Statistical Verification using Surrogate Models and Conformal Inference and a Comparison with Risk-Aware Verification

XIN QIN and YUAN XIA, University of Southern California, USA
ADITYA ZUTSHI, Galois, Inc., USA
CHUCHU FAN, Massachusetts Institute of Technology, USA
JYOTIRMOY V. DESHMUKH, University of Southern California, USA

Uncertainty in safety-critical cyber-physical systems can be modeled using a finite number of parameters or parameterized input signals. Given a system specification in Signal Temporal Logic (STL), we would like to verify that for all (infinite) values of the model parameters/input signals, the system satisfies its specification. Unfortunately, this problem is undecidable in general. *Statistical model checking* (SMC) offers a solution by providing guarantees on the correctness of CPS models by statistically reasoning on model simulations. We propose a new approach for statistical verification of CPS models for user-provided distribution on the model parameters. Our technique uses model simulations to learn *surrogate models*, and uses *conformal inference* to provide probabilistic guarantees on the satisfaction of a given STL property. Additionally, we can provide prediction intervals containing the quantitative satisfaction values of the given STL property for any user-specified confidence level. We compare this prediction interval with the interval we get using risk estimation procedures. We also propose a refinement procedure based on Gaussian Process (GP)-based surrogate models for obtaining fine-grained probabilistic guarantees over sub-regions in the parameter space. This in turn enables the CPS designer to choose assured validity domains in the parameter space for safety-critical applications. Finally, we demonstrate the efficacy of our technique on several CPS models.

CCS Concepts: • **Theory of computation** → **Logic and verification**; • **Mathematics of computing** → *Probabilistic inference problems;*

Additional Key Words and Phrases: Conformal inference, risk measures

ACM Reference format:
Xin Qin, Yuan Xia, Aditya Zutshi, Chuchu Fan, and Jyotirmoy V. Deshmukh. 2024. Statistical Verification using Surrogate Models and Conformal Inference and a Comparison with Risk-Aware Verification. *ACM Trans. Cyber-Phys. Syst.* 8, 2, Article 22 (May 2024), 25 pages.
https://doi.org/10.1145/3635160

22

The authors would like to thank the anonymous reviewers for their feedback, and thank Francesca Cairoli for valuable discussions on statistical model checking. The National Science Foundation supported this work through the following grants: CAREER award (SHF-2048094), CNS-1932620, CNS-2039087, FMitF-1837131, CCF-SHF-1932620, the Airbus Institute for Engineering Research, and funding by Toyota R&D and Siemens Corporate Research through the USC Center for Autonomy and AI. This article solely reflects the opinions and conclusions of its authors and not the sponsors.
Authors' addresses: X. Qin, Y. Xia, and J. V. Deshmukh, University of Southern California, Los Angeles, CA; e-mails: xinqin@usc.edu, yuanxia@usc.edu, jdeshmuk@usc.edu; A. Zutshi, Galois, Inc., Portland, OR; e-mail: aditya.zutshi@galois.com; C. Fan, Massachusetts Institute of Technology, Cambridge, MA; e-mail: chuchu@mit.edu.

2378-962X/2024/05-ART22
https://doi.org/10.1145/3635160

1 INTRODUCTION

Most cyber-physical systems are highly complex systems with nonlinear behaviors that operate in uncertain operating environments. As these systems are often safety-critical, it is desirable to obtain strong assurances on their safe operation. To achieve this goal, recent research has been focused on effective and sound verification algorithms [16, 17, 25, 26, 28, 49, 50], and scalable best-effort approaches which lack explicit coverage guarantees [55]. However, factors like complexity and stochasticity of the operating environments, curse of dimensionality, the nonlinearity of dynamics pose a significant scalability challenge for verification procedures. In this article, we address the problem of analyzing the effects of uncertainty in the environment on the correctness of a given CPS model $\mathcal{M}$. We assume that the uncertainty in the environment is modeled as a parameter vector (θ) that takes values from some set Θ, distributed according to some user-provided distribution $\mathcal{D}_\Theta$. Such a parameter vector could also include time-varying parameters (representing time-discretized input signals). For a sample of θ, we assume that the output trajectories of the model (denoted ξ_θ) are deterministic, i.e., the model is free of any *internal* stochastic behavior.

We assume that the correctness of the given CPS model is expressed using a real-valued function of its input/output trajectories. In many of our examples, we assume this function to be the *robust satisfaction value* or *robustness* of a given **Signal Temporal Logic (STL)** [35]. Given a formula φ and a trajectory $x(t)$, the robustness $\rho(\varphi, x)$ approximates the degree of satisfaction of φ by x [15, 18]. We are primarily interested in building a surrogate model $\hat{\mu}$ to approximate the joint distribution of θ and $\rho(\varphi, \mathcal{M}(\theta))$, and explore the use of such model to help answer the following specific questions:

(1) Given a threshold ϵ, and $\theta \sim \mathcal{D}_\Theta$, does the probability of the model satisfying a given STL property φ exceed $1-\epsilon$?

$$(\theta \sim \mathcal{D}_\Theta) \stackrel{?}{\implies} P(\mathcal{M}(\theta) \models \varphi) \geq 1-\epsilon \tag{1.1}$$

(2) For some user-provided threshold ϵ, and $\theta \sim \mathcal{D}_\Theta$, can we find an interval $[\ell, u]$ s.t. the probability that the robustness value of a model behavior $\mathcal{M}(\theta)$ w.r.t. a given STL property φ lies in $[\ell, u]$ greater than $1-\epsilon$? i.e.,

$$\theta \sim \mathcal{D}_\Theta \implies P(\rho(\varphi, \mathcal{M}(\theta)) \in [\ell, u]) > 1-\epsilon \tag{1.2}$$

Statistical model checking (SMC) [2, 30, 45, 47, 56, 57] approaches have been used in the past to establish the above two assertions. The most popular SMC methods use statistical hypothesis testing procedures to check whether the hypothesis that (1.1) and (1.2) are true can be accepted with confidence exceeding user-specified thresholds α, β for respectively committing a type I error (i.e., rejecting the hypothesis when it is true) or a type II error (i.e., accepting the hypothesis when it is not true). SMC methods provide the user with conditions on the number of simulations required, α, β and ϵ in order to accept or reject the hypotheses. Unlike SMC that requires a certain number of samples to answer the above questions, our use of surrogate models can establish the assertions without the requirements for sampling numbers, which will be explained later. Furthermore, our use of surrogate models can help automatically provide a new distribution of $\theta \sim \mathcal{D}'_\Theta$, such that (1.2) holds true.

Approach. To establish assertions such as (1.1) or (1.2), we present an approach based on conformal inference, a technique for giving confidence intervals with marginal coverage guarantees. A unique feature of our technique is that it does not make any assumptions on the user-provided distribution on the parameter space or the dynamics represented by the model. While existing techniques based on uncertainty quantification using Gaussian Process based surrogate models assume that the joint distribution of sampled parameter values and target robustness values have

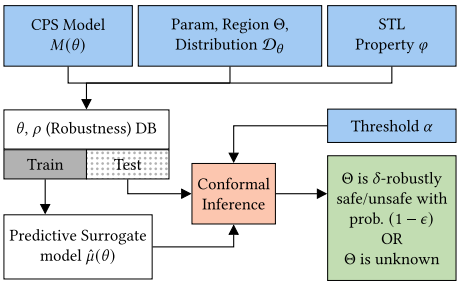


Fig. 1. Overview of our approach.

a Gaussian distribution [38]. SMC techniques although make no assumption on the input distribution, assume that the Boolean outcomes of successive runs of a given program have a binomial distribution, and probabilistic guarantees are based on analyzing properties of the binomial distribution.

The overview of our approach is shown in Figure 1. The first step of our approach is to learn a surrogate model – this can be thought of as a statistical data-driven approximation of the black-box model. For this step, we sample N parameter values from Θ, obtain their corresponding trajectories (ξ_θ) and then compute the robust satisfaction value $\rho(\varphi, \xi_\theta)$ for each trajectory. We then partition this set into a training set and a test set. We then use an off-the-shelf regression technique on the training set treating the sampled parameters in the training set as inputs and the robustness values as outputs of the learned regressor. Here, we can use any parametric technique such as polynomial regression, or a non-parametric technique based on Gaussian process regression or neural networks [7].

Due to the generalizability inherent in regression, the surrogate model can predict the robustness value for all parameter values in Θ. However, most good regression techniques avoid overfitting to the data, and hence will result in some residual error (i.e., between the predicted values and the actual values). Conformal inference is a technique that can leverage these residual errors to give confidence intervals on predicted values. The main idea in conformal inference is: for any given threshold $1-\epsilon$, there is a systematic way to find a prediction interval $[\hat{\mu}(\theta) - d, \hat{\mu}(\theta) + d]$ where the answer must lie with probability greater than $1-\epsilon$. We couple this idea with a global optimizer to obtain confidence intervals for *regions* in the parameter space rather than individual parameter values. This allows us to provide the guarantee specified in Equation (1.3), where $v_{\min}$ and $v_{\max}$ are respectively under- and overapproximations of the predicted robustness over the region Θ.

$$\theta \in \Theta \implies P\left(\rho(\varphi, \xi_\theta) \in [v_{\min} - d,\ \ v_{\max} + d]\right) \geq 1 - \epsilon. \tag{1.3}$$

A strictly positive or strictly negative interval indicates that the Θ is respectively safe or unsafe. However, if the interval contains 0, then the status of Θ remains unknown. The above procedure naturally yields a refinement procedure which allows us to start with a larger region in the parameter space, and split it into smaller regions if the region is deemed unknown. In a smaller region, the accuracy of the surrogate model improves (due to more data in a smaller region), and hence previously inconclusive regions can be resolved as safe/unsafe. A naïve version of this splitting algorithm faces the curse of dimensionality—if the parameter space is high-dimensional, then the branch-and-bound procedure ends up creating too many branches which can make the procedure intractable.

The naïve splitting algorithm crucially does not use any smart heuristic to decide where a region should be split, and this can lead to unnecessary exploration of larger regions that can be

(statistically) verified as safe or unsafe. To overcome this shortcoming, we propose the use of Gaussian Process (GP) [40] regression models. There is rich literature on the use of GP-based models for black-box function optimization, and a key idea therein is to explicitly encode sample uncertainty as an optimization objective. This allows GP-based methods to tradeoff between *exploitation* (searching in the vicinity of a local minimum) and *exploration* (searching in neighborhoods with high sample uncertainty). We propose to use a similar heuristic that adaptively splits regions based on sample uncertainty.

Our method can scale to CPS models encoding complex dynamics and large state spaces, as well as reasonably large parameter spaces. The results of our method can be used to characterize safe operating regions in the parameter space, and to build (probabilistic) safety assurance cases. With respect to analysis times, our method compares favorably with approaches based on . A key difference is that unlike SMC- and PAC-based methods, the first step in our method is to construct a surrogate model. This crucially allows us to provided a guarantee that is not a function of the number of samples. In fact, our method can potentially provide the needed level of probabilistic guarantees with *any number of samples*. This is because we build a surrogate model from samples; if the surrogate model is of poor accuracy due to a limited number of samples, conformal inference will predict a wider prediction interval *with the same probability* $1 - \epsilon$, while for a more accurate model, the prediction interval will be narrower. Thus, conformal inference allows a tradeoff between sample complexity and the tightness of the guarantee *independent* of the level of the guarantee itself. Finally, while SMC-based methods focus solely on the problem of probabilistic verification, our method can enable other model-based analyses: (1) we can give probabilistic safe regions in the parameter space; for example, these can be used to define high-confidence operating regimes for the model, (2) our technique can be used for statistical debugging approaches such as [6] and [14], and (3) we can extend our technique to identify parameter sensitivity by combining the core regression procedure with dimensionality reduction techniques such as principal component analysis [7].

While conformal prediction has emerged as an important statistical technique to provide probabilistic guarantees, an important concurrent development is the work on scenario-based verification using the notion of *risk measures* [4]. Measures such as value-at-risk and conditional-value-at-risk allow quantifying the risk of the given CPS application failing a particular quantitative specification (e.g., specified as an STL formula). In this article, we empirically compare the probabilistic guarantees obtained using the risk estimation formulation with those obtained using conformal prediction.

