Gaussian Process-Based Confidence Estimation for Hybrid System Falsification

Zhenya Zhang[1(✉)] and Paolo Arcaini[2]

[1] Kyushu University, Fukuoka, Japan
[2] National Institute of Informatics, Tokyo, Japan

Abstract. Cyber-Physical Systems (CPSs) are widely adopted in safety-critical domains, raising great demands on their quality assurance. However, the application of formal verification is limited due to the continuous dynamics of CPSs. Instead, simulation-based falsification, which aims at finding a counterexample to refute the system specification, is a more feasible and hence actively pursued approach. Falsification adopts an optimization approach, treating *robustness*, given by the quantitative semantics of the specification language (usually Signal Temporal Logic (STL)), as the objective function. However, similarly to traditional testing, in the absence of found counterexamples, falsification does not give any guarantee on the system safety. To fill this gap, in this paper, we propose a *confidence measure* that estimates the probability that a formal specification is indeed not falsifiable, by relying on the information encapsulated in the simulation data collected during falsification. Methodologically, we approximate the robustness domain by feeding simulation data into a Gaussian Process (GP) Regression process; we then do a minimization sampling on the trained GP, and then estimate the probability that all the robustness values inferred from these sampled points are positive; we take this probability as the confidence measure. We experimentally study the properties of monotonicity and soundness of the proposed confidence measure. We also apply the measure to several state-of-the-art falsification algorithms to assess the maximum confidence they provide when they do not find a falsifying input, and the stability of such confidence across different repetitions.

Keywords: Confidence estimation · Hybrid system falsification · Gaussian process regression · Surrogate model

1 Introduction

Cyber-Physical Systems (CPS) are characterized by the combination of physical systems having continuous dynamics, and discrete digital controllers; for this,

Zhenya Zhang is supported by JSPS KAKENHI Grant No.20H04168, 19K24348, 19H04086, and JST-Mirai Program Grant No. JPMJMI18BB, Japan. Paolo Arcaini is supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST.

M. Huisman et al. (Eds.): FM 2021, LNCS 13047, pp. 330–348, 2021.
https://doi.org/10.1007/978-3-030-90870-6_18

they are also called *hybrid systems*. CPSs are often employed in safety-critical domains, making their quality assurance of paramount importance. However, the continuous dynamics of hybrid systems makes their automated formal verification extremely difficult, if not impossible. Therefore, academia and industry have been pursuing the more feasible approach of *falsification* [13,22,29,30,33,36,37] that, instead of trying to prove a formal specification, attempts to find a counterexample showing its violation. Specifically, given a *model* $\mathcal{M}$ taking input signal $\mathbf{u}$ and producing output signal $\mathcal{M}(\mathbf{u})$, and a formal specification φ (a temporal formula), the falsification problem consists in finding a *falsifying input*, i.e., an input signal $\mathbf{u}$ such that $\mathcal{M}(\mathbf{u})$ violates φ.

The common approach to solve the falsification problem is to turn it into an optimization problem (also called *optimization-based falsification*), by exploiting the quantitative *robust semantics* of temporal formulas [14,19]. Robust semantics extends the classical Boolean satisfaction relation $\mathbf{w} \models \varphi$ by assigning a value $[\![\mathbf{w}, \varphi]\!] \in \mathbb{R} \cup \{\infty, -\infty\}$ (i.e., *robustness*) that tells not only whether φ is satisfied or violated (by the sign), but also *how robustly* the formula is satisfied or violated. Optimization-based falsification algorithms iteratively generate inputs with the aim of finding an input with negative robustness. Several optimization-based falsification algorithms have been developed [1,3,10,13,18,29,33,35–38].

Given a specification φ, a falsification algorithm either returns an input signal falsifying φ, or it reports that the search was *unsuccessful* if no such input was found. As usual in testing (falsification is a particular type of search-based testing approach), in the latter case, we do not know whether the specification φ is really not falsifiable, or the algorithm did not explore the search space enough. In such a case, a practitioner would like to have some estimate of the real absence of a falsifying input (and so of the satisfaction of φ).

To this aim, in this paper, we propose a *confidence measure* for hybrid system falsification. The definition of the measure starts from the observation that, as output, a falsification algorithm also provides the set of input signals that have been sampled during the search, together with their corresponding robustness values. Starting from these data[1], we try to estimate the likelihood that no falsifying input exists in the unexplored search space. The construction of the confidence measure is as follows. We first train, using *Gaussian Process Regression* [31], a *Gaussian Process* (GP) from the falsification data, acting as a *surrogate model* of the real robustness function, i.e., it provides an *estimation* of the robustness. Then, we sample in the GP the points that have the minimum values, as these are the points that have the higher probability to identify negative robustness. We then compute the cumulative probability that these surrogate sampled data are all positive, i.e., that the approximated robustness is always positive, and so the specification is not falsified. We take this probability as the *confidence value* estimating how likely it is that the specification holds.

[1] Note that we assume that the confidence measure is computed starting from non-falsifying inputs only. The measure does not make sense if at least one falsifying input is used for its computation; in that case, there is no need of the confidence measure, as we know that the specification is falsifiable.

We performed a series of experiments to assess to what extent the proposed confidence measure guarantees some desired properties:

- *monotonicity*: the confidence measure should not decrease as new falsification data (i.e., inputs with corresponding robustness values) are being added. A monotonically increasing measure does not prematurely assess high confidence, and so it can be reliably used to decide whether enough falsification search has been performed; it can also be used as a *stopping criterion* during the falsification search itself. In classical coverage criteria for software testing, increasing monotonicity is implicitly guaranteed, because the test requirements are known in advance, and so any new test input can only increase the coverage (when an uncovered test requirement is covered) or at most leave it unchanged. For our confidence measure, increasing monotonicity cannot be guaranteed, as it is always possible to find a new input signal that drastically changes the derived Gaussian Process and so the computed confidence. However, we will experimentally show that the measure can efficiently account for the unexplored search space and that monotonicity is guaranteed to some extent.
- *soundness*: intuitively, the confidence should also depend on "how robustly" a specification holds. Given two specifications that are both non-falsifiable, the one that is more robust should lead to higher confidence than the less robust one. Take the example of a car system, and two specifications requiring that the "speed is always less than 120" and "speed is always less than 150"; intuitively, in the absence of a falsifying input, the confidence of the latter specification should not be lower than that of the former one (given that a falsification approach has been run for both with the same budget).

In the experiments, we will also use the confidence measure to assess the performance of existing falsification algorithms, which implement different search strategies. Namely, we will check the confidence provided by three falsification algorithms (`Random` search, `CMAES`, and `MCTS`) executed with the same budget; moreover, we will also check their *stability*, i.e., how much the confidence measure changes in repeated runs.

Paper Structure. Section 2 provides the necessary background. Section 3 introduces the problem and overviews the approach to compute the proposed confidence measure, whose phases are described in Sect. 4. Section 5 presents the experiments done to assess the measure. Finally, Sect. 6 reviews related work, and Sect. 7 concludes the paper.

2 Background

We here review the basic concepts and approaches of hybrid system falsification.

