PART-X: A Family of Stochastic Algorithms for Search-Based Test Generation With Probabilistic Guarantees

Giulia Pedrielli, Member, IEEE, Tanmay Khandait, Member, IEEE, Yumeng Cao, Member, IEEE, Quinn Thibeault, Hao Huang, Mauricio Castillo-Effen, Member, IEEE, and Georgios Fainekos, Member, IEEE

Abstract—Requirements driven search-based testing (also known as falsification) has proven to be a practical and effective method for discovering erroneous behaviors in Cyber-Physical Systems. Despite the constant improvements on the performance and applicability of falsification methods, they all share a common characteristic. Namely, they are best-effort methods which do not provide any guarantees on the absence of erroneous behaviors (falsifiers) when the testing budget is exhausted. The absence of finite time guarantees is a major limitation which prevents falsification methods from being utilized in certification procedures. In this paper, we address the finite-time guarantees problem by developing a new stochastic algorithm. Our proposed algorithm not only estimates (bounds) the probability that falsifying behaviors exist, but also identifies the regions where these falsifying behaviors may occur. We demonstrate the applicability of our approach on standard benchmark functions from the optimization literature and on the F16 benchmark problem.

Note to Practitioners—The safety assurance problem for Cyber-Physical Systems (CPS) remains an open challenge. To demonstrate functional safety, practitioners must collect evidence that establishes that a system performs as expected under certain assumptions. The expected system behavior is typically captured through functional correctness requirements. In the case of CPS, evidence typically takes the form of test cases that are executed both on a model of the system and/or on the actual system. One of the challenges in producing such evidence is how to automatically generate test cases which are representative of the infinite execution space of CPS. Search-based test generation (SBTG) is a class of methods that can automatically generate test cases for CPS while being guided by the functional requirements. As SBTG methods try to discover test cases that invalidate, i.e., falsify, the requirements, they also collect validating, i.e., satisfying, test cases that can be used as evidence. This work introduces a method that can assess whether enough test cases have been executed given a finite testing budget. The sufficiency of the test suite is assessed by computing the probability that invalidating system behaviors may exist but have not yet been discovered. The practitioner can then adjust the number of test cases generated until a desired degree of confidence on the probability is achieved. Hence, our method not only works as an automated test case generation algorithm, but also as a method that provides formal functional performance guarantees on the system. Future directions will investigate extensions of our method to stochastic CPS.

Index Terms—Cyber physical systems, automated test generation, probabilistic guarantees, Bayesian optimization, Gaussian processes, statistical learning.

I. INTRODUCTION AND MOTIVATION

SEARCH-BASED test generation (SBTG) for Cyber-Physical Systems (CPS) [1] refers to a broad class of best-effort methods that attempt to discover system behaviors that do not satisfy a set of functional requirements. In other words, SBTG metho search the operating space of the system for behaviors that falsify (i.e., do not satisfy) the given requirements (also known as falsification process). A variety of methods for SBTG have been developed that range from tree exploration methods [2], [3], [4], [5] to black-box optimization algorithms [6], [7], [8], and related variations of these techniques [9], [10], [11], [12]. More recently, reinforcement learning [13], [14], [15] has also been explored for SBTG. The allure of SBTG techniques is that in general they do not require any information about the System under Test (SUT) and, hence, they can be easily applied to a range of challenging applications such as medical [16], automotive [17], [18], and aerospace [11], [14] (see [1], [19], [20], [21] for surveys).

A common characteristic of all the aforementioned approaches is that their goal is to discover, with as few tests as possible, a falsifying (or a most likely falsifying) behavior of the SUT [22]. In fact, SBTG methods have already shown the ability to identify counterexamples using fewer tests compared with uniform sampling approaches, and, even more importantly, they can provably find falsification where uniform sampling fails to do so [4], [6]. However, most SBTG methods (exceptions with more limited results are in [23] and [24]) cannot answer an important open question: what are the
conclusions to be drawn if no falsifying behavior has been discovered? In other words, when the test/simulation budget is exhausted and no violation has been discovered, can we conclude that the SUT is safe, or that at least it is likely safe? This is a challenging problem for almost all black-box SBTG methods since foundationally, the speedup in falsification detection is a result of smart sampling in the search space. This is a challenging problem even for SBTG approaches such as those proposed in [2] and [3], which are driven by coverage metrics over the search space. Nevertheless, a positive answer to this question is necessary if SBTG methods were ever to be adopted and incorporated into assurance procedures [25].

In this paper, we develop a general framework that can assess the probability that falsification regions exist in the search space of the SUT. Our working assumption is that the SUT can be represented as a function that takes as input a vector and returns as output a vector signal representing the (deterministic) behavior of the system. Such function can represent a software- or hardware-in-the-loop SUT, or a simulation model of the SUT. In addition, we assume that the system is checked for correctness against a formal requirement, which has a quantitative interpretation of satisfaction usually referred to as specification robustness [19], [26], [27], [28]. The robustness is positive when the system trajectory satisfies the requirement and negative otherwise. Moreover, the magnitude of the robustness represents how robustly the trajectory satisfies (or violates) the requirement. Using the robustness function, the falsification problem for CPS can be converted into a search problem where the goal is to find points that belong to the zero level set of the robustness function.

To identify the zero level set of \( f \) and, hence, estimate the probability of falsification, we propose a family of stochastic algorithms, referred to as PART-X (Partitioning with X-distributed sampling). PART-X adaptively partitions the search space \( S \) to enclose the falsifying points. The algorithm uses local Gaussian process estimates to adaptively branch and sample within the input space. The partitioning approach not only helps us identify the zero level set, but also to circumvent issues that arise due to the fact that the function \( f \) is discontinuous. In fact, the only assumption we need on \( f \) is that it is a locally continuous function. To evaluate our approach, we have built an SBTG library in Python (to be publicly released after publication) that can use the RTAMT [29] or TLTk [30] temporal logic robustness Python libraries. We demonstrate our framework on selected functions from the optimization and SBTG literature, and on the F16 benchmark problem [31]. An additional feature of our framework is that PART-X can also be utilized for evaluating test vectors generated by other SBTG tools. Therefore, PART-X can also function as an evaluation tool for other falsification algorithms in assurance procedures, or even in competitions [22].

Besides the aforementioned practical/applied contributions of our work, the PART-X framework also makes the following theoretical and technical contributions. First, it uses multiple local models as opposed to a single global process. This helps the algorithm to handle discontinuities, estimate the worst and best performing inputs, and identify disconnected zero-level sets. Second, PART-X uses a branching criterion to reduce the number of regions generated. Finally, under the aforementioned assumptions, the algorithm asymptotically achieves minimum error as a function of the minimum allowed size for a partition.

II. LITERATURE REVIEW

This work spans two macro-areas: (i) Automatic falsification of cyber-physical systems; (ii) Learning level sets of non-linear non-convex black-box functions. In the following, we first motivate and document the contribution and state-of-the-art within the field of automatic falsification through optimization-based approaches, and we identify the challenges (section II-A). Section II-B reviews techniques that can support non-stationary learning problems, with focus on the learning of a level set.

A. Search Based Test Generation Methods

The robustness of a CPS against a certain specification can be quantitatively measured using Signal Temporal Logic (STL) [26]. This can be used to formulate the CPS falsification as an optimization problem. These systems, when looked from the perspective of optimization, are notoriously known to have a highly non-linear, non-convex and often discontinuous robustness surfaces. To deal with this problem, a number of global optimization and statistical methods have been proposed that consider the CPS-under-test as a black-box and find input signals that falsify the system. From the perspective of global optimization, various approaches have been proposed that used meta heuristics including but not limited to Simulated Annealing (SA) [32], [33], Ant Colony Optimization [34], and Tabu Search [35]. Gradient descent [36], and Nelder-Meads [37] were also proposed showing however worse performance for cases with highly multi-modal rewards. To directly tackle multi-modality, cross-entropy [38] and Gaussian processes [7], [39], [40], [41] have been used as SBTG, where locations are sequentially sampled using a distribution updated with the input signals and their associated robustness values. In [42], the authors provide a global guarantee for a point estimate of the probability of a specification to hold across the input space. The proposed method entails a global and local search where the former uses Gaussian processes to build surrogates and find likely-falsifying points, while the latter tries to use the points from the global phase to search for falsifying points, if any. Our work, on the other hand, tries to find regions in the input space to give both global guarantees and local guarantees of falsification of a certain region in addition to point estimates using adaptive partitioning scheme. Similarly, Bayesian optimization (BO) is a popular black-box stochastic optimization method [43]. BO balances exploration and exploitation via surrogate modeling to produce high quality solutions in a relatively small number of iterations. However, due to the overhead costs associated to BO, such as surrogate model estimation and acquisition function optimization, this technique should be employed when observations of the objective function are expensive to collect - as in the case of observing the robustness for a given input signal. BO has proven to be quite successful over CPS falsification problems [7], [10], [44]. Recently, BO was combined with a local trust region search (the Stochastic Optimization with Adaptive Restart (SOAR))
algorithm) in an intelligent global-local optimization framework and proved highly effective for CPS falsification [45], [46]. A common aspect in Bayesian optimization algorithms is that a unique model (usually a Gaussian process) is continuously updated that guides the search towards a falsifying input. Our work is different in that we branch the original domain to find falsifications. This results into a tree of Gaussian processes and allows us to provide probabilistic guarantees also when no falsifications are found.

Falsification of CPS has also fostered the development of various platforms in the verification community that implement multiple SBTG and represent state-of-the-art benchmarking tools. S-TALIRO [32] is a MATLAB toolbox developed for monitoring and generating test cases against certain specification expressed in the form of Signal Temporal Logic (STL). The toolbox supports several optimization approaches like Simulated Annealing, the previously mentioned SOAR [45] and minSOAR [47] amongst many others [48]. PSY-TALIRO [49], the python version of S-TALIRO, is one of the newer toolboxes, which assists in automatic tests case generation using optimization procedures. It was developed from S-TALIRO and it embeds the same optimizers including our PART-X, which we described in this paper. ARIsTEO [11] works by building a surrogate of an expensive to simulate CPS using initial inputs and outputs, and attempts to find falsification directly on the surrogate model. If the falsifications found are spurious (i.e., they do not result in falsification of the real system), the surrogate model is improved before proceeding. Results were shown on publicly available non compute-intensive CPS models (i.e., Room Heating Benchmarks (RHB), Automatic Transmission (AT), Automatic Fuel Control (AFC), Insulin-Glucose Control (IGC)) and compared with S-TALIRO. Breach [37] is another MATLAB toolbox used for time series analysis and simulation-based analysis of CPS systems. FalCAuN [10] is a tool that combines falsification as an optimization black-box checking (BBC) methods for finding falsifications. It works by learning the Mealy Machine of the SUT and finding counterexamples that falsify a given property. Comparisons were drawn against Breach showing improvements in both falsification and scalability [10]. FALSTAR [50] is a falsification tool that utilizes the aLVTS to build a fast randomized falsification algorithm that takes advantage of the time-causal structure of the given problem and adapts to the local complexity. The performance of this tool was originally compared with the Breach tool (with CMA-ES algorithm) and showed superior performance in both falsification rate and time to find the first falsification on various properties over the Automatic Transmission and Powertrain Model [50]. Specifications for CPS models can contain predicate/outputs which can differ from one another in scale. FORESee [51], a tool based on Breach, tackles the scale problem by proposing QB-Robustness, which combines both quantitative robustness and classical Boolean satisfaction. It uses a Monte Carlo Tree Search to find subformulas in a specification and then performs hill-climbing optimization on the leaves in order to find a falsification.