To summarize, the main contributions of this article are:

(1) A technique based on surrogate models to approximate the robustness of a given specification learned using off-the-shelf regression techniques;
(2) A new technique for generating prediction intervals for the robustness of a specification with user-specified probabilistic thresholds;
(3) Algorithms to partition the parameter space of a model into safe, unsafe, and unknown regions based on conformal inference on the surrogate models;
(4) Experimental validation on CPS models demonstrating the real-world applicability of our methods; and
(5) Empirical comparison with methods to provide probabilistic guarantees based on estimating tail risk measures.

The rest of this article is organized as follows: Section 2 provides the background and notation; Section 3 explains how we use conformal inference for providing probabilistic guarantees on satisfaction/violation of a given STL property over a region in the parameter space; Section 4 presents

our algorithm for refining parameter spaces using Gaussian Processes. Finally, Section 5 illustrates our approach using several case studies, and Section 6 presents our conclusions.

2 PRELIMINARIES

Definition 2.1 (Signals, Black-box Models). We define a signal[1] or a trajectory ξ as a function from a finite set $\text{dom} \subseteq [0, T]$ for some $T \in \mathbb{R}^{\geq 0}$ to a compact set of values $\mathcal{X}$. The signal value at time t is denoted as $\xi(t)$. A parameter space Θ is some compact subset of $\mathbb{R}^k$. A model $\mathcal{M}$ is a function that maps a parameter value $\theta \in \Theta$ to an output signal ξ_θ.

We note that the above definition permits parameterized *input signals* for the model. We can define such signals using a function known as a signal generator that maps specific parameter values to signals. For example, a piecewise linear signal containing k linear segments can be described using $k + 1$ parameters, k corresponding to the starting point for each segment and 1 for the end-point of the final segment.

We assume that $\theta \in \Theta$ is a random variable that follows a (truncated) distribution $\mathcal{D}_\theta$ with **probability density function (PDF)** $f(\theta)$ and $\forall \theta \notin \Theta, f(\theta) = 0$. If we only wish to draw samples from a subset $S \subseteq \Theta$ (by dropping samples from $\Theta \backslash S$), the corresponding distribution of the samples is denoted by $\mathcal{D}_\theta \downarrow S$ and follows the PDF shown below:

$$f'(\theta) = \begin{cases} \frac{f(\theta)}{\int_{\tau \in S} f(\tau) d\tau} & \text{if } \theta \in S \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

Instead of closed form descriptions of the generator for ξ_θ (e.g., differential or difference equations), we assume that there is a *simulator* that can generate signals compatible with the semantics of the model $\mathcal{M}$.

Definition 2.2. A simulator for a (deterministic) set Ξ of trajectories is a function (or a program) sim that takes as input a parameter $\theta \in \Theta$, and a finite sequence of time points $t_0, \ldots, t_k$, and returns the signal $(t_0, \text{sim}(\theta, t_0)), \ldots, t_k, \text{sim}(\theta, t_k))$, where for each $i \in \{0, \ldots, k\}$, $\text{sim}(\theta, t_i) = \xi_\theta(t_i)$.

In rest of this article, unless otherwise specified, we ignore the distinction between the signals $\text{sim}(\theta, \cdot)$ and ξ_θ.

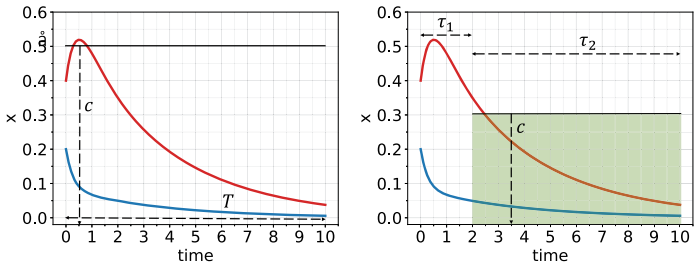
2.1 Signal Temporal Logic

Signal Temporal Logic [35] is a popular formalism that has been widely used to express safety specifications for many CPS applications. STL formulas are defined over signal predicates of the form $f(\xi) \geq c$ or $f(\xi) \leq c$, where ξ is a signal and $f : \mathbb{R}^n \to \mathbb{R}$ is a real-valued function and $c \in \mathbb{R}$. STL formulas are written using the grammar shown in Equation (2.2). Here, we assume that $I = [a, b]$, where $a, b \in \mathbb{R}^{\geq 0}, a \leq b$, and $\sim \in \{\leq, \geq\}$.

$$\varphi, \psi := \mathit{true} \mid f(\xi) \sim c \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \mathbf{F}_I \varphi \mid \mathbf{G}_I \varphi \mid \varphi \, \mathbf{U}_I \, \psi. \tag{2.2}$$

In the above syntax, **F** (eventually), **G** (always), and **U** (until) are temporal operators. Given $t \in \mathbb{R}^{\geq 0}$ and $I = [a, b]$, we use $t + I$ to denote $[t + a, t + b]$. Given a signal ξ and a time t, we use $(\xi, t) \models \varphi$ to denote that ξ satisfies φ at time t, and $\xi \models \varphi$ as shorthand for $(\xi, 0) \models \varphi$. The Boolean satisfaction

[1] Conventionally, signals are defined over continuous-time; however, in a practical setting, such as in a simulator, we only obtain signal values at a finite set of time intervals. In such a case, it is common to assume that the underlying continuous-time signal can be recovered using an appropriate interpolation scheme. The discrete-time vs. continuous-time interpretation is not of particular importance in this article. The only bearing it has is on the satisfaction of a given STL formula—we use point-wise semantics of STL over piecewise constant interpolated signals that produce identical satisfaction values for continuous/discrete-time signals.



(a) $\varphi_1 = \mathbf{G}_{[0,10]}(x(t) < 0.5)$ (b) $\varphi_2 = \mathbf{F}_{[0,2]}\mathbf{G}_{[0,8]}(\|x(t)\| < 0.3)$

Fig. 2. Trajectories satisfying/violating STL formulas.

semantics of an STL formula can be recursively in terms of the satisfaction of its subformulas over the a signal. For $\sim \in \{\leq, \geq\}$, ξ: $(\xi, t) \models f(\xi) \sim c$ if $f(\xi(t)) \sim c$ is true. The semantics of the Boolean operators for negation ($\neg$), conjunction ($\wedge$) and disjunction ($\vee$) can be obtained in the usual fashion by applying the operator to the Boolean satisfaction of its operand(s). The value of $(\xi, t) \models \mathbf{F}_I\varphi$ is true iff $\exists t' \in t + I$ s.t. $(\xi, t') \models \varphi$, while $(\xi, t) \models \mathbf{G}_I\varphi$ iff $\forall t' \in t + I$, $(\xi, t') \models \varphi$. The formula $\varphi \mathbf{U}_I \psi$ is satisfied at time t if there exists a time $t' \geq t$ s.t. ψ is true, and for all $t'' \in [t, t']$, φ is true.

STL is also equipped with quantitative semantics that define the *robust satisfaction value* or *robustness*—a function mapping a formula φ and the signal ξ to a real number [15, 18]. Informally, robustness can be viewed as a degree of satisfaction of an STL formula φ. While many competing definitions for robust satisfaction value exist [3, 27, 43], we use the original definitions [15] in this article.

Definition 2.3. The robustness value is a function ρ mapping φ, the trajectory ξ, and a time $t \in \xi.\text{dom}$ as follows:

$$\begin{aligned}
\rho(f(\xi) \geq c, \xi, t) &= f(\xi(t)) - c \\
\rho(\neg\varphi, \xi, t) &= -\rho(\varphi, \xi, t) \\
\rho(\varphi \wedge \psi, \xi, t) &= \min(\rho(\varphi, \xi, t), \rho(\psi, \xi, t)) \\
\rho(\varphi\, \mathbf{U}_I\, \psi) &= \sup_{t_1 \in t+I} \min(\rho(\psi, \xi, t_1), \inf_{t_2 \in [t, t_1)} \rho(\varphi, \xi, t_2))
\end{aligned}$$

The robustness values for other Boolean and temporal operators can be derived from the above definition; for example, $\mathbf{G}_I\varphi$ and $\mathbf{F}_I\varphi$ are a special case of the semantics for until ($\mathbf{U}_I$) respectively evaluating to the minimum and maximum of the robustness of φ over the interval I.

Example 2.4. Consider the time-reversed van Der Pol oscillator specified as $\dot{x_1} = -x_2$, $\dot{x_2} = 4(x_1^2 - 1)x_2 + x_1$. Figure 2 illustrates the satisfaction (indicated in blue) and violation (indicated in red) of two example specifications by $x_1(t)$: (a) φ_1 specifies that for any time $t \in [0, 10]$, the value of the trajectory $x(t)$ should be less than 0.5 and (b) φ_2 specifies that from some time within the first two time units, $x(t)$ settles in the region $[-0.3, 0.3]$ for eight time units.

2.2 Learning Surrogate Models

In this section, we discuss learning of surrogate models for a given black-box model $\mathcal{M}$. A surrogate model is essentially a *quantitative abstraction* of the original black-box model. Quantitative abstractions have been explored in the theory of **weighted transition systems (WTS)** [11]. A WTS is a transition system where every transition is associated with weights, and a quantitative property of the WTS maps sequences of states of the WTS to a real number computed using some arithmetic operations on the weights. Quantitative abstractions focus on sound proofs for quantitative properties. We observe that we can view the robustness of an STL property as a quantitative

property evaluated on the system trajectory. We introduce two new notions of quantitative abstractions defined on the trajectories of a system.

Definition 2.5 (δ-surrogate model). Let ξ_θ be the trajectory obtained by simulating the $\mathcal{M}$ with the parameter θ, where $\theta \in \Theta$. Let γ be a *quantitative property* on ξ_θ, i.e., γ maps ξ_θ to a real number. We say that a model $\hat{\mu}$ that maps θ to a real number is an δ-distance-preserving quantitative abstraction or an δ-surrogate model of $\mathcal{M}$ and γ if

$$\forall \theta \in \Theta : |\gamma(\xi_\theta) - \hat{\mu}(\theta)| \leq \delta. \tag{2.3}$$

Essentially, the δ-surrogate model guarantees that the value of the quantitative property γ evaluated on ξ_θ (obtained from the original model $\mathcal{M}$) is no more than δ away from the value that it predicts. The idea is that the δ-surrogate model could be systematically derived from the original model, and could be significantly simpler than the original model making it amenable to formal analysis. For example, if we have an δ-surrogate model, then we can prove that a given property holds by systematically sampling the parameter space Θ.

In general, such models could be hard to obtain; hence, we propose a probabilistic relaxation known as the (δ, ϵ)-*probabilistic surrogate* model, where condition (2.3) is replaced by (2.4).

Definition 2.6 ((δ, ϵ)-probabilistic surrogate model). Given a model $\mathcal{M}$, a quantitative property γ, and a user-specified bound $\epsilon \in [0, 1)$, we say that $\hat{\mu}$ is a (δ, ϵ)-probabilistic surrogate model if:

$$P\left(|\gamma(\xi_\theta) - \hat{\mu}(\theta)| \leq \delta \mid \theta \sim \mathcal{D}_\theta\right) \geq 1 - \epsilon. \tag{2.4}$$

We now explain how we can obtain (δ, ϵ)-probabilistic surrogate models for an arbitrary quantitative property γ. The basic idea is to use statistical learning techniques: we sample Θ in accordance with the distribution $\mathcal{D}_\theta$ to obtain a finite set of parameter values $\widehat{\Theta}$. For each $\theta_i \in \widehat{\Theta}$, we simulate the model to obtain ξ_{θ_i} and compute $\gamma(\xi_{\theta_i})$. We then compute the surrogate model $\hat{\mu}$ using parametric regression models (e.g., linear, polynomial functions) or nonparametric regression methods (e.g., neural networks and Gaussian Processes) [5, 20, 40]. We now briefly review some of these regression methods, and in Section 3 explain how we can obtain δ values for a user-provided bound ϵ.

Polynomial Regression. Polynomial regression assumes a polynomial relationship between independent variables X and the dependent variable Y. It aims to fit a polynomial curve to the input and output data in a way that minimizes a suitable loss function. A commonly used loss function is the least square error (or the sum of squares of residuals). Typically, a polynomial regression requires the user to specify the degree of the polynomial to use. Polynomial regression generally has high tolerance to the function's curvature level, but has high sensitivity to the outliers. In our experiments, we restrict the polynomial degree to 2.

Neural Network Regression. Neural networks [7] offer a high degree of flexibility for regressing arbitrary nonlinear functions. While there are many different NN architectures, we use a simple multi-layer perceptron model with a stochastic gradient-based optimizer. This model simply updates its parameters based on iterative steps along the partial derivatives of the loss function.

