithms. Statistical methods have also been applied to learn simpler models of systems, that can offer probabilistic guarantees, using a limited number of samples. These models, known as *Probabilistic Approximately Correct (PAC)* models, have been used in various works, such as [4,11,26]. One commonly used approach for learning PAC models is through *scenario optimization* [10]. Additionally, techniques based on scenario optimization have been employed to identify safe inputs for black box systems, as discussed in [32,33].

3 Background and Problem Statement

The autonomous system under consideration evolves in discrete time in $\mathbb{R}^n$, called state space, where n is the dimension of the system. A state is $x \in \mathbb{R}^n$. It is worth noting that a formal model of the system is not explicitly required to apply our proposed monitoring approach. Essentially, all that is needed is a knowledge of the I/O execution described by $f_{sys}(x_0, t, [\boldsymbol{\epsilon}]_{t-1}) = x_t$, where $x_0 \in \mathbb{R}^n$ represents the initial state, $x_t \in \mathbb{R}^n$ represents the system's state at time t, and $[\boldsymbol{\epsilon}]_{t-1} = \{\epsilon_0, \epsilon_1, \ldots, \epsilon_{t-1}\}$ is a set of environmental inputs (i.e., disturbances experienced) to the system up to time t where $\forall_i \epsilon_i \in \mathbb{R}^n$. In simpler terms, $f_{sys}(x_0, t, [\boldsymbol{\epsilon}]_{t-1})$ represents the behavior of the system from an initial state x_0 over time t to yield the state x_t, with environmental inputs $[\boldsymbol{\epsilon}]_{t-1}$. The benefit of this formulation lies in its agnosticism towards the specific details of the system's internal workings. The proposed monitoring approach operates effectively as long as the output state x_t can be generated from the given input parameters $(x_0, t, [\boldsymbol{\epsilon}]_{t-1})$. No constraints are imposed on the nature of the discrete-time system model; it can be non-linear, utilize DNNs, or exhibit various other characteristics. This flexibility in accommodating diverse system models underscores the versatility of the proposed method. In other words, this allows for our method to be applied on a wide range of autonomous systems, making it a suitable approach for handling complex and diverse scenarios where detailed knowledge of the internal system dynamics may not be readily accessible. In our experiments, we evaluated our proposed approach on both non-linear models and a model employing a DNN-based controller, all of which was successfully monitored under 250 s.

Definition 1 (Reachable Set). *We overload the function $f_{sys}(\cdot)$ to take set values as follows. Given an initial set θ_0, and environmental inputs $[\mathcal{O}]_{t-1} = \{o_1, o_2, \cdots, o_{t-1}\}$, the reachable set at time step t is given as θ_t. Formally, the*

I/O execution of the system is given as follows: $f_{sys}(\theta_0, t, [\mathcal{O}]_{t-1}) = \theta_t$*; where* $\theta_0, \theta_t \subset \mathbb{R}^n$*, and* $\forall_{i\in[0,t-1]} o_i \subset \mathbb{R}^n$*.*

Definition 2 (Log). *Given a system I/O execution model as in Definition 1, a* finite size (uncertain) log *of the system is defined as follows:* $\ell = \{(\hat{\theta}_t, t) \mid \theta_t \subseteq \hat{\theta}_t, t \leq H\}$*, where* $H \in \mathbb{N}$ *is a given time bound.*

Properties and notations for logs Each tuple $(\hat{\theta}_t, t)$ is called a *record*. It is worth noting that the records are not necessarily reduced to a *point*. The size of the log ℓ, indicating the number of records it contains, is denoted by $|\ell|$. In a log ℓ, the k-th record is denoted as $\ell_k = (\hat{\theta}_{t_k}, t_k)$, where $\hat{\theta}_{t_k}$ over-approximates the system's state at time step t_k. It is worth highlighting that the size of a log may not necessarily be equal to H, but rather, $|\ell| \leq H$. In other words, the logs are considered *scattered*, implying that they may not contain a record for every t in the range of $1, \ldots, H$. We assume that each record of the log contains the true state of the system at a given time step, i.e., it is an over-approximation of the system's state. In practical scenarios, this assumption is generally valid. Physical sensors, as employed in applications like medical devices and cars, record values within an acceptable margin of error, providing a range of values that encompass the actual state.

Definition 3 (Valid Trajectory). *A trajectory* τ *of the system is an ordered sequence of states given as follows:* $\tau = \{x_0, x_1, \cdots, x_H\}$*; where* $\forall_{t\in[0,H]}$ $f_{sys}(x_0, t, [\boldsymbol{\epsilon}]_{t-1}) = x_t$*. A trajectory* $\tau = \{x_0, x_1, \cdots, x_H\}$ *is said to be* valid w.r.t. a given log $\ell = \{(\hat{\theta}_t, t) \mid \theta_t \subseteq \hat{\theta}_t, t \leq H\}$ *(of the same system) if:* $\forall_{(\hat{\theta}_t, t)\in\ell}\ x_t \in \hat{\theta}_t$.

In other words, a valid trajectory is a valid "recreation" of the behavior of the system from the given uncertain log.

Definition 4 (Random Trajectory). *Given a log* ℓ*, let* $\bar{\tau}_{val}$ *denote the set of all valid trajectories of the system w.r.t.* ℓ*, with environmental inputs* $[\mathcal{O}]_H$ *(see Definition 3). We assume that the trajectories* $\tau \in \bar{\tau}_{val}$ *are distributed according to a given probability distribution* $\mathcal{D}$*. Let a trajectory* τ *be randomly chosen from the set of all valid trajectories* $\bar{\tau}_{val}$ *(w.r.t. to* ℓ *and environmental inputs* $[\mathcal{O}]_H$*). This is randomly drawn according to the distribution* $\mathcal{D}$*, and formally expressed as* $\tau = \texttt{Sample}(f_{sys}(\cdot), \ell, [\mathcal{O}]_H, \mathcal{D})$.

For all our experiments, we assume $\mathcal{D}$ to be a uniform distribution. However, our method is compatible with any distribution $\mathcal{D}$ provided that an implementation of $\texttt{Sample}(\cdot)$ is available.