Importantly, these tools compete in the friendly ARCH Competition every year to set the baselines for the search-based test generation community. The competition is played over several specifications based on the Automatic Transmission, Powertrain, F16, Chasing Cars, Neural Network, and Wind Turbine Simulink models [48]. In this paper we use the ARCH benchmark tools as reference to analyse the performance of our PART-X as falsification algorithm.

B. Statistical Methods for iterative Partitioning for Function Learning and Level Set Estimation

Partitioning has deep roots in exact optimization (branch and bound methods), in black-box optimization (target level set estimation), and in statistical applications such as additive modeling. In fact, partitioning is used: (i) to handle large scale data sets, by iteratively splitting the data; (ii) to handle high dimensional inputs, by iteratively splitting the original space into lower dimensional subspaces; (iii) to handle discontinuities of the reward function with respect to the input space, by iteratively branching the support of the decision space forming a partition. In this paper, we focus on the ability to estimate a surrogate for the robustness function that allows the needed flexibility to represent a function with potentially high rate of variation. Since our PART-X uses Gaussian processes (see section III-B), we are challenged by the correlation function which is assumed to be based on a stationary kernel. Partitioning allows us to change the rate of variation, through the Gaussian process correlation function, across the input space. Furthermore, we are interested in the estimation of the, potentially disconnected, set of locations that violate desired system properties. In this brief review, we focus on methods that use partitioning for level set estimation, and we highlight, when present, the type of guarantees offered by the different algorithms.

1) Surrogate Driven Approaches for Level Set Estimation: In [52], the authors propose and analyze an algorithm for the estimation of a level set for the case where the reward function can be evaluated with noise. The authors estimate a Gaussian process across the entire solution space and, at each iteration, each subregion within the current partition is further branched based on the estimated distance to the, unknown, level set, and based on the predicted variance at the centroid of the subregion. If a subregion is branched, a new location is also evaluated and more replications are ran for the subregions forming the updated partition thus updating the Gaussian process. While the scope of the paper is different from ours, the authors do not provide an error bound over the level set estimate, and use a unique model across the partition. Also, the sampling is accuracy driven and does not consider the maximization of the probability to identify the level set. Within the falsification literature, the authors [24], [53], [54] proposes to use partitioning to identify falsifying level sets. In [54], the authors provide confidence measures for the satisfaction of a property which depend on the sampling density achieved within a region. While this work uses partitioning to come up with the confidence measures, unlike our work, a local sampler is not suggested neither analyzed in terms of impacts in the convergence of the approach. We are able to provide this result for each iteration of the algorithm. More specifically, assuming a partitioning and sampling schemes are provided as input,
the authors in [24], [53], and [54] sequentially use Conformal regression to derive an estimate of the maximum and minimum function value within a subregion, with subregion-wide noise. Based on such predictions, a subregion can be eliminated, maintained (if we have confidence that the system satisfies the properties) or further branched. Conformal regression does not allow to produce a point estimate of the response thus not providing feedback for sampling decision (rather the method stops at recommending which regions to sample). On the other hand, similar to this work, the method allows to derive a probabilistic guarantee over a subregion for the correctness of eliminating and maintaining. The work in [55] is used to define the conditions over the conformal regression. Besides the lack of point estimates, another potential drawback is that the authors assume that the sampling density is defined as input. In fact, sampling densities can impact the performance of the approach, and the design of acquisition functions is central to Bayesian optimization [43]. Even if not using surrogate models, we mention Probabilistic Branch and Bound (PBNB) that uses a directed random search to approximate a target level set associated with a target quantile of the best globally optimal solutions [56], [57]. An advantage of PBNB is that it partitions the space iteratively, and performs more function evaluations in the promising regions as it refines its approximation of the target level set. In addition, there is a probabilistic bound on the error of the approximation, which we will use in the analysis of our algorithm (section IV).

2) Surrogates for Non-Stationary Responses: As mentioned, one possible criticality in using one single surrogate model is the difficulty to capture discontinuities in the function behavior. Within the statistical learning community, so-called treed Gaussian processes have been proposed with main applications in learning from non-stationary and large data sets. This literature does not address level set estimation, but it is relevant in that it addresses non-stationarity of the response and the noise. In [58] a binary tree is used to learn partitions based on Bayesian regression. The difficulty to scale the approach to high dimensional inputs/large data sets led to the computational work in [59]. One drawback of these approaches was identified in the irregularity of the variance associated with the different subregions. In particular, it was observed that some subregions tended to exhibit variances orders of magnitude larger. In [60] this problem is alleviated by using stationary processes in the different subregions (thus leading to a larger number of subregions, but better control over the variance profile). Differently, [61] deals directly with the problem of non-stationarity of the response and of the variance (heteroscedasticity) proposing a new form of Gaussian process, the Bayesian Treed Gaussian Process model. The approach combines stationary Gaussian processes and partitioning, resulting in treed Gaussian processes, and it implements a tractable non-stationary model for non-parametric regression. Along a similar line, [62] also uses a binary tree to iteratively learn different models. While the method has good fitting results, it also shows good computational performance for large data sets.

3) Reinforcement Learning Approaches: As previously mentioned, sampling can have an important impact on the ability of the search method to perform effective partitioning, i.e., branching decisions that can increase either the prediction accuracy or the probability to identify a falsifying input. In this regard, it is important to point at relevant references in the area of optimal sequential sampling. In [63], the latent action Monte Carlo Tree Search (LA-MCTS) sequentially focuses the evaluation budget by iteratively splitting the input space to identify the most promising region. Differently from our case, LA-MCTS uses non-linear boundaries for branching regions, and, like us, learns a local model to select candidates. Another relevant contribution, AlphaX, presented in [64], explores the search using distributed Monte Carlo Tree Search (MCTS) coupled with a Meta-Deep Neural Network (DNN) model that guides the search and branching decision, focusing on the sequential selection of the most promising region. In general, while relevant in terms of sampling, sequential optimization methods are designed to learn and focus on the elite region of the input space. Our premise is different: as long as a point is falsifying the user will be interested in having this information no matter how bad the falsification is.

C. Contributions

In this paper, we propose PART-X, a partitioning algorithm that relies on local Gaussian process estimates to adaptively branch and sample within the input space. PART-X brings the following innovative features over PBNB and prior falsification approaches:

1) Due to the local metamodels, as opposed to a unique global process, PART-X produces point estimates with associated predictions, for the output function. This allows to build estimates for the worst and best performing inputs, as well as the volume of falsifying sets for arbitrary levels of negative robustness.

2) The presence of a branching criterion avoids proliferation of subregions as a subregion is only branched when a criterion is satisfied (e.g., we are not able to classify the region).

3) Under mild assumptions, the algorithm asymptotically achieves minimum error as a function of the minimum allowed size for a partition. This error characterizes the classification and not only the point estimates of the probability that falsifying inputs are sampled.

4) PART-X mechanism to partition, classify and model can be used to support and evaluate any other test generation approach. This will be further detailed in section III.

We highlight that the partitioning tree and the models associated to the leaf nodes are a valuable outcome for the practitioner. Part-X could be used in a multi-stage testing setting, the practitioners may use it to quickly exclude easy-to-classify regions and proceed, for example with more expensive tests, in regions where classification is harder, i.e., where falsification is more likely.

III. PART-X: A FAMILY OF ALGORITHMS FOR PARTITIONING-DRIVEN BAYESIAN OPTIMIZATION

As previously mentioned, a system under test (SUT) can be represented as an input-output function, formally \( \mathcal{M} : \mathbb{R}^d \rightarrow (\mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^m) \), i.e., a function that takes as input a vector...
\( x \in \mathbb{R}^d \) and returns as output a vector signal \( z : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^n \) representing the (deterministic) behavior of the system. The vector \( x \) can represent system initial conditions \( x_0 \), static parameters \( p \), and/or any finite representation of an input signal \( u \) to the SUT. The vector \( x \) is usually sampled from a convex bounded space \( S \subseteq \mathbb{R}^d \), i.e., \( x \in S \), which in our case is a hypercube. These assumptions are standard in the search based test generation literature and \( M \) can represent a software- or hardware-in-the-loop SUT, or a simulation model of the SUT. In addition, we assume that the system is checked for correctness against a specification. We will refer to the formal requirement as \( \phi \), and to the robustness as the metric \( \rho_{\phi} : (\mathbb{R}_{\geq 0} \rightarrow \mathbb{R}^n) \rightarrow \mathbb{R} \cup \{-\infty, \infty\} \). The robustness \( \rho_{\phi} \) is positive when the trajectory \( z \) satisfies the requirement and negative otherwise. Moreover, the magnitude of \( \rho_{\phi} \) represents how robustly the trajectory \( z \) satisfies (or violates) the requirement. Given the robustness function \( f(x) = \rho_{\phi}(M(x)) : \mathbb{R}^d \rightarrow \mathbb{R} \), \( x \in S \), we aim at identifying its minimum. We return a prediction of the set of locations \( x \) that satisfy \( f(x) \leq 0 \), i.e., an estimate for the 0-level set of the robustness function, \( \mathcal{L}_0 \), and the associated volume. Finally, we produce a model for the robustness that can be used to predict the value of robustness at each location in the input space. The algorithm we propose is an adaptive partitioning and sampling approach that provides: (i) the probability that a falsification exists region-wide (i.e., the probability that any un-sampled location may lead to a property violation) ; (ii) the estimate of the normalized falsification volume defined as \( V^f = \nu(C_0)/\nu(S) \), where \( \nu(\cdot) \) is the volume of the relevant set. The probability in (i) is affected by the magnitude of the violation (i.e., how negative the robustness function is predicted to be), while the measure in (ii) helps engineers understanding how likely is the system to produce a falsifying behavior (i.e., convergence of the estimated level set to the true level set with probability 1).