Gaussian Process based Regression Model [40]. A **Gaussian Process (GP)** is a stochastic process, i.e., it is a collection of random variables W_θ indexed by θ, where θ ranges over some discrete or dense set. The key property of GP is that any finite sub-collection of these random variables has a multi-variate Gaussian distribution. GP models are popular as non-parametric regression methods used for approximating arbitrary continuous functions with the appropriate *kernel functions.* A GP can be used to express a prior distribution on the space of functions, e.g., from a domain $\mathbb{R}^n$ to $\mathbb{R}$. Let $F : \mathbb{R}^n \to \mathbb{R}$ be a random function. Then, we say that F is a centered Gaussian process with kernel k, if for every $(x_1, \ldots, x_n) \in \mathbb{R}^n$, there exists a positive semi-definite matrix Σ such that

$[F(x_1), \ldots, F(x_n)] \sim (0, \Sigma)$. The $(i,j)^{th}$ entry of Σ, i.e., $\Sigma_{ij} = k(x_i, x_j)$ for some kernel function k. The matrix Σ is called the covariance matrix, and the function k measures the joint variability of x_i and x_j. There are several kernel functions that are popular in literature: the squared exponential kernel, the 5/2 Matérn kernel, and so on. In our experiments, we use a *sum* kernel function that is the addition of a dot product kernel and a white noise kernel (explained in Section 3.4).

3 CONFORMAL INFERENCE

Conformal inference [31, 32] is a framework to quantify the accuracy of predictions in a regression framework [52]. It can provide guarantees using a finite number of samples, without making assumptions on the distribution of data used for regression or the technique used for regression. We explain the basic idea of conformal inference, and then explain how we adapt it to our problem setting.

3.1 Conformal Inference Recap

Consider i.i.d. regression data $Z_1, \cdots, Z_m$ drawn from an arbitrary joint $\mathcal{D}_{XY}$, where each $Z_i = (X_i, Y_i)$ is a random variable in $\mathbb{R}^n \times \mathbb{R}$, consisting of n-dimensional feature vectors X_i and a response variable Y_i. Suppose we fit a surrogate model to the data, and we now wish to use this model to predict a new response Y_{m+1} for a new feature value X_{m+1}, with no assumptions on $\mathcal{D}_{XY}$. Formally, given a positive value $\alpha \in (0,1)$, conformal inference constructs a prediction band $C \subseteq \mathbb{R}^n \times \mathbb{R}$ based on $Z_1, \cdots, Z_n$ with property (3.1).

$$P(Y_{m+1} \in C(X_{m+1})) \geq 1 - \alpha. \tag{3.1}$$

Here, the probability is over $m+1$ i.i.d. draws $Z_1, \cdots, Z_{m+1} \sim \mathcal{D}_{XY}$, and for a point $x \in \mathbb{R}^n$ we denote $C(x) = \{y \in \mathbb{R} : (x,y) \in C\}$. The parameter α is called the *miscoverage level* and $1-\alpha$ is called the *probability threshold*. Let

$$\mu(x) = \mathbb{E}(Y \mid X = x), x \in \mathbb{R}^n$$

denote the regression function, where $\mathbb{E}(W)$ denotes the expected value of the random variable W. The regression problem is to estimate such a conditional mean of the test response Y_{m+1} given the test feature $X_{m+1} = x$. Common regression methods use a regression model $g(x, \eta)$ and minimize the sum of squared residuals of such model on the m training regression data $Z_1, \cdots, Z_m$, where η are the parameters of the regression model. An estimator for μ is given by $\hat{\mu}(x) = g(x, \hat{\eta})$, where

$$\hat{\eta} = \arg\min_{\eta} \frac{1}{m} \sum_{i=1}^{m} (Y_i - g(X_i, \eta))^2 + \mathcal{R}(\eta)$$

and $\mathcal{R}(\eta)$ is a regularizer. In [31], Lei et al. provide a technique called *split conformal prediction* that we use to construct prediction intervals that satisfy the finite-sample guarantees as in Equation (3.1). The procedure is described in Algorithm 1 as a function `ConfInt` which takes as input the i.i.d. training data $\{(X_i, Y_i)\}_{i=1}^m$, miscoverage level α and any regression algorithm `Reg`. Algorithm 1 begins by splitting the training data into two equal-sized disjoint subsets. Then a regression estimator $\hat{\mu}$ is fit to the training set $\{(X_i, Y_i)\} : i \in \mathcal{I}_1)$ using the regression algorithm `Reg` (Line 2). Then the algorithm computes the absolute residuals R_is on the test set $\{(X_i, Y_i)\} : i \in \mathcal{I}_2)$ (Line 3). For the desired probability threshold $\alpha \in [0,1)$, the algorithm sorts the residuals in ascending order $\{R_i : i \in \mathcal{I}_2\}$ and finds the residual at the position given by the expression: $\lceil (n/2+1)(1-\alpha) \rceil$. This residual is used as the confidence range d. In [31], Lei et al. prove that the prediction interval at a new point X_{m+1} is given by such $\hat{\mu}$ and d that Theorem 3.1 is valid.

Theorem 3.1 (Theorem 2.1 in [31]). *If $(X_i, Y_i), i = 1, \cdots, m$ are i.i.d., then for an new i.i.d. draw (X_{m+1}, Y_{m+1}), using $\hat{\mu}$ and d constructed in Algorithm 1, we have that $P(Y_{m+1} \in [\hat{\mu}(X_{m+1}) -$*

ALGORITHM 1: Conformal regression algorithm $\texttt{ConfInt}(\{(X_i, Y_i)\}_{i=1}^m, \alpha, \texttt{Reg})$

input: Data $\{(X_i, Y_i)\}_{i=1}^m$, miscoverage level α, regression algorithm Reg
output: Regression estimator $\hat{\mu}$, confidence range d
Randomly split $\{1, \cdots, m\}$ into two equal-sized subsets $\mathcal{I}_1, \mathcal{I}_2$;
$\hat{\mu} = \texttt{Reg}((X_i, Y_i) : i \in \mathcal{I}_1)$;
$R_i = |Y_i - \hat{\mu}(X_i)|, i \in \mathcal{I}_2$;
$d =$ the kth smallest value in $\{R_i : i \in \mathcal{I}_2\}$, where $k = \lceil (m/2+1)(1-\alpha) \rceil$;
return $\hat{\mu}, d$

$d, \hat{\mu}(X_{m+1}) + d]) \geq 1 - \alpha$. *Moreover, if we additionally assume that the residuals* $\{R_i : i \in \mathcal{I}_2\}$ *have a continuous joint distribution, then* $P(Y_{m+1} \in [\hat{\mu}(X_{m+1}) - d, \hat{\mu}(X_{m+1}) + d]) \leq 1 - \alpha + \frac{2}{m+2}$. □

Generally speaking, as we improve our surrogate model $\hat{\mu}$ of the underlying regression function μ, the resulting conformal prediction interval decreases in length. Intuitively, this happens because a more accurate $\hat{\mu}$ leads to smaller residuals (or ϵ in Section 2.2), and conformal intervals are essentially defined by the quantiles of the (augmented) residual distribution. Note that Theorem 3.1 asserts marginal coverage guarantees, which should be distinguished with the conditional coverage guarantee $P(Y_{m+1} \in C(x) \mid X_{m+1} = x) \geq 1 - \alpha$ for all $x \in \mathbb{R}^n$. The latter one is a much stronger property and hard to be achieved without assumptions on $\mathcal{D}_{XY}$.

3.2 Computing (d, ϵ)Probabilistic Surrogate Models

We assume that the parameter value θ and $\rho(\varphi, \xi_\theta)$ follow a joint (unknown) distribution $\mathcal{D}_{\theta, \rho(\varphi)}$ that we wish to empirically estimate. As indicated in Section 2.2, the first step to learning a (δ, ϵ)-probabilistic surrogate model is based on sampling $\mathcal{D}_{\theta, \rho(\varphi)}$ and applying regression methods. We draw m i.i.d samples $\widehat{\Theta} = \{\theta_1, \cdots, \theta_m\}$ from $\mathcal{D}_\theta$ and compute the robustness values $\rho_i = \rho(\varphi, \xi_{\theta_i})$ for each model trajectory corresponding to the parameter θ_i. Lemma 3.2 follows from Theorem 3.1.

Lemma 3.2. *Let* $(\hat{\mu}, d) = \texttt{ConfInt}(\{\theta_i, \rho_i\}, \epsilon, \texttt{Reg})$, *where* ConfInt *is as defined in Algorithm 1,* $1 - \epsilon$ *is a user-provided probability threshold,* Reg *is some regression algorithm, and* $d \in \mathbb{R}$, *then* $\hat{\mu}$ *is a* (d, ϵ)*-probabilistic surrogate model.*

We now show how we can use (d, ϵ)-probabilistic surrogate models to perform statistical verification. Theorem 3.3 shows that the confidence range returned by the conformal inference procedure can be extended over the entire parameter space.

Theorem 3.3. *Let*

(1) $(\theta_i, \rho_i), i = 1, \cdots, m$ *be i.i.d. samples drawn from the joint distribution* $\mathcal{D}_{\theta, \rho(\varphi)}$ *of* $\theta \in \Theta$ *and* $\rho(\varphi, \xi_\theta)$,
(2) Reg *be a regression algorithm,*
(3) $1 - \epsilon$ *be a user-provided probability threshold,*
(4) $(\hat{\mu}, d) = \texttt{ConfInt}(\{\theta_i, \rho_i\}, \epsilon, \texttt{Reg})$, *i.e.,* $\hat{\mu}$ *is the surrogate model and* d *is the confidence range returned by Algorithm 1,*
(5) $v^*_{\max} = \max_{\theta \in \Theta} \hat{\mu}(\theta)$, *and,* $v^*_{\min} = \min_{\theta \in \Theta} \hat{\mu}(\theta)$.

Then,

$$P\left(\rho(\varphi, \xi_\theta) \in [v^*_{\min} - d, v^*_{\max} + d] \mid \theta \sim \mathcal{D}_\theta\right) \geq 1 - \epsilon. \tag{3.2}$$

Proof. From Theorem 3.1, we know that *any new* i.i.d. sample $(\theta', \rho(\varphi, \xi_{\theta'}))$ from $\mathcal{D}_{\theta, \rho(\varphi)}$ satisfies:

$$P(\rho(\varphi, \xi_{\theta'}) \in [\hat{\mu}(\theta') - d, \hat{\mu}(\theta') + d]) \geq 1 - \epsilon. \tag{3.3}$$

By definition, $v^*_{\min} \leq \hat{\mu}(\theta') \leq v^*_{\max}$. Combining this with Equation (3.3), we get the desired result. □

Theorem 3.3 requires us to obtain the minimum/maximum values of the surrogate model over a given region in the parameter space. If $\hat{\mu}(\theta)$ is a non-convex function and the chosen optimization algorithm cannot compute the perfect optimal value $v^*_{\min}$ or $v^*_{\max}$, but can only give conservative estimates of the optimal value, we can update the predicted interval in Theorem 3.3 as follows:

Corollary 3.4. *Let $v_{\min}$ and $v_{\max}$ be respectively under- and over-approximations of $v^*_{\min}$ and $v^*_{\max}$, then*

$$P\left(\rho(\varphi, \xi_\theta) \in [v_{\min} - d, v_{\max} + d] \mid \theta \sim \mathcal{D}_\theta\right) \geq 1 - \epsilon \tag{3.4}$$

The bounds $v_{\min}$ and $v_{\max}$ in Corollary 3.4 can be computed using global optimization solvers, SMT sovlers, or range analysis tools for neural networks [17, 50] (for neural network regression).

We can use the bounds obtained in Theorem 3.3 (similarly those with Corollary 3.4) to derive probabilistic bounds on the Boolean satisfaction of a given STL property φ, as expressed in Theorem 3.5.