System Model. Let $T \in \mathbb{R}_{\geq 0}$ be a positive real. An *M-dimensional signal* with a time horizon T is a function $\mathbf{w}\colon [0,T] \to \mathbb{R}^M$. We treat the system model as a black box, i.e., its behaviors are only observed from inputs and their corresponding outputs. Formally, a *system model*, with M-dimensional input and

N-dimensional output, is a function $\mathcal{M}$ that takes an input signal $\mathbf{u}\colon [0,T] \to \mathbb{R}^M$ and returns a signal $\mathcal{M}(\mathbf{u})\colon [0,T] \to \mathbb{R}^N$. Here the common time horizon $T \in \mathbb{R}_{\geq 0}$ is arbitrary. The process of obtaining a system output signal $\mathcal{M}(\mathbf{u})$, given an input signal $\mathbf{u}$, is called *simulation*.

Specifications. In this work, we adopt *Signal Temporal Logic (STL)* as our specification language to formalize properties that should be satisfied by the system. We introduce the syntax and semantics in the following.

Definition 1 (STL syntax). We fix a set **Var** of variables. In Signal Temporal Logic (STL), *atomic propositions* and *formulas* are defined as follows, respectively: $\alpha ::\equiv f(x_1,\ldots,x_N) > 0$, and $\varphi ::\equiv \alpha \mid \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi\, \mathcal{U}_I\, \varphi$. Here f is an N-ary function $f : \mathbb{R}^N \to \mathbb{R}$, $x_1,\ldots,x_N \in \mathbf{Var}$, and I is a closed non-singular interval in $\mathbb{R}_{\geq 0}$, i.e. $I = [a,b]$ or $[a,\infty)$ where $a,b \in \mathbb{R}$ and $a < b$. Other common connectives such as $\rightarrow, \top, \Box_I$ (always) and $\Diamond_I$ (eventually), are introduced as abbreviations: $\Diamond_I \varphi \equiv \top\, \mathcal{U}_I\, \varphi$ and $\Box_I \varphi \equiv \neg \Diamond_I \neg \varphi$.

Definition 2 (Robust semantics). Let $\mathbf{w}\colon [0,T] \to \mathbb{R}^N$ be an N-dimensional signal, and $t \in [0,T)$. The *t-shift* $\mathbf{w}^t$ of $\mathbf{w}$ is the signal $\mathbf{w}^t\colon [0, T-t] \to \mathbb{R}^N$ defined by $\mathbf{w}^t(t') := \mathbf{w}(t+t')$. Let $\mathbf{w}\colon [0,T] \to \mathbb{R}^{|\mathbf{Var}|}$ be a signal, and φ be an STL formula. We define the *robustness* $[\![\mathbf{w}, \varphi]\!] \in \mathbb{R} \cup \{\infty, -\infty\}$ as follows, by induction on the construction of formulas. $\bigsqcap$ and $\bigsqcup$ denote infimums and supremums of real numbers, respectively. Their binary version $\sqcap$ and $\sqcup$ denote minimum and maximum.

$$
\begin{array}{ll}
[\![\mathbf{w}, f(x_1,\cdots,x_N) > 0]\!] := f\big(\mathbf{w}(0)(x_1),\cdots,\mathbf{w}(0)(x_N)\big) & \\
[\![\mathbf{w}, \bot]\!] := -\infty & [\![\mathbf{w}, \neg\varphi]\!] := -[\![\mathbf{w}, \varphi]\!] \\
[\![\mathbf{w}, \varphi_1 \wedge \varphi_2]\!] := [\![\mathbf{w}, \varphi_1]\!] \sqcap [\![\mathbf{w}, \varphi_2]\!] & \\
[\![\mathbf{w}, \varphi_1\, \mathcal{U}_I\, \varphi_2]\!] := \bigsqcup_{t \in I \cap [0,T]} \big([\![\mathbf{w}^t, \varphi_2]\!] \sqcap \bigsqcap_{t' \in [0,t)} [\![\mathbf{w}^{t'}, \varphi_1]\!] \big) &
\end{array}
$$

The original STL semantics is Boolean, given by a binary relation $\models$ between signals and formulas. The robust semantics refines the Boolean one as follows: $[\![\mathbf{w}, \varphi]\!] > 0$ implies $\mathbf{w} \models \varphi$, and $[\![\mathbf{w}, \varphi]\!] < 0$ implies $\mathbf{w} \not\models \varphi$, see [19, Prop. 16].

In the following, given a fixed specification φ, we denote as $\mathsf{R}_{\mathbf{u}}$ the robustness of an input signal $\mathbf{u}$ to the specification φ, i.e., $\mathsf{R}_{\mathbf{u}} = [\![\mathcal{M}(\mathbf{u}), \varphi]\!]$.

Falsification Approaches. Falsification consists in synthesizing input signals to find one that violates the system specification. The targets of falsification, i.e., input signals $\mathbf{u}\colon [0,T] \to \mathbb{R}^M$, are time-variant continuous functions. In practice, synthesizing such continuous signals is infeasible. Hence, practitioners employ parametrized representations to characterize the signals [13,16,26]; a commonly used representation is *piecewise constant*. A piecewise constant signal $\mathbf{u}\colon [0,T] \to \mathbb{R}^M$ has a hyperparameter c, such that during each interval $[\frac{(i-1)T}{c}, \frac{iT}{c}]$ $(i \in 1,\ldots,c)$, $\mathbf{u}(t)$ is a constant. In this way, a finite number $c \cdot M$ of parameters is used to identify a signal $\mathbf{u}$. We will identify with Ω the $(c \cdot M)$-dimensional hyperrectangle identifying the *input space* (also *search space*) used for falsification.

Various approaches have been proposed to solve the falsification problem. The naive one is by *uniformly random sampling* input signals $\mathbf{u}$ in the input

space Ω, and check if the corresponding output signal $\mathcal{M}(\mathbf{u})$ violates the specification φ. A more efficient but greedy algorithm is by exploiting the STL robust semantics and turning falsification into an optimization problem that minimizes the robustness $[\![\mathcal{M}(\mathbf{u}), \varphi]\!]$; the process stops when a negative robustness result $[\![\mathcal{M}(\mathbf{u}), \varphi]\!]$ is observed (i.e., a falsifying input $\mathbf{u}$ has been found), or the search budget (in terms of number of simulations) expires.

In order to solve the optimization problem, *stochastic optimization* algorithms, such as *hill climbing*, are employed, because they work efficiently with black box objective functions, like $[\![\mathcal{M}(\mathbf{u}), \varphi]\!]$ in our case. These algorithms adopt various metaheuristic strategies so that they can efficiently explore the search space and achieve optimal solutions. For instance, *CMAES* [23] is an evolutionary search-based method that focuses on exploitation. A more recent work *MCTS* [36] employs *Monte-Carlo Tree Search* to achieve a balance between exploration and exploitation of the search space. We refer readers to [8] for a more comprehensive survey of different optimization-based falsification algorithms. Also mature tools, such as Breach [13] and S-TaLiRo [3], have been developed.

3 Problem Definition and Overview of the Proposed Approach

In this work, we tackle the problem of characterizing the likelihood that a formal specification is indeed non-falsifiable, given that a falsification algorithm has tried to falsify it using a given set of input signals, all giving positive robustness.

Definition 3 (Confidence Estimation Problem). The *confidence estimation problem* is formally defined as follows:

- **Given:** a finite set $T = \{\langle \mathbf{u}_1^*, \mathsf{R}_{\mathbf{u}_1^*}\rangle, \ldots, \langle \mathbf{u}_N^*, \mathsf{R}_{\mathbf{u}_N^*}\rangle\}$ of pairs, where each $\mathbf{u}_i^* \in \Omega$ is a point in the input space Ω, and $\mathsf{R}_{\mathbf{u}_i^*} \in \mathbb{R}_{\geq 0}$ is the robustness (a positive real) of $\mathbf{u}_i^*$.
- **Return:** the likelihood that, for all points $\mathbf{u}' \in \Omega$, it holds $\mathsf{R}_{\mathbf{u}'} > 0$.