We aim here at performing safety monitoring of systems, from an uncertain log, utilizing an I/O execution model.

Problem 1. **Inputs**:

1. the system I/O execution model $f_{sys}(\cdot)$;

2. an uncertain log $\ell = \{(\hat{\theta}_t, t) \mid \theta_t \subseteq \hat{\theta}_t, t \leq H\}$;
3. environmental inputs $[\mathcal{O}]_H$;
4. the probability distribution $\mathcal{D}$ followed by the set of valid trajectories;
5. an unsafe set $\mathcal{U}$; and
6. a confidence $c \in (0, 1)$ desired on the required safety verification.

Problem: perform monitoring to ensure safety of the system with confidence c as defined by Jefferies Bayes factor based SHT.

4 SHT for Probabilistic Monitoring

In this section, we outline the statistical framework using Jeffries Bayes factor for hypothesis testing. The goal is to derive the value of K, representing the number of valid trajectories. These trajectories must be examined for safety to confirm that the system meets the required guarantee c. Let us first provide a brief overview of SHT. Our SHT framework formulates two hypotheses: the null hypothesis H_0 and the alternate hypothesis H_1. These hypotheses aim to determine if the system violated the safety condition based on the provided log. The null hypothesis H_0 asserts that a randomly chosen valid trajectory is safe (i.e., it does not intersect with the unsafe set), with a probability of *at most* c. On the other hand, the alternate hypothesis H_1 asserts that a randomly chosen valid trajectory is safe with a probability of *at least* c. After formulating the hypotheses, we collect evidence through random sampling of a finite number of valid trajectories. The algorithm concludes when enough evidence is gathered to validate either hypothesis (H_0 or H_1). The supported hypothesis, determined by observed random samples, is accepted, while the opposing one is rejected. Subsequently, we employ a Bayesian hypothesis test to make a decision between these two hypotheses [21]. An important consequence of our test is, when the samples we have drawn do not support the alternate hypothesis, they will contain a *counterexample* (also a valid trajectory) that violates safety. Within our statistical hypothesis framework, we not only incorporate the probabilistic guarantee c but also set bounds for the so called type I and type II errors. The type I error denotes the likelihood of incorrectly concluding the alternate hypothesis when, in reality, the null hypothesis is true. Similarly, the type II error denotes the probability of incorrectly concluding the null hypothesis when in reality the alternate hypothesis holds. We choose the relevant parameters such that the type I error is kept significantly low (as desired). Type II error, which refers to the probability of incorrectly selecting the null hypothesis when the alternative hypothesis is true, is not relevant in our setting. This is because we never select the null hypothesis, which would claim the system is safe with a probability less than c. Instead, we only conclude that the system is probabilistically safe when the alternative hypothesis is selected, which asserts that the system is safe with probability is greater than or equal to c. If the alternative hypothesis is rejected, we provide a concrete counterexample. Thus, type II error does not apply in this context. Next, we describe how the hypothesis test is carried for a given

value of $c \in (0, 1)$. We use Bayesian hypothesis testing based on Jefferies Bayes factor [13]. Accordingly, we first formulate the null and alternate hypotheses:

$$H_0 : \mathcal{P}rob\left[f_{sys}(\cdot), \ell, \mathcal{D}, \mathcal{U}\right] < c, \quad (1) \qquad H_1 : \mathcal{P}rob\left[f_{sys}(\cdot), \ell, \mathcal{D}, \mathcal{U}\right] \geq c. \quad (2)$$

Here, $\mathcal{P}rob\left[f_{sys}(\cdot), \ell, \mathcal{D}, \mathcal{U}\right]$ denotes the probability that a randomly drawn valid trajectory is safe. In other words, it represents the probability that a given random trajectory τ (of the system $f_{sys}(\cdot)$), that is valid w.r.t. the log ℓ does not intersect with the unsafe set $\mathcal{U}$. Our goal is to determine if the alternate hypothesis is accepted with the given parameters. We begin by setting a sufficiently high value (say 10^5) for the *Bayes factor* B, a term we will define shortly [16]. Using this Bayes factor B and the probability c, we compute K, the number of valid trajectories needed to decide between the null and alternate hypotheses. Next, we draw K samples $X = \{\tau_1,\ \tau_2, \ldots,\ \tau_K\}$ according to the given distribution $\mathcal{D}$ over the set of all valid trajectories. That is, for each i, τ_i is randomly selected as follows: $\tau_i = \texttt{Sample}(f_{sys}(\cdot), \ell, [\mathcal{O}]_H, \mathcal{D})$. Then, we check if *each* member of X does not intersect with the unsafe set $\mathcal{U}$, i.e., $\forall_i \tau_i \cap \mathcal{U} = \emptyset$. If this is true for every trajectory, we accept the alternate hypothesis, concluding that the system is (probabilistically) safe. If not—there is at least one trajectory that violates safety—we consider the first such trajectory as a counterexample, i.e., $\tau_j \cap \mathcal{U} \neq \emptyset$. This counterexample is then returned as a result, leading to the conclusion that the system is unsafe.

Now, we will present the connection between the Bayes factor B, confidence c, and the number of valid trajectories to be generated, denoted as K. Consider $X = \{\tau_1, \tau_2, \ldots, \tau_K\}$ as a set of randomly selected valid trajectories, all of which are deemed *safe*. The probability that all these trajectories are safe, assuming the null hypothesis, is expressed as (similar to the treatment in [13]):

$$\mathbf{Pr}\left[\forall \tau \in X : \tau \cap \mathcal{U} = \emptyset \mid H_0\right] = \int_0^c q^K \, dq. \quad (3)$$

Similarly, the probability that all the trajectories are safe, given the alternate hypothesis, is expressed as:

$$\mathbf{Pr}\left[\forall \tau \in X : \tau \cap \mathcal{U} = \emptyset \mid H_1\right] = \int_c^1 q^K \, dq. \quad (4)$$

The *Bayes factor* is the ratio of the above two probabilities:

$$\frac{\mathbf{Pr}\left[\forall \tau \in X : \tau \cap \mathcal{U} = \emptyset \mid H_1\right]}{\mathbf{Pr}\left[\forall \tau \in X : \tau \cap \mathcal{U} = \emptyset \mid H_0\right]} = \frac{1 - c^{K+1}}{c^{K+1}} \quad (5)$$

The Bayes factor serves as a measure of the strength of evidence supporting the alternate hypothesis in comparison to the null hypothesis. In Jeffries Bayes factor test, we ensure that the ratio calculated in Eq. (5) exceeds the Bayes

Algorithm 1: Proposed Statistical Monitoring

input : $f_{sys}(\cdot)$, ℓ, $[\mathcal{O}]_H$, $\mathcal{D}$, $\mathcal{U}$, c
output : Return *probabilistic-safe* (resp. *unsafe*) if the system behavior is probabilistically safe (resp. unsafe with a counterexample).