Figure 1 gives an overview of the proposed approach, which we refer to as PART-X (Partitioning with X-distributed sampling). The algorithm sequentially and adaptively partitions the space and evaluates inputs (test vectors) to estimate a surrogate for the robustness function in each of the subregions that are sequentially generated. In particular, the surrogates we use are Gaussian processes [45], [65], [66] (used in Step 2 of the algorithm in Fig. 1), which allows us to define a variety of sampling distributions to be used in Step 3 to classify the subregions. At each iteration, a number of points is sampled in each sub-region to update the corresponding surrogate model. The sampling density depends on the fact that the subregion is classified or not. As shown in Fig. 1, the surrogate model is the key for Step 3 where the subregions are, temporarily, classified. In particular, a subregion can be either classified as positive (satisfying the property), negative (violating the property), remaining (neither satisfying nor violating), and undecided when it reaches a volume that is such to not allow further branching. Finally, if a region is currently positive or negative, it can be classified as reclassified from positive, or reclassified from negative, respectively. Only remaining and reclassified subregions are branched in the subsequent iteration (Step 4 in Fig. 1), in case their volume is larger than the minimum branchable volume, which the user can provide as input information. When the volume reaches the minimum branchable value, the subregion is treated as undecided and cannot be further branched. As previously mentioned classified regions are sampled with different density from non classified. The classified subregions are given an overall sampling budget of \( n_C \) observations to be distributed proportionally to the volume. The rest of the subregions are each given a budget of \( \max\{n_0 - n_{ijk}, 0\} + n_{BO} \) samples, where \( n_0 \) is the minimum number of points to build a Gaussian process in a subregion, \( n_{ijk} \) is the number of points sampled in the subregion so far, and \( n_{BO} \) is the sampling budget for Bayesian optimization-based sampling. The algorithm terminates when the budget is exhausted. Section III-A introduces the main notations and definitions used in the PART-X Algorithm, Section III-B presents the basic definitions for Gaussian processes, which we use to produce predictions of the robustness function, section III-C, introduces the scheme followed by PART-X to iteratively branch, sample, update subregion models and decide whether to classify each of the subregions.

### A. PART-X Tree Notation and Definitions

Let us first introduce a working example that can facilitate the understanding of the algorithm and the adopted notation.

**Working Example.** As an example consider the robustness landscape in Fig. 2(a), and the corresponding 0-level set.
and finally as violating (iterations classified, first as satisfying (iterations $k = 1$) to property violation. For regions that get classified (iteration $k = 3$), we see uniform samples are taken. As more branching is performed, we observe that more regions are tentatively classified, first as satisfying (iterations $k = 1, 3$; Fig.3(a)-3(b)) and finally as violating (iterations $k = 7, 8$; Fig.3(c)-3(d)).

In general, at each iteration $k$, the PART-X tree may have the following possible subregion (leaf) types:

- New satisfying region ($\sigma_{jk}^S$): for each level $j = 1, \ldots, k$, at each iteration $k$, the new set of satisfying regions is the union of the new satisfying subregions $\sigma_{jk}^S = \bigcup_j \sigma_{ijk}^S$, formed by new $N_{jk}^+$ subregions;
- New positive reclassified region ($\sigma_{jk}^P$): for each level $j = 1, \ldots, k$, at each iteration $k$, the new set of regions reclassified from positive is the union of the new reclassified subregions $\sigma_{jk}^P = \bigcup_j \sigma_{ijk}^P$, formed by new $N_{jk}^P$ subregions;
- New violating region ($\sigma_{jk}^V$): for each level $j = 1, \ldots, k$, at each iteration $k$, the new set of regions classified as violating is the union of the new violating subregions is $\sigma_{jk}^V = \bigcup_j \sigma_{ijk}^V$, formed by new $N_{jk}^V$ subregions;
- New negative reclassified region ($\sigma_{jk}^N$): for each level $j = 1, \ldots, k$, at each iteration $k$, the new set of regions reclassified from violating is the union of the new reclassified subregions $\sigma_{jk}^N = \bigcup_j \sigma_{ijk}^N$, formed by new $N_{jk}^N$ subregions;
- New remaining region ($\sigma_{jk}^R$): for each level $j = 1, \ldots, k$, at each iteration $k$, the new set of regions remaining is the union of the new remaining subregions $\sigma_{jk}^R = \bigcup_j \sigma_{ijk}^R$, formed by new $N_{jk}^R$ subregions. The new remaining subregions (e.g., $\sigma_{ijk}^R \in \sigma_{ijk}^R$ in Fig.3(b)) are branched in the subsequent iteration to form the new subregions (e.g., $\sigma_{ijk}^R \in \sigma_{ijk}^R$).

- New unclassified region ($\sigma_{jk}^U$): for each level $j = 1, \ldots, k$, at each iteration $k$, the new set of regions unclassified is the union of the new unclassified subregions $\sigma_{jk}^U = \bigcup_j \sigma_{ijk}^U$, formed by new $N_{jk}^U$ subregions. Different from the remaining subregions, unclassified subregions are not branchable in any direction, since they have reached the minimum volume.

More specifically, we define the concept of branchable dimension that, as, previously mentioned, differentiates between the remaining and undecided regions.

**Definition 1 (Branchable Dimension):** Given a subregion $\sigma_{ijk} \subseteq S \subseteq \mathbb{R}^d$, this is branchable along direction $h = 1, \ldots, d$ if the following holds:

$$\Lambda_h(\sigma_{ijk}) > \delta_v X_h,$$

where $\Lambda_h(\sigma_{ijk})$ is the operator that returns the length of the subregion $\sigma_{ijk}$, along the $d$-th dimension. Since $\sigma_{ijk}$ are hyper-rectangles, such length always exists and is finite. On the right hand side, $\delta_v$ is an input parameter, and $X_h$ is the maximum length of the input search space along dimension $h$.

As a result, a subregion becomes undecided when all its dimensions are not branchable. Once a subregion is formed, it keeps being sampled until it is branched, after which, the previously evaluated locations will be re-distributed to the child-nodes (new subregions). As a result, if we refer to $n_{ijk}$ as the number of locations sampled in the subregion $i$ at level $j$ up to iteration $k$, these will be assigned to the subregion $\sigma_{ijk}$, where, in general, $\ell \leq k$. More specifically, to estimate a surrogate model in a subregion, we will require $n_{ijk} \geq n_0$, where $n_0$ is given as input. If a subregion is classified, we will sample a fraction of $nC$ locations, proportionally to the volume of the subregion. On the other hand, if the subregion is not classified, we will sequentially sample $n_{BO}$ points, in each subregion, using a Bayesian optimization scheme.

Given the previous definitions, at the $k^{th}$ iteration, there will be a number $N_0 = \left( N_k^+ + N_{r}^+ + N_k^- + N_{r}^- + N_k^t + N_{r}^t \right)$.
μ is the constant process mean, and \( Z(x) \sim GP(0, \tau^2 R) \), with \( \tau^2 \) being the constant process variance and \( R \) the correlation matrix. Under the Gaussian correlation assumption, given \( n \) points, \( R_{nm} = \prod_{t=1}^{d} \exp(-\theta_t |x_m - x_n|) \), for \( h, m, = 1, \ldots, n \). The \( d \)-dimensional vector of hyperparameters \( \theta \) controls the smoothing intensity of the predictor in the different dimensions. The parameters \( \mu \) and \( \tau^2 \) are estimated through maximum likelihood [65]:

\[
\hat{\tau}^2 = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - \mu)^2
\]

where \( \mu \) is the constant process mean, and \( Z(x) \sim GP(0, \tau^2 R) \), with \( \tau^2 \) being the constant process variance and \( R \) the correlation matrix. Under the Gaussian correlation assumption, given \( n \) points, \( R_{nm} = \prod_{t=1}^{d} \exp(-\theta_t |x_m - x_n|) \), for \( h, m, = 1, \ldots, n \). The \( d \)-dimensional vector of hyperparameters \( \theta \) controls the smoothing intensity of the predictor in the different dimensions. The parameters \( \mu \) and \( \tau^2 \) are estimated through maximum likelihood [65]:

\[
\hat{\tau}^2 = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - \mu)^2
\]

where \( \mu \) is the constant process mean, and \( Z(x) \sim GP(0, \tau^2 R) \), with \( \tau^2 \) being the constant process variance and \( R \) the correlation matrix. Under the Gaussian correlation assumption, given \( n \) points, \( R_{nm} = \prod_{t=1}^{d} \exp(-\theta_t |x_m - x_n|) \), for \( h, m, = 1, \ldots, n \). The \( d \)-dimensional vector of hyperparameters \( \theta \) controls the smoothing intensity of the predictor in the different dimensions. The parameters \( \mu \) and \( \tau^2 \) are estimated through maximum likelihood [65]:

\[
\hat{\tau}^2 = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - \mu)^2
\]

where \( \mu \) is the constant process mean, and \( Z(x) \sim GP(0, \tau^2 R) \), with \( \tau^2 \) being the constant process variance and \( R \) the correlation matrix. Under the Gaussian correlation assumption, given \( n \) points, \( R_{nm} = \prod_{t=1}^{d} \exp(-\theta_t |x_m - x_n|) \), for \( h, m, = 1, \ldots, n \). The \( d \)-dimensional vector of hyperparameters \( \theta \) controls the smoothing intensity of the predictor in the different dimensions. The parameters \( \mu \) and \( \tau^2 \) are estimated through maximum likelihood [65]:

\[
\hat{\tau}^2 = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - \mu)^2
\]

where \( \mu \) is the constant process mean, and \( Z(x) \sim GP(0, \tau^2 R) \), with \( \tau^2 \) being the constant process variance and \( R \) the correlation matrix. Under the Gaussian correlation assumption, given \( n \) points, \( R_{nm} = \prod_{t=1}^{d} \exp(-\theta_t |x_m - x_n|) \), for \( h, m, = 1, \ldots, n \). The \( d \)-dimensional vector of hyperparameters \( \theta \) controls the smoothing intensity of the predictor in the different dimensions. The parameters \( \mu \) and \( \tau^2 \) are estimated through maximum likelihood [65]:

\[
\hat{\tau}^2 = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - \mu)^2
\]

where \( \mu \) is the constant process mean, and \( Z(x) \sim GP(0, \tau^2 R) \), with \( \tau^2 \) being the constant process variance and \( R \) the correlation matrix. Under the Gaussian correlation assumption, given \( n \) points, \( R_{nm} = \prod_{t=1}^{d} \exp(-\theta_t |x_m - x_n|) \), for \( h, m, = 1, \ldots, n \). The \( d \)-dimensional vector of hyperparameters \( \theta \) controls the smoothing intensity of the predictor in the different dimensions. The parameters \( \mu \) and \( \tau^2 \) are estimated through maximum likelihood [65]:

\[
\hat{\tau}^2 = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - \mu)^2
\]

where \( \mu \) is the constant process mean, and \( Z(x) \sim GP(0, \tau^2 R) \), with \( \tau^2 \) being the constant process variance and \( R \) the correlation matrix. Under the Gaussian correlation assumption, given \( n \) points, \( R_{nm} = \prod_{t=1}^{d} \exp(-\theta_t |x_m - x_n|) \), for \( h, m, = 1, \ldots, n \). The \( d \)-dimensional vector of hyperparameters \( \theta \) controls the smoothing intensity of the predictor in the different dimensions. The parameters \( \mu \) and \( \tau^2 \) are estimated through maximum likelihood [65]:

\[
\hat{\tau}^2 = \frac{1}{n} \sum_{i=1}^{n} (f(x_i) - \mu)^2
\]
Improvement \(\text{EI}(x)\), namely [67]:

\[
x_{r+1} \in \arg\max_{x \in \sigma_{ij}^r} \text{EI}(x) = \max \left\{ \left[ f^* - \hat{Y}_{ijk}(x) \right] \Phi \left( \frac{f^* - \hat{Y}_{ijk}(x)}{\sigma_{ij}^r(x)} \right) + \hat{\sigma}_{ijk}(x) \phi \left( \frac{f^* - \hat{Y}_{ijk}(x)}{\sigma_{ij}^r(x)} \right), 0 \right\}.
\]

(3)

where \(f^*\) is the best function value sampled so far in subregion \(\sigma_{ij}^r\), \(\Phi\) is the normal distribution function, and \(\phi\) the normal probability density function. Once the point has been sampled, we update the Gaussian process, and proceed until \(n_{BO}\) evaluations have been performed. We then proceed verifying the branching conditions and possibly updating the partition.