This problem is in general undecidable: first, the robustness computation $\mathsf{R}(\mathbf{u}) = [\![\mathcal{M}(\mathbf{u}), \varphi]\!]$ relies on a black box model $\mathcal{M}$, in which the robustness values of unexplored points are not predictable (i.e., R is unknown); moreover, there are infinitely many unexplored points in the search space Ω, regardless of the size of T, since Ω is a continuous domain.

Overview of the Proposed Approach for Confidence Estimation. We provide an overview of the proposed approach in Fig. 1. We assume that a falsification algorithm has been run for some time with the aim of falsifying a given specification for a given system, without success (i.e., no falsifying input has been found). The proposed approach takes as input the produced falsification data T, consisting of a set of system inputs $\mathbf{u}^*$ and their robustness values $\mathsf{R}_{\mathbf{u}^*}$, as defined in Definition 3. These data, after being normalized, are fed into a Gaussian Process (GP) Regression process to train a GP as a surrogate of robustness

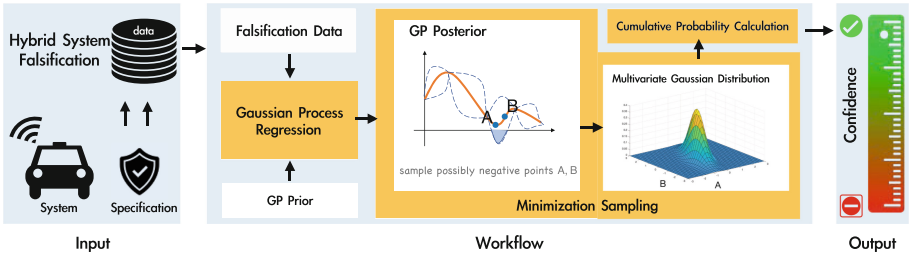


Fig. 1. The proposed GP regression-based confidence estimation approach for hybrid system falsification

function R. Using the obtained GP, we want to estimate the probability that there exists no point having negative robustness. First, we need to identify the points with lower values in the GP (so approximating lower robustness values), as these are those that can actually reduce the probability; to do this, we perform a global sampling method to collect a set of points that have low values, and then perform a local search starting from these points with the aim of finding points with even lower values. Starting from the found points, we apply an established method [6] to compute the probability that all these points still approximate positive robustness values. The approach outputs a confidence value $\mathsf{conf} \in [0, 1]$ that indicates *how likely* the system satisfies the system specification.

4 Confidence Estimation via Gaussian Process Regression

In this section, we explain all the phases of the process shown in Fig. 1. In Sect. 4.1, we first explain how we derive a Gaussian Process (GP) from the falsification data. Then, in Sect. 4.2, we describe how we sample from the obtained GP, and derive a confidence measure from the sampled data.

4.1 Building a Surrogate Model of the Robustness Function via Gaussian Process Regression

A *Gaussian Process* (GP) is a generalization of *Gaussian distribution* (a.k.a. *normal distribution*) from single/multiple variables to continuous domains [31]. While a Gaussian distribution characterizes the probability distribution of a finite set of variables, a GP models the distribution of infinite variables in a continuous domain, i.e., sampling from a GP derives a function instance in the domain. Formally, a GP over a continuous domain Ω is defined as a collection of random variables X_t indexed by $t \in \Omega$, such that any finite set $\mathsf{X} = \{X_t \mid t \in \Omega\}$ of those variables compose a *multivariate Gaussian distribution* $\mathsf{X} \sim \mathcal{N}(\mu, \Sigma)$, where μ and Σ are the *mean vector* and the *covariance matrix* of the distribution, respectively. The correlation between two variables $X_{t_i}, X_{t_j} \in \mathsf{X}$ is reflected by their covariance $\Sigma_{i,j}$—the larger $|\Sigma_{i,j}|$ is, the more highly X_{t_i} and X_{t_j} are

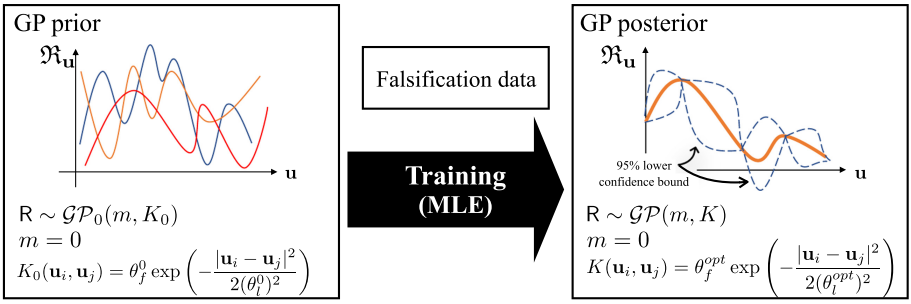


Fig. 2. Gaussian Process regression for the approximation of the robustness function

correlated. In a GP, the covariance between two variables X_{t_i} and X_{t_j} is dependent on the (Euclidean) distance of their indices t_i and t_j—the closer t_i and t_j are, the larger the covariance is. This is implemented by the *covariance function* (a.k.a. *kernel*) $K \colon \Omega \times \Omega \to \mathbb{R}$ of GP, whose complicated definition will be elaborated later.

GP regression consists in deciding the covariance function of a GP, by learning from the observed data, so that the distributions of unknown variables can be predicted, based on Bayes' rules [31, §2.1]. Unlike other regression frameworks (e.g., *deep learning*), the prediction for an unknown point made by the GP provides a *Gaussian distribution*, i.e., a *mean* and a *variance*, rather than a single value, so we can exploit this information to derive our confidence measure.

The prerequisite of GP regression is that the function that we want to approximate guarantees the following assumption [31].

Assumption 1. Two points that are geometrically closer in the input space have also closer output values.

In general, the assumption is reasonable for any robustness function, as a small tuning of input signals usually leads to a small change of the system outputs and so of the robustness value; such assumption is common in different works on testing of CPSs [2,9,32]. So, we can apply GP regression for approximating the robustness function. Figure 2 shows the GP regression we apply to learn a surrogate model for the robustness function R from the falsification data $T = \{\langle \mathbf{u}_1^*, \mathsf{R}_{\mathbf{u}_1^*} \rangle, \ldots, \langle \mathbf{u}_N^*, \mathsf{R}_{\mathbf{u}_N^*} \rangle\}$ in Definition 3. The process is elaborated in the following.

GP Prior. A GP is uniquely identified by a mean function m and a covariance function K, denoted as $\mathsf{R} \sim \mathcal{GP}(m, K)$. To determine m and K, a commonly-used approach is to first specify two parametrized function templates, and then fit the parameters based on training data. Conventionally, the mean function m can simply be a constant; here we use 0. The selection of the covariance function K is more sophisticated, as it is required to guarantee Assumption 1. Widely-used choices for K include *squared exponential kernel*, *Matérn kernel* and so on

(see a discussion in [31, §4.2]). In this work, we follow a typical selection, namely, the *squared exponential kernel*, shown as follows:

$$K(\mathbf{u}_i, \mathbf{u}_j \mid \theta_f, \theta_l) = \theta_f \exp\left(-\frac{|\mathbf{u}_i - \mathbf{u}_j|^2}{2\theta_l^2}\right)$$

where $\theta_f, \theta_l \in \mathbb{R}$ are tunable hyperparameters. Initially (the left part of Fig. 2), these parameters have arbitrary values θ_f^0 and θ_l^0 and identify a function K_0; by this function, we obtain an initial *GP prior* $\mathsf{R} \sim \mathcal{GP}_0(m, K_0)$.