Theorem 3.5. *If $v^*_{\min} - d > 0$, then $P_{\mathcal{D}_\theta}(\xi_\theta \models \varphi \mid \theta \in \Theta) \geq 1 - \epsilon$. If $v^*_{\max} + d < 0$, then $P_{\mathcal{D}_\theta}(\xi_\theta \not\models \varphi \mid \theta \in \Theta) \geq 1 - \epsilon$.*

Proof. From [18], we know that $\rho(\varphi, \xi_\theta) > 0 \implies \xi_\theta \models \varphi$. Thus, if the lower bound of the prediction interval in Theorem 3.3 is positive, then $\xi_\theta \models \varphi$. The second case follows by a similar argument. □

If the first statement in the above theorem holds, we say that Θ *safe*, if the second statement holds, we say that Θ is *unsafe*, and if neither statement holds (i.e., the predicted interval contains 0), then we say that Θ is *unknown*. While Theorem 3.5 allows us a way to identify whether a region in the parameter space is is safe (or unsafe), unfortunately there are two challenges: (1) the function mapping θ to $\rho(\varphi, \overline{\mathsf{sim}}(\theta))$ is a highly nonlinear function in general, and an *a priori* choice for a regression algorithm Reg that fits this function with small residual values may be difficult and (2) if there is large variation in the value of the regression function over Θ, it is likely that the conformal interval contains 0, thereby marking Θ as unknown. To circumvent this issue, one solution is to split the parameter space Θ into smaller regions where it may be possible to get narrow conformal intervals at the same level of probability threshold. We present a naïve algorithm based on parameter-space partitioning next.

3.3 Naïve Parameter Space Partitioning

We now present an algorithm that uses Theorem 3.3 (or Corollary 3.4) to provide probabilistic guarantee by recursively splitting the parameter space Θ into smaller regions such that each region can be labeled as safe, unsafe or unknown. The basic idea of this algorithm is to compute the conformal interval using Theorem 3.3 and then check if $v^*_{\min} - d < 0$ and $v^*_{\max} + d > 0$. If yes, we need to partition the region. After partitioning the region, we have to repeat the process of computing the conformal interval for each of the sub-regions. Note that the probability in Theorem 3.3 (and Theorem 3.1) is marginal, being taken over all the i.i.d. samples $\{\theta_i, \rho_i\}$ from $\mathcal{D}_{\theta, \rho(\varphi)}$. Therefore, when we work on each subset $S \subseteq \Theta$ after the partitions, we will have to restrict θ to be in S (according to Equation (2.1)) to ensure that the Theorem 3.3 is valid. We abuse the notation and denote the joint distribution $\mathcal{D}_{\theta, \rho(\varphi)}$ when θ is restricted to be sampled from $S \subseteq \Theta$ by $\mathcal{D}_{\theta, \rho(\varphi)} \downarrow S$.

Algorithm 2 searches over the parameter space Θ and partitions it to sets Θ^+, Θ^-, and Θ^U, along with the prediction intervals for the robustness values in each set. We first check if the robustness value is strictly positive or negative and accordingly add the region being inspected S into Θ^+ or

ALGORITHM 2: Parameter space partition with respect to STL formulas using conformal regression.

input: Parameter space Θ and corresponding distribution $\mathcal{D}_\theta$, simulator sim and interpolation method to provide $\overline{\text{sim}}$, miscoverage level α, regression algorithm Reg, an STL formula φ, a vector Δ

output: Parameter set Θ^+ that lead to satisfaction of φ, Θ^- that lead to violation of φ, and the rest parameter set Θ^U that is undecided

1 $\Theta^+, \Theta^-, \Theta^U \leftarrow \emptyset, \Theta^r \leftarrow \{\Theta\}$;
2 **while** $\Theta^r \neq \emptyset$ **do**
3 $\quad S \leftarrow \texttt{Pop}(\Theta^r)$;
4 $\quad \theta_1, \cdots, \theta_m \leftarrow \texttt{IID\_Sample}(\mathcal{D}_\theta \downarrow S)$;
5 $\quad$ **for** $i = 1, \cdots, m$ **do**
6 $\quad\quad \rho_i \leftarrow \rho(\varphi, \xi_{\theta_i})$;
7 $\quad \hat{\mu}, d \leftarrow \texttt{ConfInt}(\{(\theta_i, \rho_i)\}_{i=1}^m, \alpha, \texttt{Reg})$;
8 $\quad v_{\max} \leftarrow \max_{\theta \in S} \hat{\mu}(\theta), v_{\min} \leftarrow \min_{\theta \in S} \hat{\mu}(\theta)$;
9 $\quad$ **if** $v_{\min} - d \geq 0$ **then**
10 $\quad\quad \Theta^+ \leftarrow \Theta^+ \cup (S, [v_{\min} - d, v_{\max} + d])$;
11 $\quad$ **else if** $v_{\max} + d \leq 0$ **then**
12 $\quad\quad \Theta^- \leftarrow \Theta^- \cup (S, [v_{\min} - d, v_{\max} + d])$;
13 $\quad$ **else if** $\texttt{Diameters}(S) < \Delta\texttt{Diameters}(\Theta)$ **then**
14 $\quad\quad \Theta^U \leftarrow \Theta^U \cup (S, [v_{\min} - d, v_{\max} + d])$;
15 $\quad$ **else**
16 $\quad\quad \Theta^r.\texttt{Push}(\texttt{Partition}(S, \texttt{Reg}))$;
17 **return** $\Theta^+, \Theta^-, \Theta^U$;

Θ^- (Lines 10 and 12). When Algorithm 2 cannot decide whether S belongs to Θ^+ or Θ^- the interval contains 0, we first check if for all n, the diameter of S along the nth parameter dimension less than the fraction $\Delta_n \texttt{Diameters}(\Theta)_n$. We assume that the vector Δ is provided by the user. If yes, the region is marked as unknown. Otherwise, we partition S into a number of subregions, that are then added to a worklist of regions (Line 16). In our implementation, in order to keep the number of subsets to be explored bounded, we randomly pick a dimension in the parameter space, and split the parameter space into two equal subsets along that dimension. Note that the partitioning can be accelerated by using parallel computation, but we leave that for future exploration. For each subset S, Algorithm 2 additionally gives the corresponding prediction interval, which indicates how good (or bad) the trajectories satisfy (or violate) φ.

Theorem 3.6. *In Algorithm 1,* $P(\xi_\theta \models \varphi \mid \theta \sim \mathcal{D}_{\theta, \rho(\varphi)} \downarrow \bigcup S \in \Theta^+) \geq 1 - \epsilon$,, *and* $P(\xi_\theta \not\models \varphi \mid \theta \sim \mathcal{D}_{\theta, \rho(\varphi)} \downarrow \bigcup S \in \Theta^-) \geq 1 - \epsilon$.

Theorem 3.6 directly follows from Theorems 3.3 and 3.5 and the total probability theorem.

3.4 Gaussian Processes for Refinement

A drawback of Algorithm 2 is that the naïve splitting procedure is not scalable in high dimensions, and may have poor performance if the safe/unsafe regions have arbitrary shapes. In this section, we instead suggest the use of **Gaussian Processes (GP)** coupled with Bayesian updates to intelligently partition regions. Recall from Section 2, for each parameter value θ, the GP model allows representing the mean $\mu(\theta)$ and $\sigma^2(\theta)$ in terms of samples already explored in the parameter space. In a GP model, at sampled parameter values, the variance is zero, but at points that are away from the sampled values, the variance could be high. We now give the symbolic expressions for the mean and variance of a GP model in terms of a kernel function $k(\theta, \theta)$. A kernel function in GP is

a covariance function of the random variables inside GP. It influences the flexibility and capacity of the GP, as well as its ability to generalize to new data points. For ease of exposition let $\hat{\Theta}$ denote the vector of parameter values already sampled. Then, let Y denote the vector of robustness values for parameter values in $\hat{\Theta}$. Then, from [40], Chapter 2, the posterior of the distribution given observed samples we have: $\mu(\theta) = k(\theta, \hat{\Theta})^\top k(\hat{\Theta}, \hat{\Theta})^{-1} Y$, $\Sigma(\theta) = k(\theta, \theta) - k(\theta, \hat{\Theta}) k(\hat{\Theta}, \hat{\Theta})^{-1} k(\hat{\Theta}, \theta)$, and $\sigma(\theta) = \sqrt{\Sigma(\theta)}$. The main idea is to use the mean and variance of the GP model to prioritize searching parameter values where: (a) the robustness may be close to zero and (b) the variance of the GP model may be high. These two choices give us two different ways to partition the parameter region that we now explain.

In the literature on GP-based Bayesian optimization, there is work on defining *acquisition functions* that are used as targets for optimization. Examples include **UCB (Upper Confidence Bound)** acquisition function that is a combination of the mean and variance of the GP, **EI (Expected Improvement)** which focuses on the expected value of the improvement in the function value, and so on. Inspired by the UCB function that allows a tradeoff between exploration and exploitation, we consider two acquisition functions: (1) the first is focused on pure exploration and uses the variance of the GP as the objective for maximization and (2) the second is the difference between the mean and the standard deviation. The rationale for the second function is that if $\mu(\theta) - \sigma(\theta)$ is lower than 0, then it is an indicator of a low robustness region.

ALGORITHM 3: Parameter Space Partition using GP models `Partition`$(S, \texttt{Reg})$

input: Parameter set S, regression algorithm `Reg`
output: parameter sets $S_1, \ldots$ that will be pushed into the set Θ^r
1 $f(\theta) \leftarrow \text{acquisition}(\mu(\theta), \sigma(\theta))$;
2 $\theta' \leftarrow \arg\max_\theta f(\theta)$;
3 **return** $\text{Split}(S, \theta')$;

Algorithm 3 presents the new `Partition` function of Algorithm 2 using GP-based acquisition. The function $\text{Split}(S, \theta')$ partitions the given region S into $2^{|\theta|}$ new regions that all share θ' as a vertex.

4 RISK ESTIMATION

In previous sections, we explored how surrogate models and conformal inference can be used to obtain probabilistic guarantees on the behavior of black-box cyber-physical system models. A different way of obtaining high-confidence probabilistic statements about correctness of systems that has recently emerged is based on the idea of *risk-estimation* [4, 12, 33, 34]. Such a correctness statement provided by risk estimation is quantitative and takes the uncertainty of the behaviors of the system into account. A risk measure is simply a function g that maps a scalar random variable X to a real value $g(X)$. Typically, the random variable X may represent an observed state of the system, where the probability distribution of X may depend on particular decision parameters for the system. The distribution of X allows us to estimate how risky a particular set of system's design parameters are, and $g(X)$ is a function that can be thought of as an empirical estimate of the magnitude of risk at a given confidence threshold ε. For example, consider the *risk-aware verification* problem considered in [4]. Here, Akella et al. treat the robustness of a given STL formula with respect to a given model trajectory as a scalar random variable. The probability distribution of this random variable is induced by the uncertainty in the initial states and parameters of the model.

A popular risk measure is known as **Value-at-Risk (VaR)**. In risk-aware verification, given a risk level $\varepsilon \in [0, 1]$, Value-at-Risk level[2] ε computes the value ρ^* s.t. the probability of the robustness for a trajectory being less than ρ^* is greater than ε. Clearly, this is yet another way to obtain probabilistic guarantees on a model's behavior through simulations (which can be used to obtain empirical estimates of the probability distribution of the desired quantity, e.g., robustness w.r.t. a given formula). In what follows, we formally recap two important tail risk measures that are commonly used and compare the bounds we obtain through risk estimation with those obtained through conformal inference.

4.1 Risk Measures

The behavior of a given CPS application can vary with the values of the system parameters. If we assume that the parameter values of a system are *a priori* unknown, then we can consider the system behavior as uncertain – where the uncertainty is induced by the distribution on the parameter values. Risk measures can then be used to quantify if the system is safe with a given probability threshold. We assume that the parameter value θ and $\rho(\varphi, \xi_\theta)$ follow a joint (unknown) distribution $\mathcal{D}_{\theta,\rho(\varphi)}$. A risk measure r_ε can provide the following probabilistic guarantee about the robustness of the system, given an STL specification and a confidence level ε:

$$\Pr\left(-\rho(\varphi, \xi_\theta) \leq r_\varepsilon\right) \geq \varepsilon. \tag{4.1}$$

We now include two important risk measures used in the literature [33].

Definition 4.1 (Value-at-Risk (VaR), Conditional-Value-at-Risk (CVaR) [33]). Let Z be shorthand for $\rho(\varphi, \xi_\theta)$. The Value-at-Risk is defined as follows:

$$VaR_\varepsilon(-Z) = \inf_{\zeta \in \mathbb{R}} \left\{\zeta \mid \Pr(-Z \leq \zeta) \geq \varepsilon\right\}. \tag{4.2}$$

The conditional-value-at-risk is defined as follows:

$$CVaR_\varepsilon(-Z) = \mathop{\mathbb{E}}_{-Z \geq VaR_\varepsilon(-Z)}(-Z) \tag{4.3}$$

Essentially, both risk measures provide probabilistic upper bounds on the negative of the robustness value, or provide lower bounds on the actual robustness value, as is required in risk-aware verification [4, 33]. Figure 3 demonstrates the relationship between the two risk measures.