/* $f_{sys}()$: System input/output execution (see Definition 1); */
/* ℓ: Uncertain log (see Definition 2); */
/* $[\mathcal{O}]_H$: Environmental inputs experienced by the system up to time H; */
/* $\mathcal{D}$: Probability distribution followed by the set of all valid trajectories (see Definition 4); */
/* $\mathcal{U}$: Unsafe set; */
/* c: Required probabilistic confidence (see Section 4); */

1 $K \leftarrow$ Compute using Eq. (6) with given c;
2 **for** $k \in [1, K]$ **do**
3 $\quad \tau \leftarrow \texttt{Sample}(f_{sys}(\cdot), \ell, [\mathcal{O}]_H, \mathcal{D})$; // generate a random valid trajectory
4 $\quad$ **if** $\tau \cap \mathcal{U} \neq \emptyset$ // check if trajectory is unsafe
5 $\quad$ **then**
6 $\quad\quad$ **return** *(unsafe,*τ*)*;
7 **return** *probabilistic-safe*;

factor B specified by the user. Hence, a sufficiently high Bayes factor value indicates that the evidence favors the alternate hypothesis over the null hypothesis. We note that, for the sake of clarity in presentation, we do not treat it as an input to our method, but rather assume that it is preset by the user. However, Bayes factor is indeed a parameter chosen by the user. With B given, we can now compute the required number of samples K in the following manner:

$$\frac{1 - c^{K+1}}{c^{K+1}} > B \iff K > -\log_c(B+1) \tag{6}$$

We conclude this subsection by bounding type I error, i.e., where the alternate hypothesis accepted but in reality the null hypothesis is true. According to [13], the type I error rate is bounded by: $err(B, c) = c/(c + (1 - c)B)$. The detailed steps of our monitoring procedure are discussed in Section 5. We note that while, in theory, one can explore other hypothesis testing methods (including sequential hypothesis testing), this framework (using Jeffries Bayes factor) offers several advantages [16], notably: 1) ability to work with any distribution $\mathcal{D}$; and 2) the value of K can be pre-computed and is independent of the sample size, etc.

5 Methodology

The structure of our algorithm is in Fig. 1. The formalization of the main algorithm is presented in Algorithm 1. Let us discuss its primary components:

Computing K. We use the formula specified in Eq. (6) to calculate the value of K based on the provided c in line 1.

Checking safety of K random valid trajectories. Within the `for` loop in line 2, we generate K random valid trajectories and check its safety.

Generate a random valid trajectory. In line 3, a random valid trajectory $\tau = \{x_0, x_1, \cdots, x_H\}$ is generated according to the distribution $\mathcal{D}$ (see Definition 3).

Checking safety of τ. In line 5, we check if the trajectory τ intersects with the unsafe set $\mathcal{U}$, i.e., $\exists_t x_t \cap \mathcal{U} \neq \emptyset$. If there exists such a t at which the trajectory τ intersects with unsafe set $\mathcal{U}$, we return the trajectory τ as a counterexample, and infer the system as unsafe, in line 6. If none of the K generated random valid trajectories are unsafe, we infer the system as probabilistically safe (with the desired confidence c) in line 7.

Theorem 1 (Probabilistic Soundness of Algorithm 1). *If Algorithm 1 infers the system as probabilistic-safe (in line 7), then the system is safe with the required probabilistic confidence c.*

The proof of Theorem 1 follows from the hypothesis testing framework discussed in Section 4. While our method (Algorithm 1) is compatible with any distribution $\mathcal{D}$ provided that the implementation of `Sample`$(\cdot)$ is available, we specifically discuss a method to implement `Sample`$(\cdot)$ tailored for a uniform distribution, named **Uniform Sampling**. This adaptation is made because, for our experiments, we assume that $\mathcal{D}$ is a uniform distribution. This is incorporated into Fig. 1, providing an integrated overview of the proposed monitoring approach (see also Algorithm 1). Given a log $\ell = \{(\hat{\theta}_t, t)\}$ and environmental inputs $[\mathcal{O}]_H = \{o_1, o_2, \cdots, o_H\}$, where $\forall_{t\in[0,H]}\ \hat{\theta}_t, o_t \subset \mathbb{R}^n$, the **Uniform Sampling** algorithm proceeds as follows:

Selecting the initial set. The initial log record, denoted as $\ell_0 = (\hat{\theta}_0, t = 0)$, is considered to be the starting point for monitoring, assuming this occurs at time step $t = 0$. Consequently, $\hat{\theta}_0$ is presumed to be the over-approximated initial set of the system, serving as the starting point for monitoring.

Generating a random trajectory. The first step in generating a random valid trajectory, is to generate a trajectory of the system (not necessarily valid yet) from the given initial set $\hat{\theta}_0$ and environmental inputs $[\mathcal{O}]_H$. To do so, we first sample a state $x_0 (\in \mathbb{R}^n)$ uniformly at random from the initial set $\hat{\theta}_0$. Next, we sample the environmental inputs $[\boldsymbol{\epsilon}]_H = \{\epsilon_0, \epsilon_1, \cdots, \epsilon_H\}$, such that, for all $t \in [0, H]$ $\epsilon_t (\in \mathbb{R}^n)$ is drawn uniformly at random from the set o_t. Once the initial state x_0 and the environmental inputs $[\boldsymbol{\epsilon}]$ are sampled, we compute the trajectory as follows: $\tau = \{x_0, x_1, \cdots, x_H\}$, where $\forall_{t\in[0,H]}\ x_t = f_{sys}(x_0, [\boldsymbol{\epsilon}]_{t-1}, t)$.