Training. The parameter tuning process aims at finding the optimal values θ_f^{opt}, θ_l^{opt} for θ_f, θ_l, such that, under θ_f^{opt} and θ_l^{opt}, the likelihood of the occurrence of the falsification data T (in Definition 3) is maximized. This method is referred to as *maximum likelihood estimation (MLE)*, shown as follows:

$$\theta_f^{opt}, \theta_l^{opt} = \underset{\theta_f, \theta_l}{\arg\max}\ P\left(\mathsf{R}_{\mathbf{u}_1^*}, \ldots, \mathsf{R}_{\mathbf{u}_N^*} \mid \theta_f, \theta_l\right) \tag{1}$$

where P is the *probability density function* of the multivariate Gaussian distribution of $\left[\mathsf{R}_{\mathbf{u}_1^*} \ldots \mathsf{R}_{\mathbf{u}_N^*}\right]$ (see [31, §2.2 and §5] for more details). This problem can be solved by a numerical optimization solver, such as *Quasi-Newton method* [7].

GP Posterior. Given the decision of hyperparameters θ_f and θ_l, we can fix our *GP posterior* for the robustness function R (the right part of Fig. 2) as $\mathsf{R} \sim \mathcal{GP}(m, K)$. By this, the posterior distribution of any unknown point $\mathbf{u} \in \Omega$ is inferred as follows, according to Bayes' rules [31, §2.2 and §A.2]:

$$\mathfrak{R}_{\mathbf{u}} \sim \mathcal{N}(\mu, \sigma^2) \qquad \mu = \Sigma_{\mathbf{u}} \Sigma_{\mathbf{u}^*}^{-1} \mathsf{R}_{\mathbf{u}^*} \quad \sigma^2 = K(\mathbf{u}, \mathbf{u}) - \Sigma_{\mathbf{u}} \Sigma_{\mathbf{u}^*}^{-1} \Sigma_{\mathbf{u}}^T \tag{2}$$

where $\Sigma_{\mathbf{u}} = \left[K(\mathbf{u}, \mathbf{u}_1^*) \ldots K(\mathbf{u}, \mathbf{u}_N^*)\right]$, $\Sigma_{\mathbf{u}^*} = \begin{bmatrix} K(\mathbf{u}_1^*, \mathbf{u}_1^*) & \ldots & K(\mathbf{u}_1^*, \mathbf{u}_N^*) \\ \vdots & \ddots & \\ K(\mathbf{u}_N^*, \mathbf{u}_1^*) & & K(\mathbf{u}_N^*, \mathbf{u}_N^*) \end{bmatrix}$, and $\mathsf{R}_{\mathbf{u}^*} = \left[\mathsf{R}_{\mathbf{u}_1^*} \ldots \mathsf{R}_{\mathbf{u}_N^*}\right]$. $\mathfrak{R}_{\mathbf{u}}$ identifies the *inferred distribution* for the robustness of $\mathbf{u}$.

4.2 Confidence Estimation

In this section, we describe how we derive a *confidence measure* on the result of falsification, based on the surrogate GP posterior we obtained in Sect. 4.1. Accordingly, the problem introduced in Definition 3 is reformulated in terms of GP as follows:

Definition 4 (Confidence Estimation Based on GP Posterior)

- **Given:** the GP posterior $\mathsf{R} \sim \mathcal{GP}(m, K)$;
- **Return:** the likelihood that any inferred distribution $\mathfrak{R}_{\mathbf{u}}$ (for all $\mathbf{u} \in \Omega$) is positive.

The problem in Definition 4 asks for the *tail probability* of a given GP; the exact answer of the problem is hard in general, and is still actively pursued in the

Algorithm 1. Confidence estimation

Require: The $\mathsf{R} \sim \mathcal{GP}(m, K)$ obtained in Sect. 4.1 and its input space Ω
Require: H: the number of points to randomly sample from $\mathcal{GP}$
Require: h: the number of top-h sampled points to use for local search, s.t. $h \ll H$
Ensure: conf: the confidence value that the specification holds

1: **function** EstimateConfidence($\mathcal{GP}$, Ω, H, h)
2: collect a set U of $\mathbf{u} \in \Omega$ by randomly sampling H points in Ω
3: sort all $\mathbf{u} \in U$ ascendingly by $\mathsf{lowBound}(\mathbf{u})$ and take the first h ones $\{\mathbf{u}_1, \ldots, \mathbf{u}_h\}$
4: **for** $i \in \{1, \ldots, h\}$ **do**
5: $\mathbf{u}'_i \leftarrow \arg\min_{\mathbf{u} \in \Omega} \mathsf{lowBound}(\mathbf{u})$ starting from $\mathbf{u}_i$ ▷ local search
6: $\mathfrak{R}_{\mathbf{u}'_1}, \ldots, \mathfrak{R}_{\mathbf{u}'_h} \sim \mathcal{N}(\mu, \Sigma)$ s.t., $\begin{cases} \mu \leftarrow \Sigma_{\mathbf{u}} \Sigma_{\mathbf{u}^*}^{-1} \mathsf{R}_{\mathbf{u}^*} \\ \Sigma \leftarrow K(\mathbf{u}, \mathbf{u}) - \Sigma_{\mathbf{u}} \Sigma_{\mathbf{u}^*}^{-1} \Sigma_{\mathbf{u}}^T \end{cases}$ ▷ see Eq. 2
7: **return** $\mathsf{conf} \leftarrow \text{MvnCdf}\big(\langle \mathfrak{R}_{\mathbf{u}'_1}, \ldots, \mathfrak{R}_{\mathbf{u}'_h} \rangle, [0, \infty)\big)$

GP community [24,27,28]. Since our work mainly aims at giving a hint of *how likely* it is that there exists no counterexample in the falsification search space, an approximation of the tail probability is enough. In this section, we introduce a sampling-based approach to approximate the likelihood (i.e., the confidence measure) as defined in Definition 4.

The process of confidence estimation is shown in Algorithm 1.

It requires the GP posterior $\mathcal{GP}$ obtained in Sect. 4.1 with its input space Ω, and two natural numbers $H, h \in \mathbb{N}^+$ such that $h \ll H$. The algorithm consists in a *minimization sampling* phase, and a *cumulative probability calculation* phase; at the end, it returns a confidence value $\mathsf{conf} \in [0, 1]$ that indicates how likely it is that there exists no counterexample in Ω. In the following, we elaborate on the process.

Minimization Sampling. The likelihood in Definition 4 is decided by its *dual problem*, i.e., the likelihood that there exists $\mathbf{u} \in \Omega$ whose $\mathfrak{R}_{\mathbf{u}}$ is negative. To answer this problem, we need to collect the points in Ω, whose inferred robustness values have a considerable probability to be negative, and then calculate their cumulative probability distribution. Since Ω is continuous, there could exist infinitely many such points, and hence it is impossible to involve all of them. However, our work mainly aims at giving an approximation of the likelihood, so it suffices to select a finite set of points from Ω as representatives for that class of points. To this aim, we construct a new function $\mathsf{lowBound}(\mathbf{u}) = \mu(\mathbf{u}) - 1.96\sigma(\mathbf{u})$, that is, the 95% lower confidence bound of the posterior distribution of $\mathfrak{R}_{\mathbf{u}}$ in Eq. 2 (see an illustration in the GP posterior in Fig. 2). Intuitively, if $\mathsf{lowBound}(\mathbf{u})$ is lower than 0, there is still a considerable probability that $\mathfrak{R}_{\mathbf{u}}$ is negative; otherwise, we consider that $\mathfrak{R}_{\mathbf{u}}$ is unlikely to be negative.