On the other hand, subregions temporarily classified as satisfying \((\Theta_{ij})\) or violating \((\Theta_{ij})\) receive an overall evaluation budget of \(n_i\) locations to be distributed across all the subregions in this group. In order to perform such distribution of evaluations, we consider the Gaussian Process predictor \(\hat{Y}_{ijk}\), and we add samples to each subregion in this group using the following metric:

\[
I_{ijk} = \frac{1}{v(\sigma_{ij}^r)} \int_{\sigma_{ij}^r} \int_{-\infty}^0 f_{ijk}(\hat{Y}_{ijk}(x_0)) d\gamma d\sigma_{ij}^r(x_0).
\]

(4)

where \(x_0 = x_0 \in \sigma_{ij}^r\). The basic idea behind the metric in (4) is to sample a region proportionally to the cumulated density below 0 of the Gaussian process. The constant \(v(\sigma_{ij}^r)\) is used to normalize the indicator so that \(I_{ijk} \in (0, 1)\).

An overview of the procedure for the sampling phase for the non classified regions is reported in Algorithm 1. In the algorithm, \(n_{ijk}\) is the cumulated number of evaluations in each subregion at level \(j\) of the partitioning tree at iteration \(k\).

**Algorithm 1: Sequential Subregion Sampling With Bayesian Optimization (SampleBO)**

1: \textbf{Input:} Subregion \(\sigma_{ijk}^r \subset \mathbb{R}^d\), objective function \(f(x)\), initialization budget \(n^0\), total budget \(n_{BO}\), \(n_{ijk}\) locations sampled so far \((x_{ijk}, f_{ijk})\);
2: \textbf{Output:} best location and value \(x^*_{ijk} \in \sigma_{ijk}^r\), \(f(x^*_{ijk})\), final Gaussian process model \((\hat{Y}_{ijk}(x), \hat{\sigma}_{ijk}(x))\);
3: \textbf{Step 1:} Compute the initial required evaluation budget:
4: \textbf{if} \(n_{ijk} \geq n_0^0\) \textbf{then}
5: \textbf{Use} \(n_{ijk}\) sampled points within the subregion as initializing points for the Gaussian process estimation; \(t \leftarrow 0\);
6: \textbf{else}
7: \textbf{Sample} \(n_0 - n_{ijk}\) points using a Latin Hypercube design. Return \(x_{train} \in \sigma_{ijk}^r\), \(f(x)\), \(\forall x \in x_{train}\); \(t \leftarrow n_0 - n_{ijk}\);
8: \textbf{end if}
9: \textbf{while} \(t < n_{BO}\) \textbf{do}
10: \textbf{Step 2.1:} Estimate the GP using the training data \([x_{train}, y_{train}]\). return \((\hat{Y}_{ijk}(x), \hat{\sigma}_{ijk}(x))\);
11: \textbf{Step 2.2:} Select the next location \(x^*_{El} \leftarrow \arg \max_{x \in \mathcal{X}} \text{EI}(x)\); Evaluate and store \(f(x^*_{El})\).
12: \textbf{Step 2.3:} \(t \leftarrow t + 1\)
13: \textbf{end while}

2) Classification Scheme: At the end of the sampling stage, we have a number of Gaussian processes equal to the number of leaves in the tree associated to the \(k\)th iteration, and we need to estimate the \(\delta_C\)-quantile for the minimum and maximum value of the function in each of the subregions (tree leaves). In order to do so, we use the Monte Carlo procedure in Algorithm 2. Specifically, we first obtain a point estimate for the minimum and maximum quantiles (line 7 of the algorithm) using the fact that, conditional on a sample \(x\), the function value is distributed as a normal random variable with mean corresponding to the predicted value and variance equal to the model variance. Hence, we sample \(M\) locations, evaluate the prediction and model variance (line 5), and we obtain a point estimate for \((\bar{g}, (\delta_C), g, (\delta_C))\). Running the procedure \(R\) times returns the mean and variance estimates for the same quantiles (line 9). Here \(\delta_C\) refers to the significance chosen for the estimation of the quantile metrics (the subscript \(C\) stands for “classification”).

**Algorithm 2: Gaussian Process Based Min-Max Quantiles Estimation (MCstep)**

1: \textbf{Input:} subregion \(\sigma_{ijk}^r \subset \mathbb{R}^d\), objective function \(f(x)\), Gaussian process model \((\hat{Y}_{ijk}(x), \hat{\sigma}_{ijk}(x))\). Number of Monte Carlo iterations \(R\), number of evaluations per iteration \(M\);
2: \textbf{Output:} Estimates for the minimum and maximum \(\delta_C\)-quantiles of the minimum and maximum function value \(\hat{\gamma}_j(\delta_C), \hat{\gamma}_j(\delta_C), \hat{\gamma}_j(\delta_C), \hat{\gamma}_j(\delta_C)\);
3: \textbf{for} \(r = 1, \ldots, R\) \textbf{do}
4: \textbf{for} \(m = 1, \ldots, M\) \textbf{do}
5: \textbf{Sample} \(x_{mr}\), evaluate \((\hat{Y}_{ijk}(x_{mr}), \hat{\sigma}_{ijk}(x_{mr}))\);
6: \textbf{end for}
7: \textbf{Minimum and maximum quantile:}
8: \textbf{for} \(r = 1, \ldots, R\) \textbf{do}
9: \textbf{Minimum and maximum \(\delta_C\)-quantile:}
10: \textbf{end for}
11: \textbf{end for}

Once the estimation procedure is complete, we may classify each subregion \(\sigma_{ij}^r\), with \(\ell \leq k\) based on the estimates returned by Algorithm 2. As shown by Algorithm 3, these quantities are used differently based on the current type of the region. If the region is currently remaining, i.e., \(\gamma = r\), then we check whether we can classify it as satisfying, i.e., \(\gamma = +\), or violating, i.e., \(\gamma = -\), based on whether the lower bound of the minimum quantile is above zero, or the upper bound of the maximum quantile is below zero, respectively.
Differently if the region is currently classified, i.e., $\gamma = +$ ($\gamma = -$) we only check whether the condition that classified it is now violated. If this happens, then the region becomes $\gamma = +$ ($\gamma = -$), and it will be branched, thus becoming all remaining subregions $\gamma = r$.

Algorithm 3 Classification of a Subregion (Classify)

1: Input: subregion $\sigma_{ijk} \subset \mathbb{R}^d$, current subregion type $\gamma \in (+, -, r)$. Gaussian process model $(\hat{y}_{ijk}(x), \hat{\sigma}_{ijk}^2(x))$.
2: Output: Return region type $(r, +, -)$.
3: if $\nu(\sigma_j) \leq \prod_{d=1}^D \delta_d \cdot \bar{X}_d$ then
4: $\gamma = u$;
5: else if $\gamma = +$ then
6: if $\hat{Q}_j(\delta_C) - Z_{1-\delta_C}/2 \text{Var}(\hat{Q}_j(\delta_C)) \leq 0$ then
7: $\gamma = r$;
8: else
9: $\gamma = +$;
10: end if
11: else if $\gamma = -$ then
12: if $\hat{Q}_j(\delta_C) + Z_{1-\delta_C}/2 \text{Var}(\hat{Q}_j(\delta_C)) \geq 0$ then
13: $\gamma = r$;
14: else
15: $\gamma = -$;
16: end if
17: else if $\gamma = r$ then
18: if $\hat{Q}_j(\delta_C) + Z_{1-\delta_C}/2 \text{Var}(\hat{Q}_j(\delta_C)) < 0$ then
19: $\gamma = -$;
20: else if $\hat{Q}_j(\delta_C) - Z_{1-\delta_C}/2 \text{Var}(\hat{Q}_j(\delta_C)) > 0$ then
21: $\gamma = +$;
22: else
23: $\gamma = r$;
24: end if
25: end if

Finally, Algorithm 4 shows the overall PART-X. Each time a subregion is classified as either satisfying or violating the requirements, the remaining region ($\Theta^r$) is updated by removing the classified subregions. On the other hand, regions that are reclassified are treated as remaining regions (lines 6-7 in Algorithm 4). Given the remaining subregions $\Theta^r$, a branching algorithm is called that randomly selects a direction and cuts each subregion along that dimension into B subregions of equal volume. The partition function is referred to as $P: A \rightarrow (A_i) : \bigcup_i A_i = A, \bigcap_i A_i = \emptyset$ in Algorithm 4. In this manuscript, we use a simple algorithm that branches one dimension at each iteration bisectioning any interval. We allow subregions to be branched in direction $h = 1, \ldots, d$ only if the subregion is branchable along the $h$-th dimension, according to Definition 1. PART-X terminates when the maximum number of function evaluations has been reached. We refer to this as the total evaluation budget $T$.

3) Partitioning With X-Distributed Sampling: The procedure in Algorithm 4 summarizes the phases of the proposed approach. In the algorithm, we use the notation $v(\cdot)$ to refer to the volume of a region. Since the regions in PART-X are hyperboxes, volumes are easily computable.

Algorithm 4 Partitioning With Continued X-Distributed Sampling

1: Input: Input space $S$, function $f(x)$, initialization budget $n_0$, Bayesian optimization budget, $n_{BO}$, and unclassified subregions budget $n_u$, total number of evaluations $T$. Define branching operator $P: A \rightarrow (A_i) : \bigcup_i A_i = A, \bigcap_i A_i = \emptyset$. Number of Monte Carlo iterations $R$, number of evaluations per iteration $M$; number of cuts per dimension per subregion $B$; classification percentile $\delta_C, \delta_i$.
2: Set the iteration index $k \leftarrow 1$. Initialize the sets $\Theta^{-} \equiv \Theta^{-} = \Theta^{+} = \Theta_{-}^i = \emptyset, \Theta^{+} \equiv \Theta$; $S$.
3: Output: $\Theta^{-}, \Theta^{+}, \Theta_{-}^i, \Theta, \Theta^{+}$.