Check if τ is valid. A trajectory is considered valid when it meets the conditions outlined in Definition 3. In order for τ to be valid, all the records in ℓ must intersect with the points in τ at their respective time steps. Encountering a record in ℓ that fails to intersect with τ at a given time step implies that the trajectory τ was not observed during the real execution of the system.

Discussion Our technique is applicable whenever the I/O execution of a system is available, which is a reasonable assumption in most cases. When designing

autonomous systems, models are often created within a framework or simulations, which are also used for testing and performance evaluation. Additionally, the bounds on the initial set and environmental inputs, along with their corresponding probabilistic distributions, are typically known. In practice, these distributions are usually uniform or concentrated around a central value. This makes it reasonable to assume that the I/O execution model and the probabilistic distributions of trajectories are known in most real-world scenarios.

6 Case Studies

We illustrate the effectiveness and practicality of our method through its application to three benchmarks. The first two benchmarks involve non-linear models with added environmental uncertainties (namely jet model and van der Pol Oscillator discussed in Sections 6.1 and 6.2 respectively), while the third benchmark incorporates a DNN-based controller (namely mountain car discussed in Section 6.2). We implemented the proposed statistical monitoring approach (Algorithm 1) as a prototype tool developed in Python, named `Posto`[4,5].

Input/Output Execution Model We recall that our monitoring approach requires only the I/O execution of the system, which is capable of generating output trajectories from the inputs (and not a detailed formal model). In our experiments, we simulated such an execution behavior using the known system formal models (without the need to explicitly provide them as input to our monitoring approach) while incorporating additional uncertainties.

Generating Logs We simulate a trajectory according to **Uniform Sampling**, by randomly selecting an initial state from the initial set and a set of environmental inputs. Following the generation of this trajectory, we simulate logging on it, as per specified probability p. In essence, at each time step along the trajectory, we gather a record with a probability of p. After collecting these records, we introduce random uncertainty δ_{log} to each record, replicating sensor noise in the system. In the end, we get an uncertain log as in Definition 2.

Implementation details Next, we highlight certain implementation details of our proposed statistical monitoring approach (Algorithm 1): *i*) We set the value of $B = 10^5$ as a default in our implementation. Unless otherwise stated, for all our experiments, we use $c = 0.99$. *ii*) Since we assume a uniform distribution of valid trajectories (Definition 4), we implemented the `Sample`$(\cdot)$ method (in line 3 of Algorithm 1) according to **Uniform Sampling**. However, our proposed approach can work with any distribution.

[4]The code, binaries, models, and scripts for stochastic result recreation are open-sourced on GitHub https://github.com/bineet-coderep/posto.

[5]Experiments were performed on a Lenovo ThinkPad with i7-8750H CPU with 2.20 GHz and 32 GiB memory on Ubuntu 20.04 LTS (64 bit).

Research Questions (RQ) We consider the following research questions in our case studies:

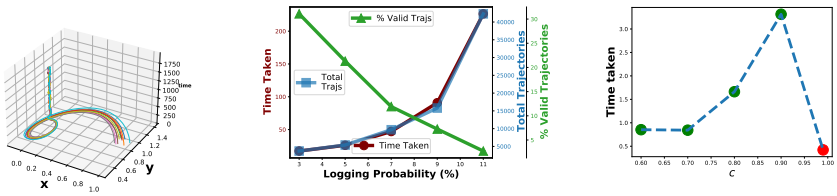
1. The impact of varying the logging probability p (i.e., the number of records in the log) on the frequency of discovering a counterexample or inferring the system as probabilistically safe.
2. The impact of varying the amount of uncertainties δ_{log} in the log records (i.e., the volume of records in the log) on the frequency of discovering a counterexample or inferring the system as probabilistically safe.
3. Impact of varying the probabilistic confidence c.
4. In the **Uniform Sampling** algorithm, the effect of various parameters (logging probability and the amount of uncertainty present in the log records) on the chances of identifying valid trajectories.

6.1 Jet Model

System Description The dynamics of the jet model we employed to simulate the I/O execution corresponds with a Moore-Greitzer model of a jet engine with a stabilizing feedback control while operating in the no-stall mode [5]. We discretized the system with additional environmental uncertainties (inputs), ϵ_x, ϵ_y, on states x and y respectively. In this model, the origin is translated to a desired no-stall equilibrium. The state variables correspond to $x = \mathcal{X} - 1$, $y = \mathcal{Y} - \mathcal{Y}_{co} - 2$, where $\mathcal{X}$ is the mass flow, $\mathcal{Y}$ is the pressure rise and $\mathcal{Y}_{co}$ is a constant.

Safety The system is considered safe if the mass flow of the jet model is maintained at all time steps as follows: $x \geq -0.10$. Imagine a crash scenario where the only available data for analysis is the log recorded by the jet prior to the crash. Conducting an analysis with our monitoring algorithm can be beneficial for investigating agencies, manufacturers, and others. It enables the detection of safety condition violations during the jet's execution and the identification of trajectories that breached safety conditions.

Experiments The initial set and the environmental inputs considered for this example are as follows: $x \in [0.8, 1], y \in [0.8, 1], \epsilon_x, \epsilon_y \in [0, 0.002]$ (see [1] for initial sets). Some random trajectories of the system are shown in Fig. 2a. In Fig. 3, the plots in the bottom row (Figs. 3c and 3d) and upper row (Figs. 3a and 3b) have logging probabilities (p) of 3% and 5% respectively. Additionally, the logs in the left column (Figs. 3a and 3c) and right column (Figs. 3b and 3d) have log uncertainty (δ_{log}) values of 0.02 and 0.04 respectively. We also conduct an analysis of our monitoring algorithm using logging probabilities (p) set at 7%, 9%, and 11%, each with a log uncertainty value of $\delta_{log} = 0.02$. Due to the similarity of the plots to those in Fig. 3, we have omitted them from the paper. For Figs. 2b and 2c, log uncertainty was set to $\delta_{log} = 0.02$. We now answer RQs (1)-(4), using Figs. 2b, 2c and 3.