To compare with the bounds provided by conformal inference, we also need to compute probabilistic guarantees on upper bounds on the robustness. These can be simply given by the risk measures $VaR_\varepsilon(\rho(\varphi, \xi_\theta))$ and $CVaR_\varepsilon(\rho(\varphi, \xi_\theta))$. For brevity, we refer to $VaR_\varepsilon(-\rho(\varphi, \xi_\theta))$ as VaR^ℓ_ε and $VaR_\varepsilon(\rho(\varphi, \xi_\theta))$ as VaR^u_ε. We will use similar notation $CVaR^\ell_\varepsilon$ and $CVaR^u_\varepsilon$.

4.2 Computing Risk Against Different System Parameters

Algorithm 2 returns regions of space that satisfy or do not satisfy a given STL formula. We now investigate whether conformal inference and risk measures will give the same conclusion on the safety of a region. We remark that the system under consideration is deterministic, but the random choice of the model parameter values θ induces a distribution of the robustness values $\rho(\varphi, \xi_\theta)$ of the given STL formula φ. Thus, the probability appearing in Equation (4.2) is associated with this distribution.

[2]Our definition is slightly different from the one in [4] and is consistent with the definitions in [41], [42], and [51]. The main difference is that in [4], the authors denote VaR at level ε to denote $\inf_\zeta P(x < \zeta) > 1 - \varepsilon$, while the probability threshold in our technique is ε.

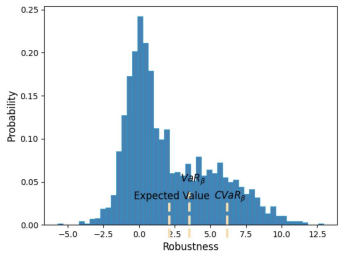


Fig. 3. Illustration of the Value-at-Risk, and the Conditional Value-at-Risk with $\varepsilon = 0.7$.

Estimation of *VaR*, *CVaR*. The definition of VaR_ε presumes that we know the joint probability distribution of $\rho(\varphi, \xi_\theta)$ and θ. However, in our problem setting, this distribution is unknown. We use the $(100 * \varepsilon)$-percentile of the samples values to approximate the VaR_ε [24].

To estimate $CVaR_\varepsilon$, we note that $CVaR_\varepsilon$ can be rewritten as the following integral:

$$CVaR_\varepsilon(-Z) = \int_0^\varepsilon VaR_\gamma(-Z)d\gamma \tag{4.4}$$

The value of the above integral can be estimated using standard Monte Carlo integration by randomly sampling the values of γ.

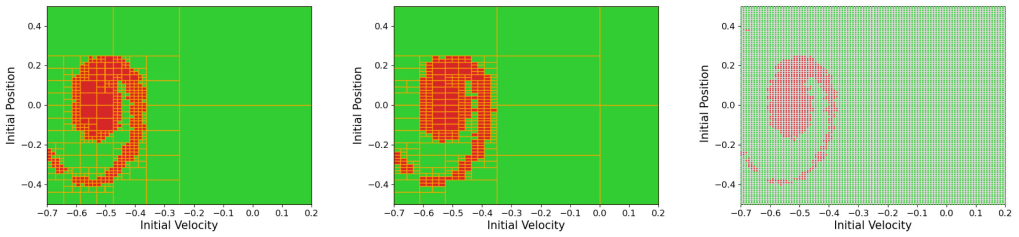
5 CASE STUDIES

In this section, we present case studies of CPS models, and identify regions in the parameter space that we can mark as safe, unsafe or unknown with high probability. We tried each of the case studies with different regression algorithms, with Gaussian Process regression leading to smaller residuals, ergo, narrower conformal intervals. We tried both (a) the naïve algorithm that recursively splits the parameter space, and (b) the algorithm which adaptively partitions the parameter space exploiting the uncertainty as expressed by a Gaussian Process prior. For all case studies, we used a miscoverage level of $\epsilon = 0.05$ (i.e., providing a correctness threshold of 95% probability).

We first compared the performance of Algorithm 2 while using different partition splitting methods. For the GP-based partitioning method, we use the *sum kernel* $k(\theta_i, \theta_j) = k_1(\theta_i, \theta_j) + k_2(\theta_i, \theta_j)$, where k_1 is the dot product kernel, i.e., $k_1(\theta_i, \theta_j) = \sigma_0^2 + \theta_i \cdot \theta_j$, and k_2 is the white kernel, where $k_2(\theta_i, \theta_j) = 1$ if $\theta_i = \theta_j$ and 0 otherwise.

Comparing Partitioning Schemes using Mountain Car. For the comparison experiment, we used a model known as mountain car popular in the reinforcement learning literature [56]. Here, the model describes an under-powered car attempting to drive up a hill. A successful strategy involves the car accumulating potential energy by going in the opposite direction and then use the gained momentum. Details of this model can be found in [56]. The parameter space for mountain car is defined by the initial position x_{init} and velocity v_{init} of the car. We wish to identify regions of space that satisfy or violate the property of reaching the goal. The region we choose for analysis is defined as as $\Theta = (x_{\text{init}}, v_{\text{init}}) \in [-0.7, 0.2] \times [-0.5, 0.5]$, which is comparable to the region used in [56]. We consider the parameter value safe if it satisfies the STL formula $\varphi_{\text{MC}} = \mathbf{F}_{[0,10]}(x(t) > 0.45)$. In Figure 4(c), we show an approximation of the ground truth obtained by a uniform grid sampling of the parameter space[3]; here, green and red dots, respectively, denote satisfaction and violation of φ_{MC}.

[3]Note that the number of grid samples used to generate the approximate ground truth far exceeds the number of simulations required for the experiments, and is only provided to enable validation of our results.



(a) Partitioning using naïve split ($\delta_{\min} = 0.02$)

(b) Partitioning using GP-based split ($\delta_{\min} = 0.02$)

(c) Mountain Car Ground Truth

Fig. 4. Mountain Car parameter space partitioning using different approaches

Table 1. Comparison of Algorithm 2 for Different Partitioning Strategies

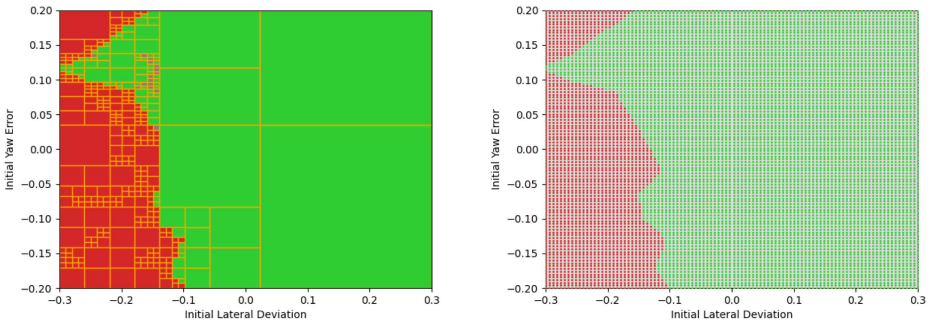
`Partition` method	num. regions explored	Ratio of Volumes (%)		
		Safe	Unsafe	Unk.
Naïve (Section 3.3)	457	89.01	10.99	0.00
Greatest uncertainty (Section 3.4)	364	88.72	11.28	0.00

$(1 - \epsilon) = 0.95$. Caption here $(1 - \epsilon)$ got splitted into a different line.

Results of comparing the naïve splitting method (Figure 4(a)) and GP-based partitioning (Figure 4(b)) are shown in Table 1. We note that the number of regions explored is much lower than the one with naïve splitting. As fewer number of regions explored translates into fewer number of simulations, it is clear that the GP-based method has superior performance. We observed similar results for other case studies in this article, but we skip the results for brevity. Due to the superiority of the GP-based method, we use this method for rest of the case studies in this article.

. *Reinforcement Learning Lane-Keep Assist.* **Lane-keep assist (LKA)** is an automated driver assistance technique used in semi-autonomous vehicles to keep the ego vehicle traveling along the centerline of a lane. We consider a reinforcement learning (RL)-based agent to perform LKA from the Matlab® RL toolbox (based on [37]). The agent has a Deep Q-Network (DQN) inside, which makes this case study a learning-enabled application. The inputs to the agent are lateral deviation e_1, relative yaw angle (i.e., yaw error) e_2, their derivatives and their integrals. The parameter space for this model consists of initial values for e_1 and e_2, where we looked at region $\Theta = (e_1, e_2) \in [-0.3, 0.3] \times [-0.2, 0.2]$. We are interested in checking properties such as overshoot/undershoot bounds and the settling time for the lateral deviation and yaw error signals. In this experiment, we consider two properties characterizing bounds on e_2 and settling time for e_1; $\varphi_{\text{LKA, settle}} : \mathbf{G}_{[2,15]}(|e_1| < 0.025)$ and $\varphi_{\text{LKA, bounds}} : \mathbf{G}_{[0,15]}(e_2 < 0.4 \wedge e_2 > -0.4)$. Figure 5 shows the parameter space partitioning results and the ground truth with respect to $\varphi_{\text{LKA, settle}}$. Our technique was able to certify that $\varphi_{\text{LKA, bounds}}$ is satisfied by the entire region with 95% confidence.

F-16 Control System. Next, we consider the verification challenge presented in [23]. This is the model of a F-16 flight control system—a hierarchical control system containing an outer-loop autopilot and an **inner-loop tracking and stabilizing controller (ILC)**, and a 13 dimensional nonlinear dynamical plant model. The plant dynamics are based on a 6 degrees of freedom standard airplane model [48] represented by a system of 13 ODEs describing the force equations, kinematics, moments, and a first-order lag model for the afterburning turbofan engine. These ODEs describe the evolution of the system states, namely velocity vt, angle of attack α, sideslip β, altitude h, attitude angles: roll ϕ, pitch θ, yaw ψ, and their corresponding rates p, q, r, engine *power* and two more states for translation along north and east. The non-linear plant model uses linearly interpolated lookup tables to incorporate wind tunnel data. The control system is composed of an autopilot



(a) Partitioning using greatest uncertainty split ($\delta_{\min} = 0.01$)

(b) LKA (settle) Ground Truth

Fig. 5. Lane Keep Assist.

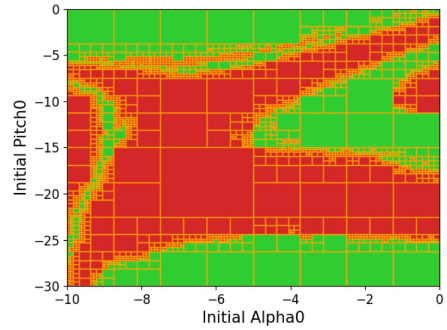
that sets the references on upward acceleration, stability roll rate and the *throttle*. The ILC uses an LQR state feedback law to track the references and computes the control input for the *aileron*, *rudder* and the *elevator*. We consider three separate scenarios capturing specific contexts; each scenario defines the parameter set and an associated specification.

F16-Pull-Up Maneuver. This scenario demonstrates the tracking of a constant autopilot command requesting an upward acceleration ($N_z = 5g$). The ILC tries to track the reference without undesirable transients like pitch oscillations and exceeding pitch rate limits. We modify the controller gains to highlight the violations of the spec $\varphi_{\text{F16,PULLUP}} : \mathbf{G}_{[0,10]} q \leq 120^\circ/s$. The parameter space is described by initial values of $\alpha \in [-10^\circ, 0^\circ]$, $\theta \in [-30^\circ, 0^\circ]$ and the results are shown in Figure 6(a) (results of Algorithm 2) and Figure 6(b) (ground truth).

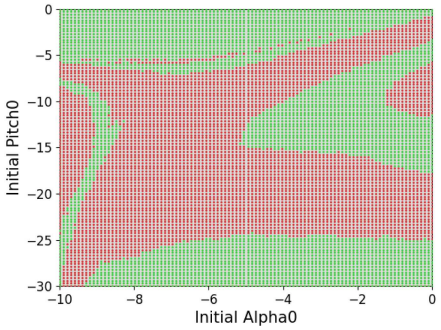
F16-Level Flight. This scenario describes straight and level flight with a constant attitude and 0 initial angular rates. The bounded parameter space is defined by the initial altitude $h \in [500, 65000]$ and velocity $vt \in [130, 1200]$. The autopilot references are set to zero, and the ILC tries to maintain a constant altitude and angle of attack α. As the F-16 can fly over a large range of altitudes and velocities, a single LQR computed against the linearzied model can not satisfy the goal and results in a stall defined by $\varphi_{\text{F16,LEVEL}} : \mathbf{G}_{[0,10]}(\alpha \leq 35^\circ)$. This is shown Figure 6.