Therefore, this phase consists of an optimization process, taking $\mathsf{lowBound}(\mathbf{u})$ as the objective function to be minimized. In order to derive comprehensive and precise estimation, we adopt a hybrid optimization approach that combines

global search and local search. First, in the global search, we perform a comprehensive random sampling in Ω (Line 2) and sort the H points ascendingly according to their 95% lower confidence bound $\mathsf{lowBound}(\mathbf{u})$ (Line 3); we select the first h sampled points to perform a further minimization process (Line 3). Namely, we take each of the h sampled points as a starting point to perform a local search, aiming at finding new points that have even lower confidence bound (Line 5); we use these points to construct a multivariate Gaussian distribution, based on the definition of GP (Line 6).

Cumulative Probability Calculation. Given the constructed multivariate Gaussian distribution $\mathfrak{R}_{\mathbf{u}'_1}, \ldots, \mathfrak{R}_{\mathbf{u}'_k} \sim \mathcal{N}(\mu, \Sigma)$ in Line 6, we can apply existing methods to calculate the probability that $\mathfrak{R}_{\mathbf{u}'_1}, \ldots, \mathfrak{R}_{\mathbf{u}'_k}$ are all positive. There have been a line of works doing this; in our work, we adopt the state-of-the-art, that is, a *minimax tilting*-based approach [6] to calculate the probability. The function MvnCdf in Line 7 shows the interface of [6]: it takes as input the multivariate Gaussian distribution, i.e., its mean and covariance matrix, and the range in which the distribution is expected to locate; it returns a value conf, that indicates how likely the multivariate Gaussian distribution distributes in the given range.

5 Experimental Evaluation

We here describe the experiments we conducted to evaluate the proposed confidence measure. We first present the experiments settings in Sect. 5.1; then, in Sect. 5.2, we analyze experimental results using a series of research questions.

5.1 Experiment Settings

As benchmarks, we selected three Simulink models and seven STL specifications defined for them, that are commonly used, in particular in falsification competitions [11,16,17]. Table 1 reports the benchmarks and the corresponding specifications. The specification ID identifies the corresponding model. A description of the models and of their specifications is as follows.

Table 1. Specifications. For each one, we list a set of parameters $\omega_1, \ldots, \omega_5$ for RQ2; the default parameter used in RQ1 and RQ3 is indicated by ω_d

Spec. ID	STL formula	ω_i $(i = 1, \ldots, 5)$	ω_d
AT1	$\square_{[0,30]}$ $(speed < \omega_i)$	{135, 140, 145, 150, 155}	ω_3
AT2	$\square_{[0,10]}$ $(speed < 50) \vee \Diamond_{[0,30]}$ $(rpm > \omega_i)$	{500, 750, 1000, 1250, 1500}	ω_4
AT3	$\square_{[0,29]}(speed < 100) \vee \square_{[29,30]}(speed > \omega_i)$	{57, 59, 61, 63, 65}	ω_2
AT4	$\square_{[0,30]}(rpm < 4770 \vee \square_{[0,1]}(rpm > \omega_i))$	{200, 300, 400, 500, 600}	ω_1
FFR1	$\neg \Diamond_{[0,5]}$ $x, y \in [3.9 + \omega_i, 4.1 - \omega_i] \wedge \dot{x}, \dot{y} \in [-0.5, 0.5]$	{0.01, 0.03, 0.05, 0.07, 0.09}	ω_1
FFR2	$\neg\Diamond_{[0,5]}\square_{[0,2]}(x, y \in [1.5 + \omega_i, 1.7 - \omega_i])$	{0, 0.02, 0.04, 0.06, 0.08}	ω_2
AFC1	$\square_{[11,50]}(controller\_mode = 1 \rightarrow \mu < \omega_i)$	{0.25, 0.3, 0.35, 0.4, 0.45}	ω_2

- *Automatic Transmission* (AT) [25] models an automotive system that has two input signals, *throttle* $\in [0, 100]$ and *brake* $\in [0, 325]$, and three outputs signals, *gear*, *speed*, and *rpm*. The specifications consider system safety: requirements on *speed* (AT1, AT3), on *rpm* (AT4), or their relation (AT2).
- *Free Floating Robot* (FFR) [9] models a robot operating in a 2D space. It has four input signals $u_1, u_2, u_3, u_4 \in [-10, 10]$ representing the boosters, and four output signals that represent the coordinates x, y of the robot position, and their one-order derivatives $\dot{x}, \dot{y}$. The specifications specify kinetic properties for the robot: FFR1 requires the robot to pass an area around the point $(4, 4)$ under an input constraint, and FFR2 requires the robot to stay in a given area for at least 2 s.
- *Abstract Fuel Control* (AFC) [26] takes two input signals, *Pedal_Angle* $\in [8.8, 70]$ and *Engine_Speed* $\in [900, 1100]$, and outputs a ratio μ reflecting the deviation of *air-fuel-ratio* from its reference value. A requirement (AFC1) of the system is that μ shouldn't deviate too much from the reference value.

Specifications are parameterized by a parameter ω_i; we set it to five different values to have specifications of different complexity for the experiments.

Software and Hardware Specifications. Our experiments rely on `Breach` [13] to interface Simulink and compute STL robustness. The experiments have been executed on an AWS EC2 c4.2xlarge instance (2.9 GHz Intel Xeon E5-2666 v3, 15 GB RAM). The code and all the experimental results are available online at https://github.com/choshina/GPConfidence.

5.2 Evaluation

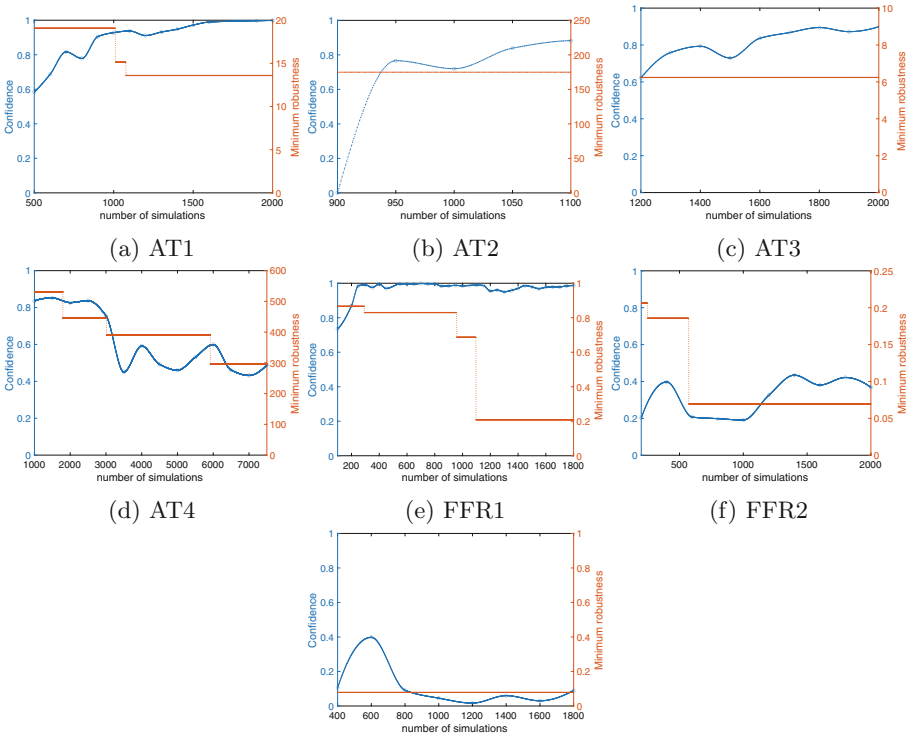
We assess the viability of the proposed confidence measure using three research questions, described as follows.