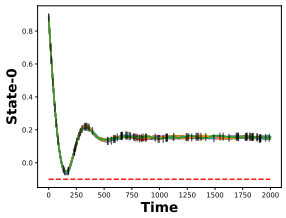


(a) **Random Trajectories** (b) **Generation of Valid Trajectories** (c) **Probabilistic confidence** c

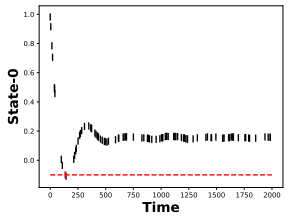
Fig. 2: **Jet Model Case Study.** *Random Trajectories (Fig. 2a).* x- and y- axis represents state variables, and z-axis represents time step. *Impact of Logging Probability on Generation of Valid Trajectories (Fig. 2b).* We illustrate how varying the logging probability affects the monitoring execution time (Algorithm 1) and the percentage of valid trajectories out of the total number of generated trajectories. *Impact of Probabilistic confidence c (Fig. 2c).* We illustrate the impact of logging probability on the time taken to perform monitoring (Algorithm 1) and safety inferences. The red dot denotes an instance where the system was inferred to be unsafe with a counterexample, and green dots denote instances where the system was inferred to be probabilistically safe (with the corresponding confidence value indicated in the x-axis.

Answer to RQ 1 (impact of logging probability p) We address this by comparing two sets of figures in Fig. 3. In the left column, with a smaller record size, Fig. 3c took 17.26 s to conclude that the system is probabilistically safe. Increasing the probability of logging (i.e., incorporating more records) as shown in Fig. 3a, also inferred the system to be probabilistically safe. However, the inference of safety took longer, requiring 26.07 s. This increase in monitoring time will be discussed in detail in RQ 4. In the right column, with larger record size, the analysis in Fig. 3d concluded in 0.51 s, determining the system behavior as unsafe and generating a counterexample (dotted red trajectory). The behavior of the system in Fig. 3b, with a logging probability of 5%, resulted in inferring the system behavior as unsafe, as three records in the input log were unsafe. In this case, the main statistical monitoring process did not initiate; instead, it concluded by merely verifying the safety of the logs, and this process took 0.006 s. We additionally assess the influence of logging probability through Fig. 2b. We further incrementally increase the logging probability to 7%, 9%, and 11%. In each of these cases, although the conclusion about the system being probabilistically safe remained consistent, the duration for performing the monitoring varied: 47.5 s, 90.88 s, and 226.43 s, respectively. Fig. 2b further illustrates how the time required for monitoring increases quite noticeably with the increase in logging probability.

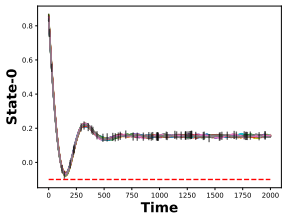
Answer to RQ 2 (impact of log uncertainty δ_{log}) We address this question by comparing two sets of figures in Fig. 3. In the bottom row, with a smaller logging probability: Increasing the uncertainty in the log leads to a change in the inference from safe (Fig. 3c) to unsafe with a counterexample (Fig. 3d). Similarly, in the top row (with higher logging probability): Increasing the uncertainty in the log results in rendering some of the log records unsafe, leading to the system being inferred as unsafe (Fig. 3b).



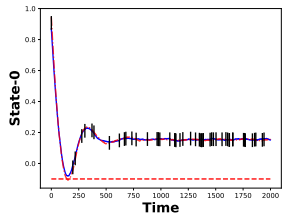
(a) Frequent records, low uncertainty



(b) Frequent records, high uncertainty



(c) Sporadic records, low uncertainty



(d) Sporadic records, high uncertainty

Fig. 3: **Statistical Monitoring.** Evolution of the state—x for the jet model—with time. The volume of the records increases from left to right, and the probability of logging increases from bottom to top. The colored trajectories are the generated random valid trajectories during the process of monitoring, the dotted trajectory in red is an unsafe trajectory discovered during the process of monitoring (see Fig. 3d), the black regions are records in the log given as an input to the monitoring algorithm, and the dark-brown regions are unsafe-records in the log (Fig. 3b). The red dotted line represents the safe range for the variable.

Answer to RQ 3 (impact of probabilistic confidence c) We address this question by using Fig. 2c. By varying the probabilistic confidence c through $0.6, 0.7, 0.8, 0.9$ and 0.99, we witness a drastic increase in the time taken by the monitoring approach: 0.85 s, 0.85 s, 1.66 s, 3.31 s and 0.42 s respectively. This is natural, as the value K (i.e., number of valid trajectories) required to infer the system as safe, with the respective confidence c, also increases drastically as per Eq. (6). Also, note that increasing the value c increases the chances of finding a counterexample that violates safety. The plot illustrates this trend, demonstrating that although no counterexamples were discovered at c values of 0.6, 0.7, and 0.8, a counterexample violating the system's safety was discovered when $c = 0.99$.

Answer to RQ 4 (chances of finding valid trajectories) We investigate this question by using Fig. 2b. When we increase the logging probability (p), even with the same level of confidence (c), it takes more time to perform monitoring. To put it simply, even if the number of valid trajectories needed to infer the system's safety (as defined by c) stays the same, increasing the number of records in the log (due to an increase in the value of p) results in a longer monitoring time. This behavior results from the implementation, which involves generating random valid trajectories based on a uniform distribution, as outlined in **Uniform Sampling**. To elaborate, when we increase the logging probability p by increments of 3%, 5%, 7%, 9%, and 11%, the proportion of valid trajectories

among all randomly generated trajectories is observed to be 31.08%, 21.3%, 11.9%, 7.26%, and 2.71%, respectively.