F16-Ground Collision Avoidance (GCAS). The final scenario describes the F-16 diving towards the ground and the GCAS autopilot trying to prevent the collision. The GCAS brings the roll angle and its rate to 0 and then accelerates upwards to avoid ground collision as defined by the spec $\varphi_{\text{F16,GCAS}} : \mathbf{G}_{[0,10]}(h \geq 0ft)$. The parameter space is described by initial values of $\alpha \in [0.075, 0.1]^c$ and $\varphi \in [-0.1, -0.075]^c$. In this case study, the ground truth and our results seem to be less well-matched than other case studies. There are a couple of reasons for this. First, observe that the ground truth is highly non-monotonic. Given the nonlinearity of the ground truth, the fitted surrogate model could tend to fit the value of the majority of the points better, which in this case are negative values, making the optimization results errs on the negative values and resulting in a region being marked unsafe. To remedy this, we would have to increase the number of simulations per region used to train the GP regression model and possibly experiment with other regression models (such as a deep neural network regressor). (We provide an illustration of the results in the Appendix in Figure A.1(c) and A.1(d).)

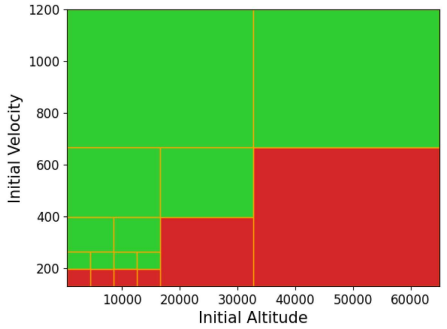
Artificial Pasncreas. Type-1 diabetes (juvenile diabetes) is a chronic condition caused by the inability of the pancreas to secrete the required amount of insulin. Simglucose [54] is a Python implementation of the FDA-approved Type-1 Diabetes simulator [36] which models glucose kinetics. We input a list of tuples of time and meal size to Simglucose and set the same scenario environment.



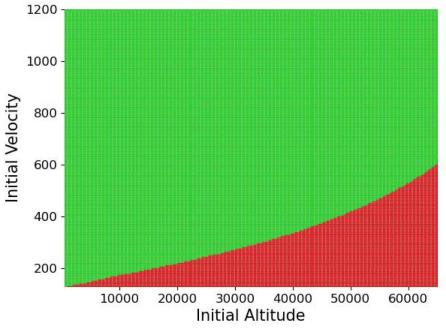
(a) Partitioning using greatest uncertainty split ($\delta_{\min} = 0.1$)



(b) Ground truth



(c) Partitioning using greatest uncertainty split ($\delta_{\min} = 100$)



(d) Ground truth

Fig. 6. F16 - Pull up (Top) Level Flight (bottom).

Choosing patients in different age will result in different simulation trace. The parameter *meal time* is constrained to be strictly increasing and the last meal of a day to be taken in less than 24 hours. For each scenario, the simulator provides traces records of different blood indicators based on a given environment setting. We are interested in checking if patients do not become hyperglycemic on the first day (i.e., when the blood glucose (BG) exceeds a certain threshold). We use $\delta = 0.5$ as the termination criteria for region splitting. We study 4 scenarios describing an adolescent patient who takes 2, 3, 4, and 5 meals a day, respectively. The meals of size s_i are consumed at time t_i. The parameters space is then defined by $S_1 \times S_2 \times \cdots \times S_n$, the dimension of the parameter space equals the number of meals taken. We denote n as the total number of meals. We can calculate t_i using equation $t_i = [(i-1) * \frac{24}{n} + 1, i * \frac{24}{n}]$, and $S_i \subseteq [1, 20]$.

The property $\varphi_{\text{hyper},c}$ specifies that the patient should not become hyperglycemic. Our results predict the entire region as 100% safe region with 95% confidence for all cases where the patient had 3 or more meals. For the two meal case for the property $\varphi_{\text{hyper},155}$ our implementation result of 54.44% matches well with the ground truth where we see a 52.47% unsafe volume (obtained by expensive grid-based sampling). For higher dimensional cases, the initial S_i considered are different. This could imply more frequent meals with less amounts each can help control blood glucose.

Impact and Discussion of Results. The results obtained for all the case studies are summarized in Table 2. Our tool is capable of producing heatmap style representation of the unsafe parameter

Table 2. Performance of Algorithm 2 using the GP-based Greatest Uncertainty Split Method with 95% Confidence Level

Case Study	Ratio of Volumes (%)			Sims./ region	Spec.
	Safe	Unsafe	Unk.		
Mountain Car 1	88.72	11.28	0.00	100	φ_{MC}
Lane Keep Assist 1	100	0.00	0.00	100	$\varphi_{\text{LKA, bounds}}$
Lane Keep Assist 2	77.23	21.97	0.80	100	$\varphi_{\text{LKA, settle}}$
F16 Level Flight	67.18	32.81	0.00	100	$\varphi_{\text{F16,LEVEL}}$
F16 Pull up	43.52	56.09	0.40	100	$\varphi_{\text{F16,PULLUP}}$
F16 GCAS	3.91	96.09	0.00	100	$\varphi_{\text{F16,GCAS}}$
Simglucose 2D	45.45	54.55	0.00	10	$\varphi_{\text{hyper, 155}}$
Simglucose 2D	100	0.00	0.00	10	$\varphi_{\text{hyper, 170}}$
Simglucose 3D	100	0.00	0.00	10	$\varphi_{\text{hyper, 155}}$
Simglucose 4D	100	0.00	0.00	10	$\varphi_{\text{hyper, 155}}$
Simglucose 5D	100	0.00	0.00	10	$\varphi_{\text{hyper, 155}}$

regions when projected to two parameter dimensions. For higher number of parameter dimensions, visualization is more difficult. Hence, we also report the percentage volume of regions found safe or unsafe by our method. We observe that, in most cases, the volume of regions that remain unknown is quite low. As some of the case studies are those of learning-enabled CPS applications, it is expected to see a high volume of unsafe regions—this can happen if the **learning-enabled components (LECs)** are effectively trained in all parameter regions. Thus, our tool can provide useful information to algorithms for *training* such LECs.

We remark that the runtime for our method is dominated by the time required for running the simulations—a step that is easily parallelizable. We can also reuse simulations performed on a given sub-region of a coarser region when the region is split. Our prototype tool also does not include either of these optimizations. The time required for training the GP surrogate with 100 data points takes 0.035 seconds on an average. The naïve refinement procedure takes around 7.1 μs for models with 2D parameter spaces and 24 μs for 3D parameter spaces. With GP-based refinement (which requires the use of optimization with acquisition functions), the runtime is 0.006 seconds. Thus, with the ability to parallelize and reuse simulations, the additional overhead induced by our method (e.g., in comparison to an SMC method) is minimal. We do acknowledge that SMC methods can perhaps obtain guarantees with a fewer number of simulations using statistical hypothesis testing; however, SMC methods typically do not learn surrogate models and cannot generate parameter space partitioning. We finally remark that if the model parameters are being chosen by an outer loop supervisory control, then the partitions that we generate create conditional contracts on the safety of the CPS model; such contracts can be used for constructing safety assurance cases [46].

5.1 Comparing with Risk Measures

Figures 7-9 demonstrate the VaR_ε and $CVaR_\varepsilon$ risk assessment for each region. For the same confidence level $\varepsilon = 95\%$, the conformal inference procedure computes $[v_{\min}, v_{\max}]$ as the bounds on the robustness value, while with a given risk measure r_ε, let $[r^\ell, r^u]$ indicate the (probabilistic) lower and upper bounds on the robustness values. Recall that, if $v_{\max} < 0$, then note the region is red; and if $v_{\min} > 0$, the region is green. We use the following color coding:

(1) *For red regions*: If $v_{\max} < r^u < 0$, then we use a lighter shade of red, and if $0 > v_{\max} > r^u$, we use a darker shade. Intuitively, a lighter shade indicates that the risk-based probability

Table 3. Number of CVaR and VaR Not Bounded by the upper and Lower Bound of Conformal Inference, with $-\rho$ as Loss Function and 95% as Confidence Level

Sims./ region	CVaR Lower	CVaR Higher	Diff	# Region	VaR Lower	VaR Higher	Diff	# Region
5	17 (54.84%)	0 (0.0%)	3 (9.68%)	31	17 (60.71%)	2 (7.14%)	1(3.57%)	28
10	34 (55.74%)	7 (11.48%)	3 (4.92 %)	61	26 (65.0%)	2(5.0%)	6(15.0%)	40
50	148 (62.18%))	0 (0.0%)	24 (10.08%)	238	158 (55.83%)	1 (0.35%)	17 (6.01%)	283
100	879 (35.37%)	1(0.04%)	61(2.45%)	2485	647 (27.79%)	0 (0.0%)	30 (1.24%)	2425
200	830 (36.24%)	3 (0.13%)	50 (2.18%)	2290	637 (28.0%)	0 (0.0%)	15 (0.66%)	2275
300	801 (33.61%)	2 (0.08%)	50 (2.10%)	2383	612 (27.70%)	2 (0.09%)	13 (0.59%)	2209
500	757 (32.50%)	2 (0.09%)	34 (1.46%)	2329	592 (25.19%)	6 (0.26%)	10 (0.43%)	2350
1000	739 (31.73%)	8 (0.34%)	35 (1.50%)	2329	579 (24.21 %)	11 (0.46%)	7 (0.29%)	2392

measures deem the region "less unsafe" (as compared to the bounds computed by the split conformal predictor), and a darker shade indicates riskier or more unsafe regions (for the same comparison).

(2) *For green regions*: If $r^\ell > v_{\min} > 0$, then we use a lighter shade of green, and if $0 < r^\ell < v_{\min}$, we use a darker shade. Intuitively, a darker shade indicates that the risk-based probability measures deem the region "less safe", and a lighter shade indicates a more robustly safe region (according to risk estimation).

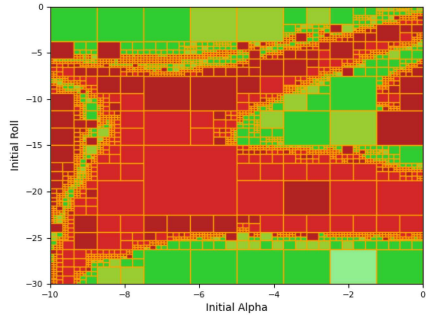
(3) *For blue regions*: If $v_{\max} < 0 < r^u$ (i.e., the conformal bound deems the region unsafe, but the risk measure either deems the region safe or inconclusive), then we color the region light blue. If $r^\ell < 0 < v_{\min}$ (i.e., the conformal bound deems the region safe, but the risk measure either deems it unsafe or inconclusive), then we color the region dark blue. Blue regions indicate that the two methods are unable to agree on the classification of a region as safe or unsafe.

In conclusion, conformal inference bounds are not guaranteed to bound the risk (as computed by using risk measures). However, risk measures largely agree with conformal inference on region safety. Table 3 and Table 4 show risk measure performance compared to the bounds computed by conformal inference as a function of the number of samples used per region to compute either kind of bounds. The column labeled "Diff" counts regions where the conformal inference is: (1) not assigned unknown to the region, and (2) the risk measure has a different conclusion about the safety of that region. We can see that the maximum disagreement is still less than 5%, and the two methods tend to agree more when we sample more per region. We remark that a surrogate model's training set could still miss rare unsafe points due to sampling vagaries, so the surrogate model can provide an optimistic $v_{\min}$, which may mark some regions containing unsafe points as safe (because the regression-based surrogate model lacks some critical data points). However, risk-based analysis directly does quantile estimation on the robustness values (without fitting a surrogate model), and hence, in some cases, may help (conservatively) label regions as unsafe.