4: while $T_k \geq 0$ do
5: Branching
6: for $\sigma_{ijk} \in (\Theta^{-} \cup \Theta^{+} \cup \Theta_{-}^i)$ do
7: Evaluate $\sigma_{ijk} \in (\Theta^{-} \cup \Theta^{+} \cup \Theta_{-}^i)$
8: end for
9: Count $N_k^C \leftarrow$ number of non-classified leaves of the partitioning tree at iteration $k$; $N_k^C \leftarrow$ number of classified leaves of the partitioning tree at iteration $k$;
10: if $T_k \geq n_{BO} \cdot N_k^C + \sum_{\sigma_{ijk} \in (\Theta^{-} \cup \Theta^{+} \cup \Theta_{-}^i)} \max(n_{jk} - n_0, 0)$ then
11: for All unclassified subregions $\sigma_{ijk}, \gamma(i,j)$ do
12: Execute SampleBO($\sigma_{ijk}, n_{ijk}, n_{ijk}$);
13: Return the quantiles for the minimum and maximum function value executing MCstep($R, M, \sigma_{ijk}, \hat{y}_{ijk}, \hat{\sigma}_{ijk}$);
14: Update subregions:
15: $\text{Classify}(\sigma_{ijk}, \hat{y}_{ijk}, \hat{\sigma}_{ijk})$, $\Theta^r \leftarrow \Theta^r \cup \sigma_{ijk}$
16: $\Theta^r \leftarrow \Theta^r \setminus \sigma_{ijk}$ \quad $N_k^+ \leftarrow N_k^+ + 1$
17: $n_{ijk} \leftarrow n_{ijk} + n_{BO} + \max(n_{ijk} - n_0, 0)$;
18: $T_k \leftarrow T_k - n_{BO} - \max(n_{ijk} - n_0, 0)$;
19: end for
20: for all subregions type $\gamma \in \{+,-\}$ do
21: Allocate $n_{ijk}$, across the subregions proportionally to the probability metric in eqn. (4), update $n_{ijk}$;
22: Return the quantiles for the minimum and maximum function value executing MCstep($R, M, \sigma_{ijk}, \hat{y}_{ijk}, \hat{\sigma}_{ijk}$);
23: Update subregions type:
24: $\text{Classify}(\sigma_{ijk}, \hat{y}_{ijk}, \hat{\sigma}_{ijk})$, $\Theta^{+} \leftarrow \Theta^{+} \cup \sigma_{ijk}$
25: $\Theta^{+} \leftarrow \Theta^{+} \setminus \sigma_{ijk}$;
26: end for
27: Evaluate the function at the sampled point and update the Gaussian processes;
28: $T_k \leftarrow 0$;
29: end if
30: $k \leftarrow k + 1$;
31: end while
32: Return $v = \frac{v(\Theta^{-} \cup \Theta^{+} \cup \Theta_{-}^i)}{v(\Theta)}$.

Remark 1 (PART-X as evaluation tool): As mentioned in section II-C, PART-X can be used by any test generation to evaluate it in its ability to identify falsifying regions. In order to do so Algorithm 4 can be executed without sampling. In fact, at each iteration, the locations are made available from the sampling function of the search algorithm of choice. As a result, PART-X will only branch based on the robustness values sampled by the method under evaluation, and, in the process, a subregion may be classified. Once either there
are less than \( n_0 \) points in a subregion, or the subregion has reached the minimum volume, the branching stops. Doing so, the falsification volume will be computed and different test generation tools can be compared.

IV. PART-X THEORETICAL ANALYSIS

Figure 5 shows the 7th, and 8th iterations of PART-X applied to the 2-d Himmeblau’s function (section V-A). The Figure 5 highlights the main quantities at the core of our algorithm analysis, i.e., the concept of mis-classification volumes for the classified regions (for brevity referred to as \( \Delta^+_k \) and \( \Delta^-_k \) in Figure 5 for the satisfying and violating region, respectively), and mis-classification events \( C^+_k \), and \( C^-_k \). The following results provide bounds, with associated guarantees, of the mis-classification volume at each iteration of the algorithm. We start providing an important result from the literature that allows us to analyse the error within a single subregion. Based on this result, we bound the mis-classification error and recovery of such error, both for satisfying as well as violating subregions. Such bound is probabilistic in nature and it is defined at each level of the partitioning tree, and for each iteration of the algorithm (Lemmas 3-4). We then extend these results to the overall error in Theorems 1-2, while considering a bound on the error to be true. We generalize the results removing the assumption on the error bound in Theorems 3-4.

A. Assumptions and Definitions

In the following, we provide the assumptions underlying the theoretical analysis of the PART-X algorithm.

Assumption 1: \( S \) is a compact space.

Since PART-X is applied to analyse the physical behavior of CPS’s inputs, they are usually interpreted as initial conditions and controls, hence assumption 1 is common.

Assumption 2: \( f(x) \) is locally smooth, i.e., there exists a collection of subregions with positive Lebesgue measure such that \( f(x) \) is smooth within each subregion.

While assumption 2 may be globally hard to satisfy, when applied to the subregions, the robustness function is the result of the dynamics of a system, hence locally large behavioral changes are unlikely (otherwise the region will be branched). Hence, this is an acceptable assumption. In fact, the algorithm empirically performs well in presence of switching dynamics (e.g., AT Benchmark in Section V-C). This assumption supports the use of Gaussian processes as local model forms for each subregion.

Assumption 3: The hyperparameters \( \tau \), \( \theta \) of the Gaussian process model are assumed to be known.

This assumption is common in the analysis of surrogate based approaches. Nonetheless, the performance of the algorithm is empirically robust to the hyperparameters being estimated.

Assumption 4: The initial sample set of \( n_0 \) points \( \{x_i\}_{i=1}^{n_0} \), produces a Gaussian process model that satisfies cross-validation.

This is a common assumption in the analysis of surrogate based optimization approaches. Nonetheless, empirically, our PART-X shows performance that is robust to the violation of cross-validation of the models.

Assumption 5: The true function to be optimized, \( f \), is bounded over \( S \).

This assumption holds under practical restrictions on the structure of the specification. Namely, the specification should not contain the constants TRUE or FALSE, and the duration of the test should be longer than the temporal horizon of the formula (see [68] for details). In addition, we assume that any sets in the specification are non-empty and that they are not the whole universe, and therefore a distance to such a set is always finite. This is an assumption on the correctness/usefulness of the STL formula.

Definition 2 (Significance): Let \( \alpha_j \), be the significance of a probabilistic statement at level \( j \) of the partitioning tree. Then, given any \( \alpha > 0 \), we will have that the significance at level \( j \) satisfies:

\[
\alpha_j = \begin{cases} 
\alpha & \text{if } j = 1 \\
\alpha_{j-1}/B & \text{if } j > 1 
\end{cases}
\]  (5)

Definition 3: We will refer to the 0-level set \( \mathcal{L}_0 \) as

\[
\mathcal{L}_0 = \{ x \in S : f(x) \leq 0 \}
\]  (6)

Definition 4 (\( \epsilon_{jk}^+, \epsilon_{jk}^-, \Theta_{jk}^+, \Theta_{jk}^- \)): Let \( \epsilon_{jk}^+ = v(\Theta_{jk}^+ \cap \mathcal{L}_0), j = 1, \ldots, k \) denote the volume incorrectly classified as \( \Theta^+ \) at the \( j^{th} \) level of the partitioning tree, up to the \( k^{th} \) iteration of the algorithm. Then, \( \epsilon_{jk}^+ = \sum_{j=1}^k v(\Theta_{jk}^+ \cap \mathcal{L}_0) \) will be the volume incorrectly classified as \( \Theta^+ \) up to the \( k^{th} \) iteration of the algorithm. Equivalently, \( \Theta_{jk}^+ = \bigcup_{j=1}^k \Theta_{jk}^+ \).

Definition 5 (\( \epsilon_{jk}^-, \Theta_{jk}^- \)): Let \( \epsilon_{jk}^- = v(\Theta_{jk}^- \cap \mathcal{L}_0), j = 1, \ldots, k \) denote the volume incorrectly classified as \( \Theta^- \) at the \( j^{th} \) level of the partitioning tree, up to the \( k^{th} \) iteration of the algorithm. Then, \( \epsilon_{jk}^- = \sum_{j=1}^k [v(\Theta_{jk}^- \cap \mathcal{L}_0)] \) will be the volume incorrectly classified as \( \Theta^- \) up to the \( k^{th} \) iteration of the algorithm. Equivalently, \( \Theta_{jk}^- = \bigcup_{j=1}^k \Theta_{jk}^- \).
In the first part of the proof, we will assume that (level formulations will also be used applying definitions 4-5).

\[ 0 \leq \epsilon_k^+ \leq \frac{v(\hat{\Theta}_{k}^{+})}{v(S)}, \quad 0 \leq \epsilon_k^- \leq \frac{v(\hat{\Theta}_{k}^{-})}{v(S)}. \]

where the user-defined parameter \( \epsilon \) represents, for each sub-region, the volume that the user tolerates to be wrongly classified.

**Definition 6 (Misclassification Events):** Let \( C_k = (C_k^+ \cap C_k^-) \) denote the mis-classification event at the \( k \)th iteration to be defined as:

\[
C_k^+ = \left\{ v(\Theta_k^+ \cap L_0) \leq \frac{\epsilon v(\hat{\Theta}_k^+)}{v(S)} \right\}.
\]

\[
C_k^- = \left\{ v(\Theta_k^- \cap L_0) \geq \frac{\epsilon v(\hat{\Theta}_k^-)}{v(S)} \right\}. \tag{7}
\]

let event \( C_k = (C_k^+ \cap C_k^-) \) ensure that the volume of incorrectly classified subregions satisfy the upper bound.

**PART-X** returns \( \hat{\Theta}_k \), as an estimate for the true, unknown 0-level set \( L_0 = \{ x \in S : f(x) \leq 0 \} \). In the following analysis we will consider for each subregion at level \( j \), iteration \( k \) of the algorithm an accumulated number of observations \( n_{jk} \leq n_{jk} \forall i \). This value is solely a function of the level and the iteration index, and it represents a lower bound to the actual number of observations accumulated at each subregion. This allows us to avoid keeping memory of all the re-classifications a subregion has undergone.

**B. Main Results**

The first result we provide exploits the guarantee of error bound for biased sampling distributions discussed in [69] (Thm. 1, p. 358). The result allows us to express the error rate as a function of the sampling probability in each subregion.