RQ1. Does the confidence measure monotonically increase?

As explained in Sect. 1, a confidence measure, in order to be reliable, should be monotonically increasing, i.e., it should not decrease as new falsification data is considered. As previously explained, this cannot be theoretically guaranteed by the proposed metric, as it is always possible to find a new input that drastically changes the learned GP. Therefore, we here want to assess to what extent the proposed metric is monotonically increasing. For each benchmark (using the default value ω_d), we have randomly sampled an increasing number of points in Ω; then, for each set of sampled points T, we have computed the minimum robustness and the confidence measure.

Figure 3 shows, for each benchmark, how the minimum robustness value (right axis and red plot) and the confidence measure (left axis and blue plot) change by increasing the size of T.[2] The plots show the intervals of sizes of T in which the confidence measure changes significantly, before stabilizing to a given

[2] Since computing the confidence measure can take up to 30 s, we have computed it only for some sizes of $|T|$.



(a) AT1 (b) AT2 (c) AT3

(d) AT4 (e) FFR1 (f) FFR2

(g) AFC1

Fig. 3. RQ1 – Experimental results on monotonicity

value. We observe that, in most of the cases (i.e., AT1, AT2, AT3, FFR1, and FFR2), the measure is quite monotonic, with few oscillations for lower number of simulations (e.g., FFR2); these initial oscillations are expected, because the precision of the GP depends on the number of elements of the training data. For some cases, instead, the confidence measure remains high for some time at the beginning (i.e., AT4 and AFC1) and then stabilizes at a lower value. For AFC1, the reason is that the minimum robustness is always very low; so, with more observations, the GP learns such low robustness, and so the confidence on the absence of a falsifying input decreases. For AT4, the main decrease in confidence occurs due to big drops in minimum robustness. A way to increase the confidence also in the presence of low robustness would be to add more training data, so that the GP can get a better estimation; this is not applicable in our context, because we rely only on the falsification data. However, in the next research question, we show that the confidence metric is still reliable.

Answer to RQ1: Most of the time, the confidence measure is reliable, as it does not decrease (significantly) with increasing number of simulations. However, in some cases, the measure is fluctuating for a lower number of simulations, due to big changes in minimum robustness.

RQ2. Is the confidence measure sound?

We claim that a confidence measure is sound if it is related to the difficulty of falsifying a specification, which can be assessed by the minimum robustness that is achieved by a falsification algorithm. Therefore, we expect that the lower the minimum robustness is, the lower the confidence should be. To assess to what extent the proposed confidence measure is sound, we compare the confidence results for the different instantiations of a specification type in Table 1; indeed, these represent similar problems that, however, can have different values of minimum robustness (as the specification is more or less demanding). Concretely, for each specification type and for each of the different values of the parameter ω_i, we have run random sampling for 2000 simulations, and we have collected the minimum robustness and the confidence measure over the simulations. Then, for each specification type, we have selected the minimum number of simulations for which, in at least one instantiation of ω_i, the confidence measure reaches 100% (or we select the maximum number 2000 if none of them reaches 100%), and we take this number of simulations to compare the results of the specification type: in this way, we can observe whether there are actually differences, and we avoid the saturation effect that may be obtained by using a lot of simulations in which the results converge to the same value.

Table 2 reports, for each specification type, results of the different instantiations, in terms of minimum robustness (minRob), confidence measure (conf), and confidence computation time (in secs); results are sorted increasingly by minRob.

Table 2. RQ2 – Experimental results on soundness. conf is in %; time is in secs.

AT1				AT2				AT3				AT4			
ω_i	minRob	conf	time	ω_i	minRob	conf	time	ω_i	minRob	conf	time	ω_i	minRob	conf	time
ω_1	5.12	39.23	22.8	ω_5	33.42	0	30	ω_5	4.82	82.52	22.4	ω_3	355.44	56.59	29.9
ω_2	10.06	97.77	26.3	ω_4	34.55	0.01	33.4	ω_3	5.56	93.89	19.6	ω_4	445.41	87.62	30.9
ω_3	13.6	49.93	24.2	ω_3	43.72	0	27.9	ω_4	6.24	89.67	21.4	ω_5	466.57	85.86	26.6
ω_4	18.1	100	23.43	ω_2	250	26.33	29.3	ω_2	7.13	93.34	22.5	ω_2	473.54	90.79	30.6
ω_5	22.41	100	26.6	ω_1	500	88.92	29.7	ω_1	9.6	97.45	23.4	ω_1	608.4	91.99	28.5

FFR1				FFR2				AFC1			
ω_i	minRob	conf	time	ω_i	minRob	conf	time	ω_i	minRob	conf	time
ω_5	0.21	99.28	20	ω_3	0.01	1.27	23.2	ω_1	0.03	1.84	0.3
ω_3	0.49	99.95	20.3	ω_1	0.02	11.17	21.2	ω_2	0.08	10.48	0.3
ω_2	0.5	90.84	21.5	ω_2	0.03	2.53	20.2	ω_3	0.13	99.95	0.2
ω_1	0.58	98.88	24.2	ω_4	0.07	47.23	16.8	ω_4	0.18	100	0.3
ω_4	0.64	92.37	23.1	ω_5	0.08	0.91	21.1	ω_5	0.23	100	0.3

We note that, overall, the metric is sound because, given a specification type, lower confidence values are obtained for lower robustness. Usually, large change in confidence occurs when there is a big percentage change in robustness, such as AT2, where the maximum value of minRob is 21.3% bigger than the minimum value, and the difference in confidence is 88.92 perc. points. In some cases, the variance of conf across the instantiations is not too high, as in AT3, AT4, and FFR1 (differences of 14.93, 35.4, and 9.11 perc. points); we notice that, in these cases, the percentage change of robustness is small (0.99%, 0.36%, and 2.04%).

Answer to RQ2: Overall, the confidence measure is sound as it is related to the difficulty of the specification, measured by the minimum robustness.

RQ3. What is the confidence provided by different falsification algorithms?

We are here interested in investigating what is the confidence provided by different falsification algorithms (i.e., implementing different search strategies) when they are not able to falsify. We took three representative algorithms: `Random` search that performs pure *exploration*, `CMAES` [23] that is a greedy approach that favours *exploitation*, and `MCTS` [36] that provides a balance between exploration and exploitation. We have applied the three algorithms (for 2000 simulations) to three benchmark specifications (AT3, FFR1, and AFC1 instantiated with ω_d as reported in Table 1); we repeated the experiments 10 times.