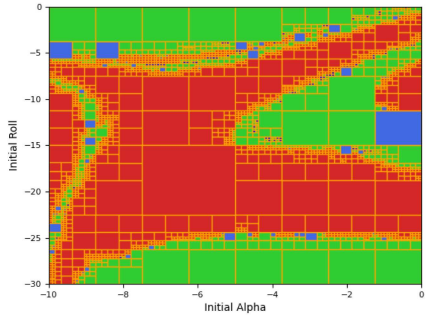
6 RELATED WORK AND CONCLUSIONS

Related Work. Methods based on **Statistical Model Checking (SMC)** [29, 30, 57] can overcome the hurdles like scalability and nonlinearity and provide probabilistic guarantees [1, 13, 45, 53, 56]. These methods are based on statistical inference methods like sequential probability ratio tests [13, 29, 45, 47], Bayesian statistics [57], and Clopper-Pearson bounds [56]. Another line of works use **Probably Approximately Correct (PAC)** learning theory to give probabilistic bounds for Markov decision processes and black-box systems [19, 21].

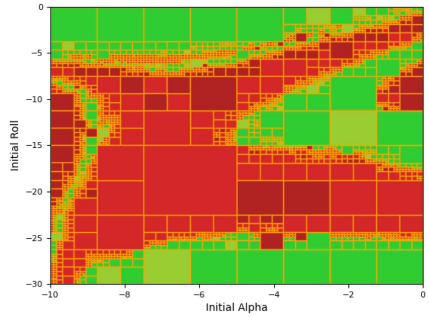
In contrast to SMC- and PAC-learning techniques, our approach is sample independent and can provide the required probabilistic guarantees with any number of samples. This is because we build



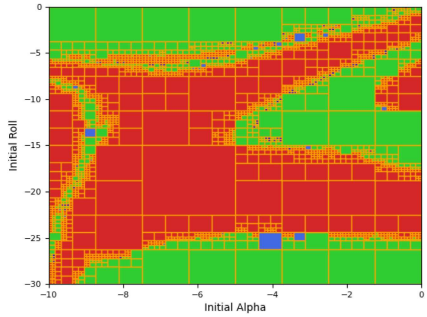
(a) CVaR with $-\rho$ as loss. Partitioning using greatest uncertainty split ($\delta_{\min} = 0.1$)



(b) Comparing Conformal Inference and CVaR. Regions with the color blue indicate they have different conclusions on the safety of the region.



(c) VaR with $-\rho$ as loss. Partitioning using greatest uncertainty split ($\delta_{\min} = 0.1$)



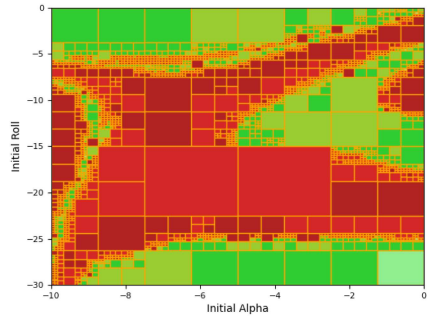
(d) Comparing Conformal Inference and VaR. Regions with the color blue indicate they have different conclusions on the safety of the region.

Fig. 7. F16 - Pull up. 100 samples per region.

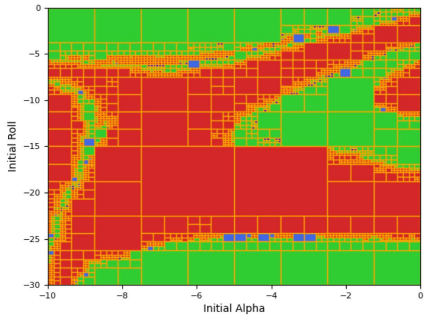
Table 4. Number of CVaR and VaR Not Bounded by the Upper and Lower Bound of Conformal Inference, with ρ as Loss Function and 95% as Confidence Level

Sim./ region	CVaR Lower	CVaR Higher	Diff	# Region	VaR Lower	VaR Higher	Diff	# Region
5	0 (0.0%)	10 (62.5%)	9(56.25%)	16	0 (0.0%)	5 (71.43%)	5 (71.43%)	7
10	0 (0.0%)	15 (48.39%)	9 (29.03%)	31	1 (3.57 %)	16 (57.14%)	13 (46.43%)	28
50	0 (0.0%)	97 (62.99%)	40 (25.97%)	154	1 (1.06%)	57 (60.64%)	26 (27.66%)	94
100	1 (0.04%)	37 (1.51%)	1 (0.04%)	2458	8 (0.34%)	1 (0.04%)	0 (0.0%)	2350
200	3 (0.13%)	150 (6.34%)	8 (0.34%)	2365	5 (0.21%)	30 (1.28%)	4 (0.17%)	2338
300	3 (0.13%)	169 (7.16%)	7 (0.30%)	2359	9 (0.37%)	27 (1.10%)	0 (0.0%)	2446
500	5 (0.21%)	208 (8.62%)	4 (0.17%)	2413	8 (0.35%)	48 (2.09%)	1 (0.04%)	2293
1000	16 (0.70%)	223 (9.80%)	14 (0.62%)	2275	17 (0.74%)	60 (2.61%)	2 (0.09%)	2299

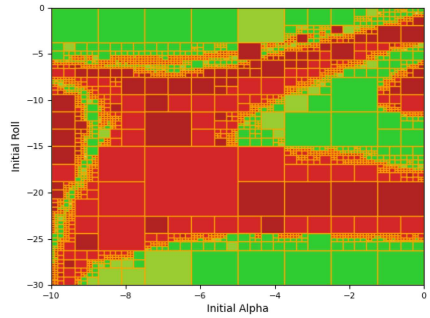
a guaranteed regression model from the system parameters with respect to the robust satisfaction value of the corresponding STL properties. If the regression model is of poor quality (due to few samples), using the calibration step in conformal regression, the predicted (but wider) interval can still have the same level of guarantee. Conformal regression lets us trade off the quality of the regression model (w.r.t. the data) and the width of the interval for which we have high-confidence property satisfaction, and **not** the level of the guarantee itself.



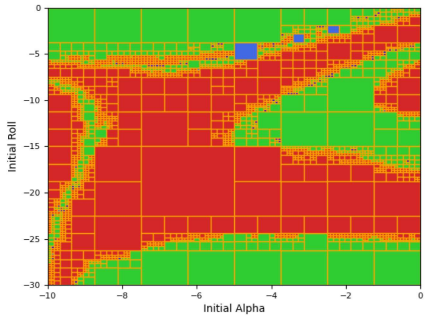
(a) CVaR with $-\rho$ as loss. Partitioning using greatest uncertainty split ($\delta_{\min} = 0.1$)



(b) Comparing Conformal Inference and CVaR. Regions with the color blue indicate they have different conclusions on the safety of the region.



(c) VaR with $-\rho$ as loss. Partitioning using greatest uncertainty split ($\delta_{\min} = 0.1$)

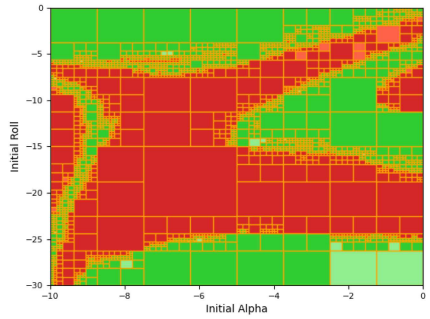


(d) Comparing Conformal Inference and VaR. Regions with the color blue indicate they have different conclusions on the safety of the region.

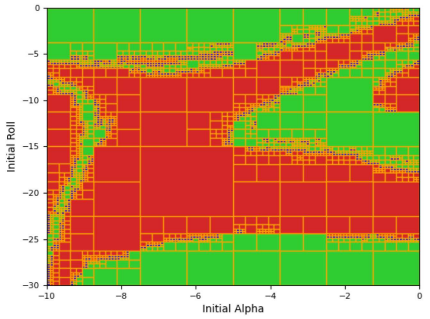
Fig. 8. F16 - Pull up. With more samples (1000) per region.

Recent work on using conformity measures is quite relevant to our work [8, 9]; the main contribution here is that, in order to handle high-dimensional inputs in real-time, the authors compute a nonconformity score using an embedding representation of deep neural network models. This work however focuses on a classification problem and not on obtaining probabilistic guarantees on (closed-loop) system correctness. The idea of obtaining trusted confidence bounds is however similar, albeit applied in a different context. The work in [10] and [22] focuses on detecting regions of the input space of a learning-enabled component that lack training data and hence can potentially have large (prediction) errors. These approaches are more suitable for runtime assurance or statically characterizing uncertainty in predictions performed by learning-enabled components. In our article, we use regression-style learning algorithms to approximate the model itself and use such surrogate models to obtain probabilistic guarantees. While the models we consider may themselves have **learning enabled components (LECs)**, our approach is black-box: it does not reason about the LECs themselves.

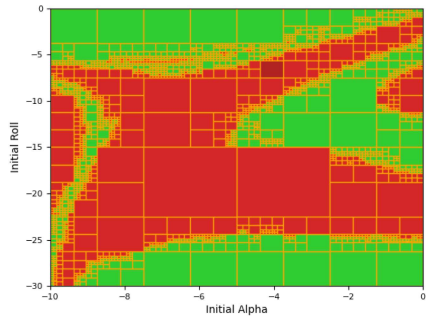
Risk is an excellent way to analyze the robustness of systems. Control design sees risk as essential to calculate and optimize during the process. Works on stochastic system verification find risk measures give more information than just expectation values of cost. We perform a detailed comparison of our work in [39] with the guarantees obtained using risk measures. We see that the



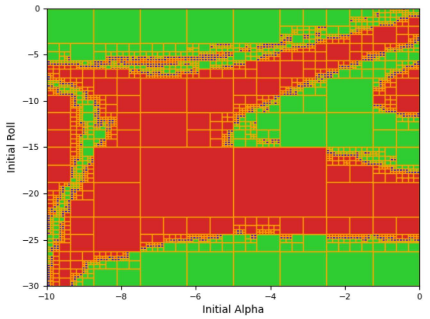
(a) CVaR with ρ as loss. Partitioning using greatest uncertainty split ($\delta_{\min} = 0.1$)



(b) Comparing Conformal Inference and CVaR. Regions with the color blue indicate they have different conclusions on the safety of the region.



(c) VaR with ρ as loss and Partitioning using greatest uncertainty split ($\delta_{\min} = 0.1$)



(d) Comparing Conformal Inference and VaR. Regions with the color blue indicate they have different conclusions on the safety of the region.

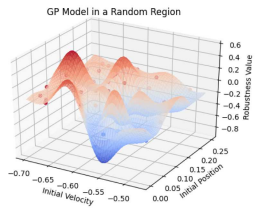
Fig. 9. F16 - Pull up 100 samples per region.

verification results of our proposed methods agree with each other in most cases. We argue that when the two guarantees do not agree, regions where such disagreements occur would require further investigation. A potential reason for mismatch is that the empirical distribution of residual error (when training the surrogate model) may differ from the distribution of robustness values in this region (due to non-linearities in the CPS model's dynamics, how well the chosen surrogate model fits the data, etc.).

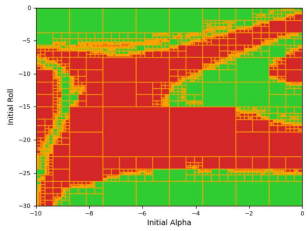
7 CONCLUSION

In this article, we proposed a verification framework that can search the parameter space to find the regions that lead to satisfaction or violation of given specification with probabilistic coverage guarantees. There are a couple of directions we aim to explore as future work: (1) We used a very basic version of conformal regression in Algorithm 1, which gives a constant confidence range d across all X. Techniques based on quantile regression [44] and locally weighed conformal [31] can make d a function of X and give much shorter prediction intervals and (2) We plan to explore probabilistic regret bounds for Gaussian process optimization to help obtain (probabilistic) upper and lower bounds on the value of the surrogate model when using GP-based regression.

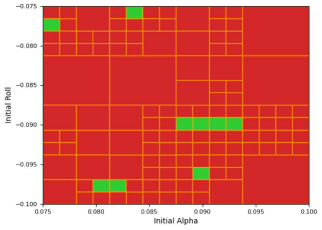
APPENDIX



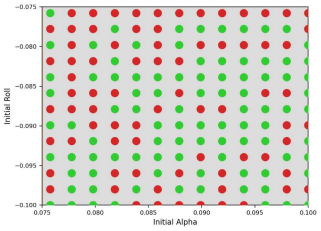
(a) Gaussian Process Regression Visualization for the Mountain Car model



(b) F16 - Pull up 100 samples per region with smaller $\delta_{min} = 0.001$, partitioning using greatest uncertainty split



(c) F16 - Ground Collision Avoidance Parameter space partitioning obtained using greatest uncertainty split ($\delta_{\min}$=0.002)



(d) GCAS Ground truth

Fig. A.1. Experimental results for the F16 case study.