**LEMMA 1 (General PAC Guarantee for Subregion Classification):** Let us refer to \( \mathcal{H} \) as the classification of the subregion \( \hat{\sigma}_{ijk} \). Then, we have for \( \alpha_i > 0 \), that with probability at least \( 1 - \alpha_i \) over the choice of a sample of \( n_{jk} \) observations, the subregion \( \hat{\sigma}_{ijk} \) is consistent with the sample and the subregion is measurable \( \mathbb{P}(\hat{\sigma}_{ijk} > 0) > 0 \), satisfies the following.

\[
\epsilon_{ijk}^+ \leq \frac{\ln \frac{1}{\mathbb{P}(\sigma_{ijk} > 0)} + \ln \frac{1}{\alpha} + 2 \ln n_{jk} + 1}{n_{jk}}.
\]

where, the error rate \( \epsilon_{ijk}^+ \) associated to \( \hat{\sigma}_{ijk} \) is defined for a given desired hypothesis (target) \( \mathcal{H} \) to be the probability over the choice of the input \( x \) over \( \hat{\sigma}_{ijk} \) disagrees with \( \mathcal{H} \).

**Proof:** This Lemma is a readaptation of the result in Thm. 1, p. 358 of [69], therefore we omit the proof here. \( \square \)

In this analysis, we will be particularly interested in classification of subregions as violating, satisfying, violating reclassified, satisfying reclassified, remaining or unclassified (such classifications are referred to as \( \mathcal{H} \) in the Lemma. The bound over the error rate in classifying each subregion is a function of the significance \( \alpha_i > 0 \), which depends on the level of the subregion within the classification tree, the probability associated to samples in the region, which depends on the sampling distribution produced by **PART-X** \( (P(\hat{\sigma}_{ijk} > 0)) \), and the number of samples in the subregion. This error rate is used to bound the probability to misclassify the new subregions that **PART-X** produces at each iteration, at each level of the partitioning tree. The error rate is fundamental to quantify a bound over the volume of misclassified inputs which will be the focus of Lemmas 2-5.

**Lemma 2 (Positive Classified Error Bound):** Let \( L_0 \) denote the 0-level set defined in (6), and \( \hat{\sigma}_k \) be the set of subregions classified as satisfying at the \( j \)th level of the **PART-X** tree, at the \( k \)th iteration of the algorithm. Refer to \( \hat{\sigma}_k \) as the set of subregions classified as satisfying at the \( k \)th iteration of the algorithm. Assume the event \( C_k \) in Definition 6 to be true. Then, the following holds for the volume misclassified as satisfying at the \( k \)th iteration of **PART-X**:

\[
P(\hat{\sigma}_{ijk}^+ \cap L_0) \leq N_{jk}^+ \epsilon_{ijk}^+ | C_k | \geq \prod_{r=1}^{N_{jk}^+} [1 - \delta_{ijk}]. \tag{8}
\]

where, \( \hat{\sigma}_{ijk}^+ \cap L_0 \) is the volume of new regions misclassified at level \( j \) of the partitioning tree by iteration \( k \), and \( \epsilon_{ijk}^+ \) intuitively represents the subregion error bound and it is therefore multiplied by the number of novel classified regions \( N_{jk}^+ \). Under event \( C_k^+ \), \( 0 \leq \epsilon_{jk}^+ \leq \frac{\epsilon v(\hat{\Theta}_k^+)}{v(S)} \), and \( \delta_{ijk} = \frac{\ln \frac{1}{\mathbb{P}(\sigma_{ijk} > 0)} + \ln \frac{1}{\alpha} + 2 \ln n_{jk} + 1}{n_{jk}} \) represents the probability to make a classification mistake according to the bound in Lemma 1. Since the sampling across subregions in each level and across levels is independent the overall error probability can be estimated as the product of the subregion-classification probabilities, as shown in the following.

\[
P(\hat{\sigma}_{ijk}^+ \cap L_0) \leq N_{jk}^+ \epsilon_{ijk}^+ | C_k | \geq \prod_{j=1}^{k} \prod_{i=1}^{N_{jk}^+} [1 - \delta_{ijk}]. \tag{9}
\]

where, under event \( C_k^+ \), \( 0 \leq \epsilon_{jk}^+ \leq \frac{\epsilon v(\hat{\Theta}_k^+)}{v(S)}. \)
Proof: By the definition of $L_0$, 
\[
P \left( v \left( \sigma^+_j \cap L_0 \right) \leq N^+_j \epsilon^+_k | C_k \right) 
= P \left( v \left( x_j : f(x_j) \leq 0, x_j \in \sigma^+_j \right) \leq N^+_j \epsilon^+_k | C_k \right), \forall k \tag{10} \]
If event $C_k$ is true, there exists $y^+_jk$, such that 
\[
P \left( v \left( x_j : f(x_j) \leq y^+_jk, x_j \in \sigma^+_j \right) \right) 
= \frac{v \left( x_j : f(x_j) \leq y^+_jk \right)}{v \left( \sigma^+_j \right)} = N^+_j \epsilon^+_k \tag{11} \]
Therefore, by applying equation (11) to equation (10), we have 
\[
P \left( v \left( \sigma^+_j \cap L_0 \right) \leq N^+_j \epsilon^+_k | C_k \right) 
= P \left( v \left( x_j : f(x_j) \leq 0 \right) \leq v \left( x_j : f(x_j) \leq y^+_jk | C_k \right) \right) \]
which is equal to $P \left( y^+_jk < 0 | C_{k-1} \right)$. Part-X decides the characterization of each subregion based on the minimum quantile. Let us define the event $F_{ijk}$ occurring when $v \left( \sigma^+_ijk \cap L_0 \right) \leq \epsilon^+_ijk$, such that:
\[
0 \leq \epsilon^+_ijk \leq \frac{\epsilon(v(\sigma^+_ijk))}{v(S)} \]
Consider as the PAC sampling measure, i.e., $P(\sigma^+_ijk)$ in Lemma 1, the probability $P(f \leq 0 | \sigma^+_ijk)$, where $\sigma^+_ijk$ is the set of sampled points in the $j^{th}$ subregion at level $j$, at iteration $k$. Such probability is implied by our separable kernel in equations (1)-(2), and it is therefore known at each iteration. Since each subregion has associated a different Gaussian process, we refer to this probability as $\rho_{ijk}$. Then the probability associated to the event $F_{ijk}$ is:
\[
P \left( F_{ijk} \right) \geq P \left( v \left( \sigma^+_ijk \cap L_0 \right) \leq \epsilon^+_ijk | C_k \right) \]
where $\sigma^+_ijk$ is the $j^{th}$ subregion at level $j$ at iteration $k$. Let us now refer to the mis-classification probability of this positively classified volume as $\epsilon^+_ijk$. Then, from Lemma 1, we know that:
\[
P \left( \epsilon^+_ijk \leq \frac{1}{\rho_{ijk}} + \frac{1}{\alpha_{ijk}} + 2 \ln n_j k + 1 \right) \geq 1 - \alpha_{ijk} \tag{12} \]
It is important to connect the classified volume and the error rate $\epsilon^+_ijk$. In particular, we have that:
\[
\epsilon^+_ijk = \left[ v \left( \sigma^+_ijk \cap L_0 \right) / v(S) \right] \tag{12} \]
And, conditional on the event $C_k$, the relationship (12) implies the following:
\[
\epsilon^+_ijk \leq \frac{\epsilon^+_ijk}{v(S)} \tag{12} \]
Given the definition of our event $F_{ijk}$, the following holds:
\[
P \left( v \left( \sigma^+_ijk \cap L_0 \right) \leq \epsilon^+_ijk | C_k \right) \geq 1 - \epsilon^+_ijk \]
\[
\geq 1 - \frac{\ln \left( \frac{1}{\rho_{ijk}} + \frac{1}{\alpha_{ijk}} + 2 \ln n_j k + 1 \right)}{n_j k} \tag{13} \]
Now, we want to derive the level-probability as the measure associated to the event $F_{jk} = \bigcap_i F_{ijk}$. Since the $N^+_jk$ subregions that get positively classified are independent, the following holds:
\[
P \left( F_{jk} \right) \geq \prod_i \left[ 1 - \frac{\ln \left( \frac{1}{\rho_{ijk}} + \frac{1}{\alpha_{ijk}} + 2 \ln n_j k + 1 \right)}{n_j k} \right] \tag{13} \]
Finally, we need to derive a bound for our original probability, i.e., $P \left( v \left( \sigma^+_ijk \cap L_0 \right) \leq N^+_jk \epsilon^+_jk | C_k \right)$, clearly the following holds:
\[
P \left( v \left( \sigma^+_ijk \cap L_0 \right) \leq N^+_jk \epsilon^+_jk | C_k \right) \geq P \left( F_{jk} | C_k \right) \tag{13} \]
Hence:
\[
P \left( v \left( \sigma^+_ijk \cap L_0 \right) \leq N^+_jk \epsilon^+_jk | C_k \right) \]
\[
\geq \prod_i \left[ 1 - \frac{\ln \left( \frac{1}{\rho_{ijk}} + \frac{1}{\alpha_{ijk}} + 2 \ln n_j k + 1 \right)}{n_j k} \right] \tag{13} \]
\[
= \prod_i \left[ 1 - \delta_{ijk} \right] \tag{13} \]
Now, we need to derive the iteration-level guarantee. We notice that, for different levels $1 \leq j \leq k$, the sampling and classification are independent processes. In fact, as the volume $v \left( \sigma^+_ijk \cap L_0 \right)$ is the union of the disjoint regions $v \left( \sigma^+_ijk \cap L_0 \right)$, so it is the volume $v \left( \sigma^+_k \cap L_0 \right)$ resulting from the union of the disjoint volumes $v \left( \sigma^+_ijk \cap L_0 \right), j \leq k$.
Let us define the event $A_{jk}$ to occur when $v \left( \sigma^+_ijk \cap L_0 \right) \geq N^+_jk \epsilon^+_jk | C_k$. Then, we are interested in the $P(\bigcap_i A_{jk}) = \prod_i P(A_{jk}) \geq \prod_i \left( 1 - \delta_{ijk} \right)$. Finally, we have our result:
\[
P \left( v \left( \sigma^+_k \cap L_0 \right) \leq N^+_k \epsilon^+_k \right) \geq \prod_j \prod_i \left( 1 - \delta_{ijk} \right), \forall k \tag{13} \]
This completes the proof.