Figure 4 reports how the confidence measure changes across the 10 repetitions. We observe that higher confidence is usually obtained by `Random`: since it explores more, it can provide more distributed data for GP learning, so obtaining a better GP (with less uncertainty). On the contrary, `CMAES` obtains low confidence as it does not explore enough and so the GP learning does not have training data for large parts of the search space in which the uncertainty of the GP will be high. `MCTS` can obtain good results when it explores enough (as in

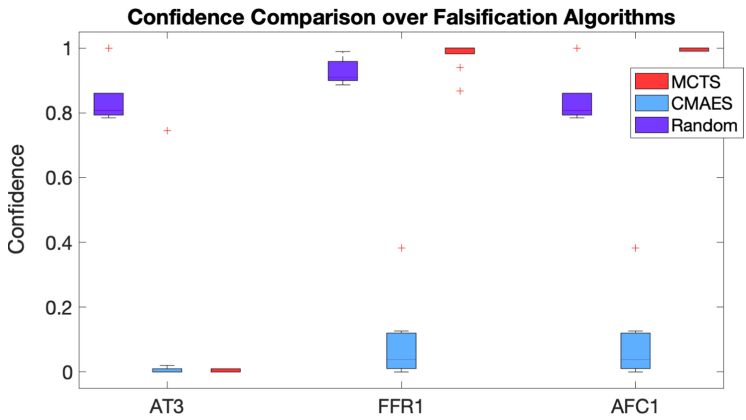


Fig. 4. Confidence comparison between different falsification algorithms

FFR1 and AFC1), but sometimes it can be too greedy (as in AT3) and so it obtains low confidence. Regarding the variance of the results, we observe that `CMAES` is the one with the highest variance, as different search trials can lead to different search paths and so different confidence values; also `Random` has some variance across the results, but not as much as `CMAES`. `MCTS` is the more stable approach, as the balance between exploration and exploitation leads most of the times to the same search paths, and so to the same confidence values.

Answer to RQ3: Falsification approaches that provide enough exploration lead to higher confidence. `MCTS` is the more stable algorithm.

6 Related Work

Gaussian processes and Gaussian process regression are widely applied in different contexts. In statistics, GP regression, usually referred to as *Kriging methods*, is used to build surrogate models for measuring the probability of rare events, e.g., [5,21,34]. These works usually develop new sampling techniques to derive rare events probability using limited numbers of observations, and have been widely applied in engineering domains. Theoretical explorations in tail probability of GPs have been heavily conducted in the statistics community [24,27,28]. In general, this is a hard problem and research efforts are still on-going; achievements so far usually limit GP to certain types, e.g., *Brownian motion*.

In falsification, GP regression has been mainly used for guiding the search process, and not for confidence estimation as done in this paper. Deshmukh et al. [9] apply *Bayesian optimization*, an optimization approach derived from GP regression, to falsification, and investigates a dimension reduction technique. Akazaki [2] uses it for the falsification of conditional properties $\square_I \varphi_{cond} \rightarrow \varphi_{safe}$ in which the antecedent φ_{cond} must be satisfied in order to be able to violate the property; GP regression is used to guide the search towards input satisfying φ_{cond}. Silvetti et al. [32] use GP regression to approximate the STL semantics and so identify the inputs that have higher probability to have lower robustness. Apart from falsification, also in the more general context of search-based testing (SBT), the usage of GP regression has been recently advocated [20] for speeding up the search. However, to the best of our knowledge, no work in falsification or SBT uses GP regression for confidence estimation on the final result.

Our confidence measure has similarities with coverage criteria used in testing, as they both aim at giving a confidence on the absence of faults. The main difference is that coverage criteria fix a set of test requirements that need to be covered, and the coverage level acts as a confidence value; in our case, the confidence measure does not specify what needs to be covered, but implicitly considers the coverage of the input space. Coverage criteria have been proposed for falsification. Dokhanchi et al. [12] define coverage based on the blocks of Simulink models, and integrate the coverage level as part of the objective function of the falsification problem. Dreossi et al. [15] use "star discrepancy" as a

measure of input space coverage, and use it to guide the falsification search. Adimoolam et al. [1] classify inputs according to their robustness values, and define coverage based on the clustered inputs. All these works use coverage as a means for guiding the search, but they do not explicitly use it as a confidence measure. Although our confidence measure is not a coverage criterion, it has some good properties as monotonicity that would allow to also use it as stopping criterion for falsification; this usage of the measure is part of our future work.

In the verification community, other approaches provide some type of confidence on the verification results. For example, in probabilistic model checking [4, Chapter 10], the probability that a given property holds can be estimated; the approach is, however, much different from ours, as it operates on much simpler models as Markov chains.

7 Conclusion

In this paper, we have proposed a confidence measure that, in case a falsification algorithm has terminated without returning any falsifying input, estimates the probability that the specification is indeed non-falsifiable. The measure is computed by first performing a Gaussian Process (GP) Regression process that learns the robustness function from the falsification data, and by then deriving from it the probability that no falsifying input exists.

In this work, we have considered one particular kernel function for GP; as future work, we plan to investigate whether other kernel functions provide better results. The proposed approach works better if the fitness landscape is smooth (see Assumption 1); however, in some cases, the fitness landscape of the robustness could be quite complicated, in particular when output signals of different magnitudes are used in the specification (called the *scale problem* in [37–39]); as future work, we plan to devise techniques that mitigate the scale problem, and so make it easier to learn the robustness function.

The approach is applicable to hybrid systems in which Assumption 1 holds, i.e., those for which the Gaussian approximation is suitable. While this usually holds for continuous dynamics, it may not hold when discrete modes (e.g., states of a hybrid automaton) are involved. As future work, we plan to perform a much larger empirical evaluation to try to properly characterize the systems in which the assumption holds, and so the confidence measure is more trustworthy.

References

1. Adimoolam, A., Dang, T., Donzé, A., Kapinski, J., Jin, X.: Classification and coverage-based falsification for embedded control systems. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 483–503. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_24
2. Akazaki, T.: Falsification of conditional safety properties for cyber-physical systems with Gaussian process regression. In: Falcone, Y., Sánchez, C. (eds.) RV 2016. LNCS, vol. 10012, pp. 439–446. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46982-9_27