6.2 Other Benchmarks and General Observations

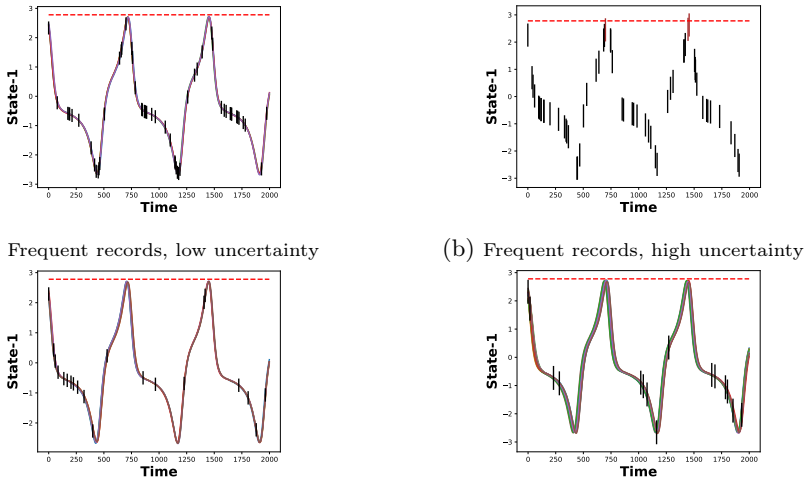
Let us briefly discuss the other two case studies (namely van der Pol and mountain car). Given the similar trends observed in these two case studies with respect to RQs (1)-(4), and further owing to space limitations, we cumulate the observations from all the experiments and discuss them in detail here as general observations. The details of the mountain car model are discussed next, while the van der Pol oscillator model is presented in Section 6.2, due to its similarity to the previously discussed jet model (Section 6.1) and space limitations.

Mountain Car We now evaluate our proposed approach on a mountain car model with a DNN-based controller. Given the complexity of a DNN-based controller and the increasing deployment of such controllers in safety-critical scenarios, it is important to scalably investigate such systems in the event of a crash. This helps us to determine the cause of the crash and identify the trajectory that violated safety by analyzing the system's log. Thus, making this model an ideal candidate to evaluate our monitoring approach.

System Description We consider a scenario where an under-powered car needs to climb a steep hill. Because the car lacks sufficient power to directly go up the hill, it has to drive up the hill in the opposite direction first to build up enough momentum to reach its destination. The DNN-based controller $f_{DNN}(\cdot)$ uses the car's position (p) and velocity (v) as inputs and provide an acceleration command as output (refer to [17] for the discrete time dynamics).

Experiments The system is considered safe if the velocity of mountain car is maintained at all time steps as follows: $v \leq 0.055$. The initial set considered for this example are as follows: $p \in [-1.2, -1], v \in [-0.07, 0.07]$ (see [17]). To assess our monitoring approach in this case study, we conducted four distinct analyses, categorized into two sets (similar to the other case studies): one involving varying logging probabilities p (20% and 40%), and the other involving varying log uncertainties δ_{log} (0.004 and 0.008). As the observed plots closely resemble those depicted in Figs. 3 and 4, we have not included them here. We additionally conducted analysis using c values of 0.7, 0.7, 0.8, and 0.99, while keeping $\delta_{log} = 0.004$. We have not included the corresponding plot since it closely resembles Fig. 2c.

Van der Pol Oscillator In this case study, the dynamics of the van der Pol model (as in [2]) was discretized with additional environmental uncertainties ($\epsilon_x, \epsilon_y, \epsilon_\mu$) to simulate the I/O execution. Here, the state variables x and y represent position and velocity, respectively, while μ denotes the damping strength that remains constant in the system.



(a) Frequent records, low uncertainty

(b) Frequent records, high uncertainty

(c) Sporadic records, low uncertainty

(d) Sporadic records, high uncertainty

Fig. 4: **Van der Pol Oscillator Monitoring.** Evolution of the state—y for the Van der Pol model—with time. The volume of the records increases from left to right, and the probability of logging increases from bottom to top. The colored trajectories are the generated random valid trajectories during the process of monitoring, the dotted trajectory in red is an unsafe trajectory discovered during the process of monitoring (see Fig. 4d), the black regions are records in the log given as an input to the monitoring algorithm, and the dark-brown regions are unsafe-records in the log (Fig. 4b). The red dotted line represents the safe range for the variable.

Experiments The system is considered safe if the velocity is maintained at all time steps as follows: $y \leq 2.78$. The initial set and the environmental inputs considered for this example are as follows: $x \in [1.25, 1.45], y \in [2.25, 2.35], \epsilon_x, \epsilon_y \in [0, 0.004]$, $\epsilon_\mu \in [0, 0.01]$ (see [1, 2] for initial sets). The illustration of random trajectories of this system is omitted as it is similar in essence to that of the jet model. In Fig. 4, the plots in the bottom row (Figs. 4c and 4d) and upper row (Figs. 4a and 4b) have logging probabilities (p) of 1% and 3% respectively. Additionally, the logs in the left column (Figs. 4a and 4c) and right column (Figs. 4b and 4d) have log uncertainty (δ_{log}) values of 0.2 and 0.4 respectively. We additionally conducted analysis using c values of 0.6, 0.7, 0.8, and 0.99, while keeping $\delta_{log} = 0.2$. We omit the plot as it similar to Fig. 2c.

General Observations

Answer to RQ 1 As evidenced across nearly all three benchmarks (with the exception of a single sub case in van der Pol), increasing the logging probability shows no significant effect on the inference of system's safety by the proposed monitoring approach. However, there is a noticeable increase in the computational time required for the monitoring algorithm as the logging probability increases. This can primarily be attributed to the decreasing chances of discov-

ering valid trajectories with higher logging probabilities, which will be elaborated in RQ 4.

Answer to RQ 2 As observed in the first two benchmarks, increasing the uncertainty in the log, increases the chances of discovering a counterexample (while the third benchmark remained probabilistically safe in all cases). With an increase in log uncertainty, we observed noticeable decrease in monitoring time in some sub-cases. This could again be attributed to increasing chances of finding valid trajectories with an increase in log uncertainties. See RQ 4.