Lemma 3 (Positive Reclassified Error Bound): Let $L_0$ denote the 0-level set defined in (6), and $\sigma^+_jk$ be the set of subregions re-classified from satisfying at the $j^{th}$ level of the PART-X tree, at the $k^{th}$ iteration of the algorithm. Refer to $\sigma^+_jk$ as the set of subregions re-classified from satisfying at the $k^{th}$ iteration of the algorithm. Assume the event $C_k$ in Definition 6 to be true. Then, the following holds:
\[
P \left( v \left( \sigma^+_ijk \cap L_0 \right) \leq N^+_jk \epsilon^+_jk \right) \geq \prod_{i=1}^{N^+_jk} \left( 1 - \delta_{ijk} \right), k > 1 \tag{13} \]
where, under event $C_k$, $0 \leq \epsilon^+_jk \leq \epsilon(v(\sigma^+_jk)), \text{ and } \delta_{ijk} = \frac{\ln \left( \frac{1}{\rho_{ijk}} + \frac{1}{\alpha_{ijk}} + 2 \ln n_j k + 1 \right)}{n_j k}$. Accounting for all levels yields the
following result:
\[
\begin{align*}
P \left( v \left( \hat{\sigma}^r_{jk} \cap S \setminus L_0 \right) \right) & \geq N^+_{jk} \epsilon^+_{jk-1} |C_k| \\
& \geq \prod_{j=1}^{k} \prod_{i=1}^{N^+_{jk}} [1 - \delta_{ijk}], \quad k > 1.
\end{align*}
\] (14)

where, under event \( C^+_k, 0 \leq \epsilon^+_j \leq \frac{\epsilon v(\hat{\sigma}^-_{jk})}{v(S)} \).

Proof: The proof follows the previous and it is therefore omitted here.

The same results can be proved for the negatively classified volume, leading to lemma 4-5, presented below.

**Lemma 4 (Negative Classified Error Bound):** Let \( L_0 \) denote the 0-level set defined in (6), and \( \hat{\sigma}^-_{jk} \) be the set of subregions classified as violating at the \( j^\text{th} \) level of the PART-X tree, at the \( k^\text{th} \) iteration of the algorithm. Refer to \( \hat{\sigma}^-_{jk} \) as the set of subregions classified as violating at the \( k^\text{th} \) iteration of the algorithm. Assume the event \( C^+_k \) in Definition 6 to be true. Then, the following holds:
\[
P \left( v \left( \hat{\sigma}^-_{jk} \cap (S \setminus L_0) \right) \right) \leq N^-_{jk} \epsilon^-_{jk} |C_k| \geq \prod_{i=1}^{k} [1 - \delta_{ijk}], \quad \epsilon^-_{jk} \leq \frac{\epsilon v(\hat{\sigma}^-_{jk})}{v(S)}.
\] (15)

where, under event \( C^-_k, 0 \leq \epsilon^-_j \leq \frac{\epsilon v(\hat{\sigma}^-_{jk})}{v(S)} \).

Proof: The proof is identical to the one of Lemma 3, and it is therefore omitted here.

**Lemma 5 (Negative Reclassified Error Bound):** Let \( L_0 \) denote the 0-level set defined in (6), and \( \hat{\sigma}^-_{jk} \) be the set of subregions re-classified from violating at the \( j^\text{th} \) level of the PART-X tree, at the \( k^\text{th} \) iteration of the algorithm. Refer to \( \hat{\sigma}^-_{jk} \) as the set of subregions re-classified from violating at the \( k^\text{th} \) iteration of the algorithm. Assume the event \( C^+_k \) in Definition 6 to be true. Then, the following holds:
\[
P \left( v \left( \hat{\sigma}^-_{jk} \cap (S \setminus L_0) \right) \right) \geq N^-_{jk} \epsilon^-_{jk} |C_{k-1}| \geq \prod_{i=1}^{N^-_{jk}} [1 - \delta_{ijk}], \quad k > 1.
\] (17)

where, under event \( C^-_k, 0 \leq \epsilon^-_j \leq \frac{\epsilon v(\hat{\sigma}^-_{jk})}{v(S)} \), and \( \delta_{ijk} = \frac{\ln \frac{1}{n_{jk}} + \ln \frac{1}{n_{jk} + 2 n_{jk} + 1}}{n_{jk}} \).

By construction:
\[
P \left( v \left( \hat{\sigma}^-_{jk} \cap (S \setminus L_0) \right) \right) \geq N^-_{jk} \epsilon^-_{jk} |C_k| \geq \prod_{j=1}^{k} \prod_{i=1}^{N^-_{jk}} [1 - \delta_{ijk}], \quad k > 1.
\] (18)

where, under event \( C^-_k, 0 \leq \epsilon^-_j \leq \frac{\epsilon v(\hat{\sigma}^-_{jk})}{v(S)} \).

Proof: The proof is identical to the one of Lemma 3, and it is therefore omitted here.

We are now ready to study the cumulated error of the algorithm at each iteration considering both classification and reclassification.

**Theorem 1:** Let \( L_0 \) and \( \epsilon \) be the 0-level set defined in (6), and the maximum tolerated error for PART-X, respectively. Assume the event \( C^+_k \) in Definition 6 to be true. Then, at each iteration \( k \) of the algorithm, the following bounds hold for the accumulated probability of misclassification:
\[
P \left( v \left( \hat{\sigma}^-_{jk} \cap L_0 \right) \right) \leq \sum_{j=1}^k \left( N^+_{j} \epsilon^+_{j} - N^-_{j} \epsilon^-_{j} \right) |C_k| \geq \prod_{j=1}^k \eta_j^+ \geq \prod_{j=1}^k \eta_j^-.
\] (19)

where,
\[
\eta_j^+ = \left( \prod_{h=1}^k (1 - \delta_{ijk}) \right) \left( \prod_{h=2}^k (1 - \delta_{ijk}) \right).
\] (20)

Proof: Let us first remind that for \( \hat{\sigma}^-_{jk}, \) and \( \hat{\sigma}^-_{jk}, \) the following holds:
\[
\hat{\sigma}^-_{jk} = \bigcup_{k=1}^k \left( \hat{\sigma}^-_{jk} \setminus \hat{\sigma}^-_{jk} \right), \quad \hat{\sigma}^-_{jk} = \bigcup_{k=1}^k \left( \hat{\sigma}^-_{jk} \setminus \hat{\sigma}^-_{jk} \right).
\]

Consequently, we study
\[
P \left( v \left( \bigcup_{k=1}^k \left( \hat{\sigma}^-_{jk} \setminus \hat{\sigma}^-_{jk} \right) \cap L_0 \right) \right) \leq \epsilon \left( \bigcap_{k=1}^k C_k \right).
\]

Let us define the event \( F_{jk} \), such that if \( F_{jk} \) is true, then \( v \left( \left( \hat{\sigma}^-_{jk} \setminus \hat{\sigma}^-_{jk} \right) \cap L_0 \right) \leq \epsilon_j, \) where \( \epsilon_j = \epsilon \frac{v(\hat{\sigma}^-_{jk})}{v(S)} \).

Here we are particularly interested in the event \( F^+_j \), which looks at the cumulative of error up to iteration \( k \). Hence, \( F^+_j \) holds if \( v \left( \left( \bigcup_{h=1}^k \hat{\sigma}^+_{jh} \right) \cap L_0 \right) \) and \( v \left( \left( \bigcup_{h=1}^k \hat{\sigma}^-_{jk} \right) \cap L_0 \right) \) are both less than \( \epsilon_j \).

From Lemma 2-3, we can derive a upper bound for \( v \left( \left( \bigcup_{h=1}^k \hat{\sigma}^+_{jh} \right) \cap L_0 \right) \) and a lower bound for \( v \left( \left( \bigcup_{h=1}^k \hat{\sigma}^-_{jk} \right) \cap L_0 \right) \).

For the reclassified volume, given an iteration \( h \), we know the following from Lemma 2-3:
\[
P \left( v \left( \hat{\sigma}^-_{jh} \cap L_0 \right) \right) \leq v \left( \hat{\sigma}^-_{jh} \setminus \hat{\sigma}^-_{jh} \right) \leq N^+_{jh} \epsilon^+_{jh} - N^-_{jh} \epsilon^-_{jh-1} |C_k| \geq \prod_{i=1}^{h} (1 - \delta_{ijk}) \prod_{i=1}^{h} (1 - \delta_{ijkl}), \quad h > 1.
\]
Then if we want to consider the iterations \( h = 1, \ldots, k \), we obtain the following, assuming the event \( \bigcap_{h=1}^{k} C_h \) holds:

\[
P \left( v \left( \bigcup_{h=1}^{k} \tilde{\sigma}_{jh}^+ \right) \cap L_0 \right) - v \left( \bigcup_{h=1}^{k} \tilde{\sigma}_{jh}^+ \right) \cap L_0 \right) 
\]

\[
\leq N_j^+ \epsilon_j^+ - N_j^+ \epsilon_{j,h-1} \bigcap \bigcup_{h=1}^{k} \bigcap_{i=1}^{l} \left( 1 - \delta_{ijk} \right) 
\]

\[
\geq \left( \prod_{h=1}^{k} N_j^+ \right) \prod_{i=1}^{l} \left( 1 - \delta_{ijk} \right) = \eta_j^+. 
\]

Then, considering the condition to hold at all levels, we have:

\[
P \left( v \left( \bigcap_{h=1}^{k} C_h \right) \right) \leq \sum_{j=1}^{k} \left( N_j^+ \epsilon_j^+ - N_j^+ \epsilon_{j-1} \right) \bigcap \bigcap_{h=1}^{k} \prod_{i=1}^{l} \left( 1 - \delta_{ijk} \right) \geq \prod_{j=1}^{k} \eta_j^+. 
\]

This concludes the proof. \( \square \)

**Theorem 2:** Let \( L_0 \) and \( \epsilon \) be the 0-level set defined in (6), and the maximum tolerated error for PART-X, respectively. Assume the event \( C_k \) in Definition 6 to be true. Then, at each iteration \( k \), the following bounds hold for the probability of mis-classification:

\[
P \left( v(\widetilde{\sigma}_k^- \cap (S \setminus L_0)) \right) 
\]

\[
\leq \sum_{j=1}^{k} \left( N_j^- \epsilon_j^- - N_j^- \epsilon_{j-1} \right) \bigcap \bigcap_{h=1}^{k} \prod_{i=1}^{l} \left( 1 - \delta_{ijk} \right) = \prod_{j=1}^{k} \eta_j^-. 
\]

**Proof:** The proof is identical to the one of Theorem 1, and it is therefore omitted here. \( \square \)

Note that \( \epsilon = \sum_{j=1}^{k} \left( N_j^+ \epsilon_j^- - N_j^+ \epsilon_{j-1} \right) + \sum_{j=1}^{k} \left( N_j^- \epsilon_j^- - N_j^- \epsilon_{j-1} \right) \) is the maximum overall error allowed over the input space, the study of the error will be shown in the analysis of the density associated to the event \( C_h \) in Lemma 6. Before the event analysis, let us provide two results that follow from the previous theorems.