3. Annpureddy, Y., Liu, C., Fainekos, G., Sankaranarayanan, S.: S-TaLiRo: a tool for temporal logic falsification for hybrid systems. In: Abdulla, P.A., Leino, K.R.M. (eds.) TACAS 2011. LNCS, vol. 6605, pp. 254–257. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19835-9_21
4. Baier, C., Katoen, J.P.: Principles of Model Checking (Representation and Mind Series). The MIT Press (2008)
5. Balesdent, M., Morio, J., Marzat, J.: Kriging-based adaptive importance sampling algorithms for rare event estimation. Struct. Saf. **44**, 1–10 (2013)
6. Botev, Z.: The normal law under linear restrictions: simulation and estimation via minimax tilting. J. Roy. Stat. Soc. Ser. B (Stat. Methodol.) **1**(79), 125–148 (2017)
7. Broyden, C.G.: A class of methods for solving nonlinear simultaneous equations. Math. Comput. **19**(92), 577–593 (1965)
8. Corso, A., Moss, R.J., Koren, M., Lee, R., Kochenderfer, M.J.: A survey of algorithms for black-box safety validation. arXiv preprint arXiv:2005.02979 (2020)
9. Deshmukh, J., Horvat, M., Jin, X., Majumdar, R., Prabhu, V.S.: Testing cyber-physical systems through Bayesian optimization. ACM Trans. Embed. Comput. Syst. **16**(5s) (2017). https://doi.org/10.1145/3126521
10. Deshmukh, J., Jin, X., Kapinski, J., Maler, O.: Stochastic local search for falsification of hybrid systems. In: Finkbeiner, B., Pu, G., Zhang, L. (eds.) ATVA 2015. LNCS, vol. 9364, pp. 500–517. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24953-7_35
11. Dokhanchi, A., et al.: ARCH-COMP18 category report: results on the falsification benchmarks. In: 5th International Workshop on Applied Verification of Continuous and Hybrid Systems, ARCH18. EPiC Series in Computing, vol. 54, pp. 104–109. EasyChair (2018). https://doi.org/10.29007/t85q
12. Dokhanchi, A., Zutshi, A., Sriniva, R.T., Sankaranarayanan, S., Fainekos, G.: Requirements driven falsification with coverage metrics. In: Proceedings of the 12th International Conference on Embedded Software, EMSOFT 2015, pp. 31–40. IEEE Press (2015)
13. Donzé, A.: Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 167–170. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14295-6_17
14. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: Chatterjee, K., Henzinger, T.A. (eds.) FORMATS 2010. LNCS, vol. 6246, pp. 92–106. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15297-9_9
15. Dreossi, T., Dang, T., Donzé, A., Kapinski, J., Jin, X., Deshmukh, J.V.: Efficient guiding strategies for testing of temporal properties of hybrid systems. In: Havelund, K., Holzmann, G., Joshi, R. (eds.) NFM 2015. LNCS, vol. 9058, pp. 127–142. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17524-9_10
16. Ernst, G., et al.: ARCH-COMP 2020 category report: falsification. In: 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20), ARCH20. EPiC Series in Computing, vol. 74, pp. 140–152. EasyChair (2020). https://doi.org/10.29007/trr1
17. Ernst, G., et al.: ARCH-COMP 2019 category report: falsification. In: 6th International Workshop on Applied Verification of Continuous and Hybrid Systems, ARCH19. EPiC Series in Computing, vol. 61, pp. 129–140. EasyChair (2019). https://doi.org/10.29007/68dk

18. Ernst, G., Sedwards, S., Zhang, Z., Hasuo, I.: Fast falsification of hybrid systems using probabilistically adaptive input. In: Parker, D., Wolf, V. (eds.) QEST 2019. LNCS, vol. 11785, pp. 165–181. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30281-8_10
19. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theor. Comput. Sci. **410**(42), 4262–4291 (2009). https://doi.org/10.1016/j.tcs.2009.06.021
20. Feldt, R., Poulding, S.: Broadening the search in search-based software testing: it need not be evolutionary. In: Proceedings of the Eighth International Workshop on Search-Based Software Testing, SBST 2015, pp. 1–7. IEEE Press (2015)
21. Giordano, S., Gubinelli, M., Pagano, M.: Rare events of gaussian processes: a performance comparison between bridge Monte-Carlo and importance sampling. In: Koucheryavy, Y., Harju, J., Sayenko, A. (eds.) NEW2AN 2007. LNCS, vol. 4712, pp. 269–280. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74833-5_23
22. Gladisch, C., Heinz, T., Heinzemann, C., Oehlerking, J., von Vietinghoff, A., Pfitzer, T.: Experience paper: search-based testing in automated driving control applications. In: Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering. ASE 2019, pp. 26–37. IEEE Press (2019). https://doi.org/10.1109/ASE.2019.00013
23. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evol. Comput. **11**(1), 1–18 (2003)
24. Harper, A.J.: Bounds on the suprema of Gaussian processes, and omega results for the sum of a random multiplicative function. Ann. Appl. Probab.**23**(2), 584–616 (2013). https://doi.org/10.1214/12-AAP847
25. Hoxha, B., Abbas, H., Fainekos, G.E.: Benchmarks for temporal logic requirements for automotive systems. In: 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems, ARCH@CPSWeek 2014, Berlin, Germany, 14 April 2014 / ARCH@CPSWeek 2015, Seattle, USA, April 13, 2015. EPiC Series in Computing, vol. 34, pp. 25–30. EasyChair (2014)
26. Jin, X., Deshmukh, J.V., Kapinski, J., Ueda, K., Butts, K.: Powertrain control verification benchmark. In: Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control, HSCC 2014, pp. 253–262. ACM (2014). https://doi.org/10.1145/2562059.2562140
27. Li, W.V., Shao, Q.M., et al.: Lower tail probabilities for gaussian processes. Ann. Probab. **32**(1A), 216–242 (2004)
28. Marcus, M.B., Shepp, L.A., et al.: Sample behavior of gaussian processes. In: Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 2: Probability Theory. The Regents of the University of California (1972)
29. Menghi, C., Nejati, S., Briand, L., Parache, Y.I.: Approximation-refinement testing of compute-intensive cyber-physical models: an approach based on system identification. In: Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. ICSE 2020, pp. 372–384. Association for Computing Machinery, New York (2020). https://doi.org/10.1145/3377811.3380370

30. Nejati, S., Gaaloul, K., Menghi, C., Briand, L.C., Foster, S., Wolfe, D.: Evaluating model testing and model checking for finding requirements violations in Simulink models. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE 2019, pp. 1015–1025. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3338906.3340444
31. Rasmussen, C.E., Williams, C.K., Bach, F.: Gaussian Processes for Machine Learning. MIT Press (2006)
32. Silvetti, S., Policriti, A., Bortolussi, L.: An active learning approach to the falsification of black box cyber-physical systems. In: Polikarpova, N., Schneider, S. (eds.) IFM 2017. LNCS, vol. 10510, pp. 3–17. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66845-1_1
33. Yamagata, Y., Liu, S., Akazaki, T., Duan, Y., Hao, J.: Falsification of cyber-physical systems using deep reinforcement learning. IEEE Trans. Softw. Eng. (2020). https://doi.org/10.1109/TSE.2020.2969178
34. Zanette, A., Zhang, J., Kochenderfer, M.J.: Robust super-level set estimation using Gaussian processes. In: Berlingerio, M., Bonchi, F., Gärtner, T., Hurley, N., Ifrim, G. (eds.) ECML PKDD 2018. LNCS (LNAI), vol. 11052, pp. 276–291. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-10928-8_17
35. Zhang, Z., Arcaini, P., Hasuo, I.: Hybrid system falsification under (in)equality constraints via search space transformation. IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst. **39**(11), 3674–3685 (2020). https://doi.org/10.1109/TCAD.2020.3013073
36. Zhang, Z., Ernst, G., Sedwards, S., Arcaini, P., Hasuo, I.: Two-layered falsification of hybrid systems guided by Monte Carlo Tree Search. IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst. **37**(11), 2894–2905 (Nov 2018). https://doi.org/10.1109/TCAD.2018.2858463
37. Zhang, Z., Hasuo, I., Arcaini, P.: Multi-armed bandits for Boolean connectives in hybrid system falsification. In: Dillig, I., Tasiran, S. (eds.) CAV 2019. LNCS, vol. 11561, pp. 401–420. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25540-4_23
38. Zhang, Z., Lyu, D., Arcaini, P., Ma, L., Hasuo, I., Zhao, J.: Effective hybrid system falsification using Monte Carlo tree search guided by QB-robustness. In: Silva, A., Leino, K.R.M. (eds.) CAV 2021. LNCS, vol. 12759, pp. 595–618. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81685-8_29
39. Zhang, Z., Lyu, D., Arcaini, P., Ma, L., Hasuo, I., Zhao, J.: On the effectiveness of signal rescaling in hybrid system falsification. In: Dutle, A., Moscato, M.M., Titolo, L., Muñoz, C.A., Perez, I. (eds.) NFM 2021. LNCS, vol. 12673, pp. 392–399. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-76384-8_24