Answer to RQ 3 Increasing the probabilistic confidence increases the chances of finding a counterexample that violates safety, and also the compute time by the monitoring algorithm increases. This outcome is to be expected, given that the value of K (i.e., number of valid trajectories) required to infer the system as safe, with the respective confidence c, also increases drastically as per Eq. (6). Specifically, for c values of 0.6, 0.7, 0.8, 0.9, and 0.99, the computed values of K are 23, 33, 52, 110, and 1146, respectively.

Answer to RQ 4 As observed in several of the cases across the three benchmarks, the chances of finding valid trajectories increases with: *i*) decrease in logging probability p, and *ii*) increase in log uncertainty δ_{log}. This behavior arises from how random valid trajectories are generated in **Uniform Sampling**. Nevertheless, it is worth noting that across all our benchmarks (including all subcases), which includes nonlinear and DNN-based controllers, the monitoring was completed within 250 s on a standard laptop.

7 Conclusion and Perspectives

This work opts to trade-off formal safety guarantees with high-confidence probabilistic assurances to ensure trustworthy monitoring suitable for most practical applications. To this end, we propose an SHT-based framework for monitoring to detect safety violations from an input log with noisy and missing records. Our implementation into a prototype tool `Posto` showed that we can successfully monitor non-linear and DNN based benchmarks under 250 seconds. Given the scalability of our method and the observations it offer, this work paves the way for several future research directions. Our ultimate aim is to adopt a fully representation-free strategy, where the I/O execution representation will be co-designed with the monitoring approach. The goal of this synergy is to mutually enhance both the I/O execution representation and the monitoring process, to achieve a better scalability for complex models and are tuned towards finding counterexamples faster.

References

1. Jet Model, https://ths.rwth-aachen.de/research/projects/hypro/benchmarks-of-continuous-and-hybrid-systems/

2. Althoff, M., Stursberg, O., Buss, M.: Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In: CDC. pp. 4042–4048. IEEE (2008). https://doi.org/10.1109/CDC.2008.4738704
3. André, É., Coquard, E., Fribourg, L., Jerray, J., Lesens, D.: Parametric schedulability analysis of a launcher flight control system under reactivity constraints. FI **182**(1), 31–67 (Sep 2021). https://doi.org/10.3233/FI-2021-2065
4. Ashok, P., Kretínský, J., Weininger, M.: PAC statistical model checking for markov decision processes and stochastic games. In: Dillig, I., Tasiran, S. (eds.) CAV, Part I. LNCS, vol. 11561, pp. 497–519. Springer (2019). https://doi.org/10.1007/978-3-030-25540-4_29
5. Aylward, E.M., Parrilo, P.A., Slotine, J.J.E.: Stability and robustness analysis of nonlinear systems via contraction metrics and SOS programming. Automatica **44**(8), 2163–2170 (2008). https://doi.org/10.1016/J.AUTOMATICA.2007.12.012
6. Bakhirkin, A., Ferrère, T., Nickovic, D., Maler, O., Asarin, E.: Online timed pattern matching using automata. In: Jansen, D.N., Pavithra, P. (eds.) FORMATS. LNCS, vol. 11022, pp. 215–232. Springer (2018). https://doi.org/10.1007/978-3-030-00151-3_13
7. Bartocci, E., Deshmukh, J.V., Donzé, A., Fainekos, G.E., Maler, O., Nickovic, D., Sankaranarayanan, S.: Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications. In: Bartocci, E., Falcone, Y. (eds.) Lectures on Runtime Verification – Introductory and Advanced Topics, LNCS, vol. 10457, pp. 135–175. Springer (2018). https://doi.org/10.1007/978-3-319-75632-5_5
8. Basin, D.A., Klaedtke, F., Zalinescu, E.: The MonPoly monitoring tool. In: Reger, G., Havelund, K. (eds.) RV-CuBES. Kalpa Publications in Computing, vol. 3, pp. 19–28. EasyChair (2017)
9. Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., Hwang, L.J.: Symbolic model checking: 10^{20} states and beyond. Information and Computation **98**(2), 142–170 (1992). https://doi.org/10.1016/0890-5401(92)90017-A
10. Calafiore, G.C., Campi, M.C.: The scenario approach to robust control design. IEEE Transactions on Automatic Control **51**(5), 742–753 (2006). https://doi.org/10.1109/TAC.2006.875041
11. Chen, Y., Hsieh, C., Lengál, O., Lii, T., Tsai, M., Wang, B., Wang, F.: PAC learning-based verification and model synthesis. In: Dillon, L.K., Visser, W., Williams, L.A. (eds.) ICSE. pp. 714–724. ACM (2016). https://doi.org/10.1145/2884781.2884860
12. Clarke, E.M., Zuliani, P.: Statistical model checking for cyber-physical systems. In: Bultan, T., Hsiung, P.A. (eds.) ATVA. LNCS, vol. 6996, pp. 1–12. Springer (2011). https://doi.org/10.1007/978-3-642-24372-1_1
13. Diwakaran, R.D., Sankaranarayanan, S., Trivedi, A.: Analyzing neighborhoods of falsifying traces in cyber-physical systems. In: Martínez, S., Tovar, E., Gill, C., Sinopoli, B. (eds.) ICCPS. pp. 109–119. ACM (2017). https://doi.org/10.1145/3055004.3055029
14. Donzé, A., Ferrère, T., Maler, O.: Efficient robust monitoring for STL. In: Sharygina, N., Veith, H. (eds.) CAV. LNCS, vol. 8044, pp. 264–279. Springer (2013). https://doi.org/10.1007/978-3-642-39799-8_19
15. Ghosh, B., André, É.: Offline and online energy-efficient monitoring of scattered uncertain logs using a bounding model. LMCS **20**(1), 2:1–2:33 (Jan 2023). https://doi.org/10.46298/lmcs-20(1:2)2024