REFERENCES

[1] H. Abbas, B. Hoxha, G. Fainekos, and K. Ueda. 2014. Robustness-guided temporal logic testing and verification for stochastic cyber-physical systems. In *4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*. IEEE, 1–6.

[2] Gul Agha and Karl Palmskog. 2018. A survey of statistical model checking. *ACM Transactions on Modeling Computing and Simululation* 28, 1 (2018), 6:1–6:39. https://doi.org/10.1145/3158668

[3] Takumi Akazaki and Ichiro Hasuo. 2015. Time robustness in MTL and expressivity in hybrid system falsification. In *CAV*. 356–374.

[4] Prithvi Akella, Mohamadreza Ahmadi, and Aaron D. Ames. 2022. A Scenario Approach to Risk-Aware Safety-Critical System Verification. (2022). https://doi.org/10.48550/ARXIV.2203.02595

[5] Andrew R. Barron. 1993. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory* 39, 3 (1993), 930–945.

[6] Ezio Bartocci, Thomas Ferrère, Niveditha Manjunath, and Dejan Ničković. 2018. Localizing faults in simulink/stateflow models with STL. In *Proc. of HSCC*. 197–206.

[7] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

[8] Dimitrios Boursinos and Xenofon Koutsoukos. 2020. Assurance monitoring of cyber-physical systems with machine learning components. *arXiv preprint arXiv:2001.05014* (2020).

[9] Dimitrios Boursinos and Xenofon Koutsoukos. 2020. Trusted confidence bounds for learning enabled cyber-physical systems. In *2020 IEEE Security and Privacy Workshops (SPW)*. 228–233. https://doi.org/10.1109/SPW50608.2020.00053

[10] Feiyang Cai and Xenofon Koutsoukos. 2020. Real-time out-of-distribution detection in learning-enabled cyber-physical systems. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 174–183.

[11] Pavol Cerny, Thomas A. Henzinger, and Arjun Radhakrishna. 2013. Quantitative abstraction refinement. In *40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. 115–128.

[12] Margaret P. Chapman, Jonathan Lacotte, Aviv Tamar, Donggun Lee, Kevin M. Smith, Victoria Cheng, Jaime F. Fisac, Susmit Jha, Marco Pavone, and Claire J. Tomlin. 2019. A risk-sensitive finite-time reachability approach for safety of stochastic dynamic systems. In *2019 American Control Conference (ACC)*. 2958–2963. https://doi.org/10.23919/ACC.2019.8815169

[13] E. M. Clarke, J. R. Faeder, C. J. Langmead, L. A. Harris, S. K. Jha, and A. Legay. 2008. Statistical model checking in biolab: Applications to the automated analysis of t-cell receptor signaling pathway. In *CMSB*. Springer, 231–250.

[14] Jyotirmoy Deshmukh, Xiaoqing Jin, Rupak Majumdar, and Vinayak Prabhu. 2018. Parameter optimization in control software using statistical fault localization techniques. In *ICCPS*. IEEE, 220–231.

[15] Alexandre Donzé and Oded Maler. 2010. Robust satisfaction of temporal logic over real-valued signals. In *FORMATS*. Springer, 92–106.

[16] Tommaso Dreossi, Daniel J. Fremont, Shromona Ghosh, Edward Kim, Hadi Ravanbakhsh, Marcell Vazquez-Chanlatte, and Sanjit A. Seshia. 2019. VerifAI: A toolkit for the formal design and analysis of artificial intelligence-based systems. In *CAV*. 432–442.

[17] Souradeep Dutta, Susmit Jha, Sriram Sankaranarayanan, and Ashish Tiwari. 2018. Learning and verification of feedback control systems using feedforward neural networks. *IFAC-PapersOnLine* 51, 16 (2018), 151–156.

[18] Georgios E. Fainekos and George J. Pappas. 2009. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science* 410, 42 (2009), 4262–4291.

[19] Chuchu Fan, Bolun Qi, Sayan Mitra, and Mahesh Viswanathan. 2017. Dryvr: Data-driven verification and compositional reasoning for automotive systems. In *CAV*. 441–461.

[20] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The Elements of Statistical Learning*. Vol. 1. Springer series in statistics, New York.

[21] Jie Fu and Ufuk Topcu. 2014. Probably approximately correct MDP learning and control with temporal logic constraints. *arXiv preprint arXiv:1404.7073* (2014).

[22] Xiaozhe Gu and Arvind Easwaran. 2019. Towards safe machine learning for CPS: Infer uncertainty from training data. In *10th ACM/IEEE International Conference on Cyber-Physical Systems*. 249–258.

[23] Peter Heidlauf, Alexander Collins, Michael Bolender, and Stanley Bak. 2018. Verification challenges in F-16 ground collision avoidance and other automated maneuvers. In *ARCH@ ADHS*. 208–217.

[24] L. Jeff Hong, Zhaolin Hu, and Guangwu Liu. 2014. Monte Carlo methods for value-at-risk and conditional value-at-risk: A review. *ACM Trans. Model. Comput. Simul.* 24, 4, Article 22 (Nov 2014), 37 pages. https://doi.org/10.1145/2661631

[25] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2017. Safety verification of deep neural networks. In *International Conference on Computer Aided Verification*. Springer, 3–29.

[26] Radoslav Ivanov, James Weimer, Rajeev Alur, George J. Pappas, and Insup Lee. 2019. Verisig: Verifying safety properties of hybrid systems with neural network controllers. In *HSCC*.

[27] Stefan Jakšić, Ezio Bartocci, Radu Grosu, Thang Nguyen, and Dejan Ničković. 2018. Quantitative monitoring of STL with edit distance. *Formal Methods in System Design* 53, 1 (Aug. 2018), 83–112. https://doi.org/10.1007/s10703-018-0319-x

[28] Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. 2017. Reluplex: An efficient SMT solver for verifying deep neural networks. In *CAV*, Rupak Majumdar and Viktor Kunčak (Eds.). 97–117.

[29] Axel Legay, Benoît Delahaye, and Saddek Bensalem. 2010. Statistical model checking: An overview. In *Runtime Verification*. Springer Berlin , Berlin, , 122–135.

[30] Axel Legay and Mahesh Viswanathan. 2015. Statistical model checking: challenges and perspectives. *International Journal on Software Tools for Technology Transfer* 17 (2015), 369–376.

[31] Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. 2018. Distribution-free predictive inference for regression. *J. Amer. Statist. Assoc.* 113, 523 (2018), 1094–1111.

[32] Jing Lei and Larry Wasserman. 2014. Distribution-free prediction bands for non-parametric regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 76, 1 (2014), 71–96.

[33] Lars Lindemann, Lejun Jiang, Nikolai Matni, and George J. Pappas. 2022. Risk of stochastic systems for temporal logic specifications. (2022). https://doi.org/10.48550/ARXIV.2205.14523

[34] Anirudha Majumdar and Marco Pavone. 2020. How should a robot assess risk? Towards an axiomatic theory of risk in robotics. In *Robotics Research*, Nancy M. Amato, Greg Hager, Shawna Thomas, and Miguel Torres-Torriti (Eds.). Springer International Publishing, Cham, 75–84.

[35] Oded Maler and Dejan Nickovic. 2004. Monitoring temporal prop/hastieerties of continuous signals. *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*. Springer, 152–166.

[36] Chiara Dalla Man, Francesco Micheletto, Dayu Lv, Marc Breton, Boris Kovatchev, and Claudio Cobelli. 2014. The UVA/PADOVA type 1 diabetes simulator: New features. *Journal of Diabetes Science and Technology* 8, 1 (2014), 26–34.

[37] Mathworks 2020. Train DQN agent for Lane Keep Assist. https://www.mathworks.com/help/reinforcement-learning/ug/train-dqn-agent-for-lane-keeping-assist.html (n.d.).

[38] Giulia Pedrielli, Tanmay Khandait, Surdeep Chotaliya, Quinn Thibeault, Hao Huang, Mauricio Castillo-Effen, and Georgios Fainekos. 2021. Part-X: A family of stochastic algorithms for search-based test generation with probabilistic guarantees. *arXiv preprint arXiv:2110.10729* (2021).

[39] Xin Qin, Yuan Xia, Aditya Zutshi, Chuchu Fan, and Jyotirmoy V. Deshmukh. 2022. Statistical verification of cyber-physical systems using surrogate models and conformal inference. In *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 116–126.

[40] Carl Edward Rasmussen. 2003. Gaussian processes in machine learning. In *Summer School on Machine Learning*. Springer, 63–71.

[41] R. Tyrrell Rockafellar and Stanislav Uryasev. 2002. Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance* 26, 7 (2002), 1443–1471. https://doi.org/10.1016/S0378-4266(02)00271-6

[42] R. Tyrrell Rockafellar and Stanislav Uryasev. 2000. Optimization of conditional value-at-risk. *Journal of Risk* 2 (2000), 21–41.

[43] Alena Rodionova, Ezio Bartocci, Dejan Nickovic, and Radu Grosu. 2016. Temporal logic as filtering. In *19th International Conference on Hybrid Systems: Computation and Control (HSCC '16)*. (2016), 11–20. arXiv:1510.08079

[44] Yaniv Romano, Evan Patterson, and Emmanuel Candes. 2019. Conformalized quantile regression. In *NeurIPS*. 3538–3548.

[45] Nima Roohi, Yu Wang, Matthew West, Geir E. Dullerud, and Mahesh Viswanathan. 2017. Statistical verification of the Toyota powertrain control verification benchmark. In *20th International Conference on Hybrid Systems: Computation and Control (HSCC '17)*. ACM, New York, , 65–70. https://doi.org/10.1145/3049797.3049804

[46] John Rushby. 2002. Partitioning for safety and security: Requirements, mechanisms, and assurance. *AFRL-IF-RS-TR'-2002-85* (2002), 9.

[47] Koushik Sen, Mahesh Viswanathan, and Gul Agha. 2004. Statistical model checking of black-box probabilistic systems. In *Computer Aided Verification*, Rajeev Alur and Doron A. Peled (Eds.). Springer Berlin , Berlin, , 202–215.

[48] Brian L. Stevens, Frank L. Lewis, and Eric N. Johnson. 2015. *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*. John Wiley & Sons.

[49] Xiaowu Sun, Haitham Khedr, and Yasser Shoukry. 2019. Formal verification of neural network controlled autonomous systems. In *ACM International Conference on Hybrid Systems: Computation and Control (HSCC'19)*. ACM, New York, , 147–156. https://doi.org/10.1145/3302504.3311802

[50] Hoang-Dung Tran, Diago Manzanas Lopez, Patrick Musau, Xiaodong Yang, Luan Viet Nguyen, Weiming Xiang, and Taylor T. Johnson. 2019. Star-based reachability analysis of deep neural networks. In *Formal Methods – The Next 30 Years*, Maurice H. ter Beek, Annabelle McIver, and José N. Oliveira (Eds.). Springer International Publishing, Cham, 670–686.

[51] S. Uryasev. 2000. Conditional value-at-risk: Optimization algorithms and applications. In *IEEE/IAFE/INFORMS 2000 Conference on Computational Intelligence for Financial Engineering (CIFEr) (Cat. No.00TH8520)*. 49–57. https://doi.org/10.1109/CIFER.2000.844598

[52] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. 2005. *Algorithmic Learning in a Random World*. Springer Science & Business Media.

[53] Yu Wang, Mojtaba Zarei, Borzoo Bonakdarpour, and Miroslav Pajic. 2019. Statistical verification of hyperproperties for cyber-physical systems. *ACM Trans. Embed. Comput. Syst.* 18, 5s, Article Article 92 (Oct. 2019), 23 pages. https://doi.org/10.1145/3358232

[54] Jinyu Xie. 2018. Simglucose v0.2.1. https://github.com/jxx123/simglucose. (2018).

[55] Shakiba Yaghoubi and Georgios Fainekos. 2019. Gray-box adversarial testing for control systems with machine learning components. In *HSCC*. 179–184.

[56] Mojtaba Zarei, Yu Wang, and Miroslav Pajic. 2020. Statistical verification of learning-based cyber-physical systems. In *23nd ACM International Conference on Hybrid Systems: Computation and Control*.

[57] Paolo Zuliani, André Platzer, and Edmund M. Clarke. 2010. Bayesian statistical model checking with application to simulink/stateflow verification. In *HSCC*. 243–252.

Received 9 November 2022; revised 15 August 2023; accepted 14 November 2023