16. Ghosh, B., Hobbs, C., Xu, S., Duggirala, P.S., Anderson, J.H., Thiagarajan, P.S., Chakraborty, S.: Statistical hypothesis testing of controller implementations under timing uncertainties. In: RTCSA. pp. 11–20. IEEE (2022). https://doi.org/10.1109/RTCSA55878.2022.00008
17. Ivanov, R., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verisig: Verifying safety properties of hybrid systems with neural network controllers. In: Ozay, N., Prabhakar, P. (eds.) HSCC. pp. 169–178. ACM (2019). https://doi.org/10.1145/3302504.3311806
18. Jakšić, S., Bartocci, E., Grosu, R., Nguyen, T., Ničković, D.: Quantitative monitoring of STL with edit distance. FMSD **53**(1), 83–112 (2018). https://doi.org/10.1007/s10703-018-0319-x
19. Jha, S.K., Clarke, E.M., Langmead, C.J., Legay, A., Platzer, A., Zuliani, P.: A bayesian approach to model checking biological systems. In: Degano, P., Gorrieri, R. (eds.) CMSB. LNCS, vol. 5688, pp. 218–234. Springer (2009). https://doi.org/10.1007/978-3-642-03845-7_15
20. Kaivola, R., Ghughal, R., Narasimhan, N., Telfer, A., Whittemore, J., Pandav, S., Slobodová, A., Taylor, C., Frolov, V.A., Reeber, E., Naik, A.: Replacing testing with formal verification in Intel CoreTM i7 processor execution engine validation. In: Bouajjani, A., Maler, O. (eds.) CAV. LNCS, vol. 5643, pp. 414–429. Springer (2009). https://doi.org/10.1007/978-3-642-02658-4_32
21. Kass, R., Raftery, A.: Bayes factors. Journal of the American Statistical Association **90**(430), 773–795 (1995)
22. Larsen, K.G., Lorber, F., Nielsen, B.: 20 years of UPPAAL enabled industrial model-based validation and beyond. In: Margaria, T., Steffen, B. (eds.) ISoLA, Part IV. LNCS, vol. 11247, pp. 212–229. Springer (2018). https://doi.org/10.1007/978-3-030-03427-6_18
23. Legay, A., Delahaye, B., Bensalem, S.: Statistical model checking: An overview. In: Barringer, H., Falcone, Y., Finkbeiner, B., Havelund, K., Lee, I., Pace, G.J., Rosu, G., Sokolsky, O., Tillmann, N. (eds.) RV. LNCS, vol. 6418, pp. 122–135. Springer (2010). https://doi.org/10.1007/978-3-642-16612-9_11
24. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Lakhnech, Y., Yovine, S. (eds.) FORMATS and FTRTFT. LNCS, vol. 3253, pp. 152–166. Springer (2004). https://doi.org/10.1007/978-3-540-30206-3_12
25. Mamouras, K., Chattopadhyay, A., Wang, Z.: A compositional framework for quantitative online monitoring over continuous-time signals. In: Feng, L., Fisman, D. (eds.) RV. LNCS, vol. 12974, pp. 142–163. Springer (2021). https://doi.org/10.1007/978-3-030-88494-9_8
26. Park, S., Bastani, O., Matni, N., Lee, I.: PAC confidence sets for deep neural networks via calibrated prediction. In: ICLR. OpenReview.net (2020), https://openreview.net/forum?id=BJxVI04YvB
27. Roohi, N., Wang, Y., West, M., Dullerud, G.E., Viswanathan, M.: Statistical verification of the Toyota powertrain control verification benchmark. In: Frehse, G., Mitra, S. (eds.) HSCC. pp. 65–70. ACM (2017). https://doi.org/10.1145/3049797.3049804
28. Sen, K., Viswanathan, M., Agha, G.: On statistical model checking of stochastic systems. In: Etessami, K., Rajamani, S.K. (eds.) CAV. LNCS, vol. 3576, pp. 266–280. Springer (2005). https://doi.org/10.1007/11513988_26
29. Waga, M., Akazaki, T., Hasuo, I.: A Boyer-Moore type algorithm for timed pattern matching. In: Fränzle, M., Markey, N. (eds.) FORMATS. LNCS, vol. 9884, pp. 121–139. Springer (2016). https://doi.org/10.1007/978-3-319-44878-7_8

30. Waga, M., André, É., Hasuo, I.: Model-bounded monitoring of hybrid systems. ACM Transactions on Cyber-Physical Systems **6**(4), 30:1–30:26 (Nov 2022). https://doi.org/10.1145/3529095
31. Wang, Y., Sibai, H., Yen, M., Mitra, S., Dullerud, G.E.: Differentially private algorithms for statistical verification of cyber-physical systems. IEEE Open Journal of Control Systems **1**, 294–305 (2022). https://doi.org/10.1109/OJCSYS.2022.3207108
32. Xue, B., Liu, Y., Ma, L., Zhang, X., Sun, M., Xie, X.: Safe inputs approximation for black-box systems. In: Pang, J., Sun, J. (eds.) ICECCS. pp. 180–189. IEEE (2019). https://doi.org/10.1109/ICECCS.2019.00027
33. Xue, B., Zhang, M., Easwaran, A., Li, Q.: PAC model checking of black-box continuous-time dynamical systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **39**(11), 3944–3955 (2020). https://doi.org/10.1109/TCAD.2020.3012251
34. Younes, H.L.S., Simmons, R.G.: Probabilistic verification of discrete event systems using acceptance sampling. In: Brinksma, E., Larsen, K.G. (eds.) CAV. LNCS, vol. 2404, pp. 223–235. Springer (2002). https://doi.org/10.1007/3-540-45657-0_17
35. Zarei, M., Wang, Y., Pajic, M.: Statistical verification of learning-based cyber-physical systems. In: Ames, A.D., Seshia, S.A., Deshmukh, J. (eds.) HSCC. pp. 12:1–12:7. ACM (2020). https://doi.org/10.1145/3365365.3382209
36. Zuliani, P., Platzer, A., Clarke, E.M.: Bayesian statistical model checking with application to simulink/stateflow verification. In: HSCC. p. 243–252. Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1755952.1755987