**Corollary 1 (Per Iteration Positive Classification Error):** Let \( L_0 \) and \( \epsilon \) be the 0-level set defined in (6), and the maximum tolerated error for PART-X, respectively. Assume the event \( C_k \) in Definition 6 to be true. Then, at each iteration \( k \) of the algorithm, the following bounds hold for the probability of mis-classification of positive volume at iteration \( k \):

\[
P \left( v(\widetilde{\sigma}_k^+ \cap L_0) \right) 
\]

\[
- v(\widetilde{\sigma}_k^- \cap (S \setminus L_0)) \leq N_k^+ \epsilon_k^+ - N_k^- \epsilon_{k-1} \mid C_k 
\]

\[
\geq \prod_{j=1}^{k} \left( N_j^- \epsilon_j^- \right) \prod_{i=1}^{l} \left( 1 - \delta_{ijk} \right) = \gamma_k^+, k > 1. 
\]

Similarly, for the negatively classified volume, we have:

\[
P \left( \bigcap_{h=1}^{k} \left( C_h^+ \cap C_k^- \right) \right) \geq \prod_{h=1}^{k} \gamma_h^+ \prod_{h=1}^{k} \gamma_h^- 
\]

**Corollary 2 (Per Iteration Negatively Classification Error):** Let \( L_0 \) and \( \epsilon \) be the 0-level set defined in (6), and the maximum tolerated error for PART-X, respectively. Assume the event \( C_k \) in Definition 6 to be true. Then, at each iteration \( k \) of the algorithm, the following bounds hold for the probability of mis-classification of negative volume at iteration \( k \):

\[
P \left( v(\widetilde{\sigma}_k^- \cap (S \setminus L_0)) \right) 
\]

\[
- v(\widetilde{\sigma}_k^- \cap (S \setminus L_0)) \leq N_k^- \epsilon_k^- - N_k^- \epsilon_{k-1} \mid C_k 
\]

\[
\geq \prod_{j=1}^{k} \left( N_j^- \epsilon_j^- \right) \prod_{i=1}^{l} \left( 1 - \delta_{ijk} \right) = \gamma_k^-, k > 1. 
\]
Then, for \( k = j + 1 \), we have:

\[
P\left(\bigcap_{h=1}^{j+1} \left( C_{h}^+ \cap C_{h}^- \right) \right) = P\left( \left\{ C_{j+1}^+ \cap C_{j+1}^- \right\} \mid \bigcap_{h=1}^{j} \left( C_{h}^+ \cap C_{h}^- \right) \right) \\
\geq P\left( C_{j+1}^+ \right) \cdot P\left( \bigcap_{h=1}^{j} \left( C_{h}^+ \cap C_{h}^- \right) \right) \\
= P\left( \bigcap_{h=1}^{j} \left( C_{h}^+ \cap C_{h}^- \right) \right).
\]

Consider the Corollary 1-2 that bound the error for the classified volume at a specific iteration. In particular, since \( \hat{\Theta}_{j+1} = \hat{\Theta}_j \cup \hat{\Theta}_j^- \), \( \hat{\Theta}_{j+1}^- = \hat{\Theta}_j \cap \hat{\Theta}_j^- \), we have:

\[
(29) \geq P\left( \hat{\Delta}_{j+1}^+ \leq \Delta_{j+1}^+ \mid \bigcap_{h=1}^{j} \left( C_{h}^+ \cap C_{h}^- \right) \right) \\
\cdot P\left( \hat{\Delta}_{j+1}^- \leq \Delta_{j+1}^- \mid \bigcap_{h=1}^{j} \left( C_{h}^+ \cap C_{h}^- \right) \right) \\
\cdot P\left( \bigcap_{h=1}^{j} \left( C_{h}^+ \cap C_{h}^- \right) \right).
\]

where \( \hat{\Delta}_{j+1}^+ = v(\hat{\Theta}_j \cap \mathcal{L}_0) - v(\hat{\Theta}_j^+ \cap \mathcal{L}_0) \), and \( \hat{\Delta}_{j+1}^- = v(\hat{\Theta}_j^- \cap (S \setminus \mathcal{L}_0)) - v(\hat{\Theta}_j^- \cap (S \setminus \mathcal{L}_0)) \), \( \Delta_{j+1}^+ = N_j^+ \epsilon_j^+ - N_j^+ \epsilon_{j-1} \). Then, considering Corollary 1-2, and the assumption (28), we have:

\[
(30) \geq \gamma_{j+1}^+ \cdot \gamma_{j+1}^- \cdot \prod_{h=1}^{j} \gamma_{h+1}^+ \prod_{h=1}^{j} \gamma_{h+1}^-.
\]

Thus proving the claim.

**Theorem 3:** Let \( L_0 \) and \( \epsilon \) be the 0-level set defined in (6), and the maximum tolerated error for PART-X, respectively. Then, at each iteration \( k \) of the algorithm, the following bounds hold for the accumulated probability of mis-classification:

\[
P\left( v(\hat{\Theta}_k \cap \mathcal{L}_0) \leq \epsilon^+ \right) \geq \prod_{h=1}^{k} \gamma_{h+1}^+ \prod_{h=1}^{k} \gamma_{h+1}^-.
\]

**Proof:** The result follows from Lemma 6.

For the negatively classified volume:

**Theorem 4:** Let \( L_0 \) and \( \epsilon \) be the 0-level set defined in (6), and the maximum tolerated error for PART-X, respectively. Assume the event \( C_k \) in Definition 6 to be true. Then, at each iteration \( k \) of the algorithm, the following bounds hold for the accumulated probability of mis-classification:

\[
P\left( v(\hat{\Theta}_k^-) - v(\hat{\Theta}_k^- \cap \mathcal{L}_0) \leq \epsilon^- \right) \geq \prod_{h=1}^{k} \gamma_{h+1}^- \prod_{h=1}^{k} \gamma_{h+1}^-.
\]

**Proof:** The result follows from Lemma 6.

**V. Numerical Results**

We have implemented PART-X as a Python library\(^1\) that can be used as a stand-alone level set approximation toolbox, or as a module within our SBTG tool [49]. Section V-A, shows the performance of PART-X in terms of level set approximation accuracy over a set of nonlinear functions. In particular, we show the results for 2-dimensional cases for which we can plot the zero-level sets. Section V-B, instead shows the ability of the algorithm to identify bugs for a realistic engineering application [31]. Finally, Section V-C establishes that the performance of PART-X is comparable to other falsification tools, which do not provide probabilistic guarantees.

Among other outputs, PART-X returns an estimate of the falsification volume for a user specified quantile.

**Definition 7 (Falsification Volume):** Let \( \sigma_{i,j,k} \) be the sub-regions produced by the PART-X algorithm up to the \( k \)th iteration each with associated surrogate model \( \hat{Y}_{i,j,k}(x), x \in \sigma_{i,j,k} \), then the falsification volume calculated for \( \delta \)-quantile is:

\[
V_k^\delta \mathcal{C}(x,y) = \int_{\bigcup_{i,j,k} \sigma_{i,j,k}} l_{\mathcal{C} < 0}. \tag{33}
\]

**A. Non Linear Non Convex Optimization Examples**

First, we test PART-X for level set estimation with several non-linear non-convex functions: the Rosenbrock, Goldstein, and Himmelblau’s functions (see Figs. 6(a), 6(b), 6(c)), defined as:

- **Rosenbrock’s function** \((-1 \leq x_i \leq 1, i = 1, \ldots d)\):

  \[
f(x) = \sum_{i=1}^{d-1} \left( 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right) - 20
\]

  For \( d = 2 \) (2D), it has a symmetric level set in the region:

  \( S = [-1, 1] \times [-1, 1] \).

- **Goldstein-Price function** \((-1 \leq x, y \leq 1)\):

  \[
f(x, y) = (1 + (x + y + 1)^2 (19 - 14x + 3x^2 - 14y + 6xy + 3y^2))\left(30 + (2x - 3y)^2(18 - 32x + 12x^2 + 48y - 36xy + 27y^2)\right) - 50.
\]

  The Goldstein function has a smaller zero level set \( \mathcal{L}_0 \) volume compared to the Rosenbrock function.

- **Himmelblau’s function** \((-5 \leq x, y \leq 5)\):

  \[
f(x, y) = (x^2 + y^2 - 11)^2 + (x + y^2 - 7)^2 - 40.
\]

The level set \( \mathcal{L}_0 \) for the Himmelblau is disconnected.

PART-X was ran with initialization budget \( n_0 = 10 \), unclassified subregions, per-subregion budget \( n_{BO} = 10 \), classified subregions budget \( n_c = 100 \), maximum budget \( T = 5000 \), number of Monte Carlo iterations \( R = 10 \), number of evaluations per iterations \( M = 100 \), number of cuts \( B = 2 \), and classification percentile \( \delta_C = 0.05 \). Also, we used \( \delta_v = 0.001 \) to identify dimensions that should not be branched.

---

\(^1\)https://gitlab.com/bose1/part-x
Fig. 6. Sampling pattern for identifying the 0-level set (black solid lines).

![Graph](image)

**TABLE I**

| Algorithm | Mean | std err | Mean | std err | Mean | std err |
|-----------|------|---------|------|---------|------|---------|
| P-X ($\delta = 0.5$) | 2.686 | 7.628E-04 | 1.740 | 2.703E-03 | 34.313 | 1.830E-02 |
| P-X ($\delta = 0.95$) | 1.633 | 4.180E-08 | 0.030 | 2.278E-08 | 17.756 | 1.728E-05 |
| P-X ($\delta = 0.99$) | 1.635 | 4.452E-08 | 0.030 | 2.064E-08 | 17.803 | 1.747E-05 |
| Monte Carlo | 1.526 | 1.526 | 1.526 | 1.526 |

**TABLE II**

| Algorithm | Mean | std err | Mean | std err | Mean | std err |
|-----------|------|---------|------|---------|------|---------|
| P-X ($\delta = 0.5$) | 2.798 | 6.475E-04 | 0.700 | 2.198E-03 | 34.355 | 2.389E-02 |
| P-X ($\delta = 0.95$) | 1.628 | 6.494E-09 | 0.030 | 1.582E-09 | 17.652 | 4.549E-06 |
| P-X ($\delta = 0.99$) | 1.634 | 9.389E-09 | 0.030 | 1.598E-09 | 17.776 | 5.417E-06 |
| Monte Carlo | 1.526 | 1.526 | 1.526 | 1.526 |

Fig. 6(a) shows the partitioning obtained from P-X for Rosenbrock’s function from a randomly chosen macro-replication. The black solid lines identify the contour of the true level set $L_0$. The red subregions are classified by P-X as violating a region where the function has negative values, while the green subregions are classified as satisfying (positive values). It can be observed that the algorithm correctly identifies satisfying and violating regions. As the grid size decreases, we noticed that the performance of the algorithm remained relatively unaffected. This is a positive outcome, since it demonstrates the robustness of P-X with respect to the size of the grid adopted. Observing the pattern formed by the sampled locations, we can see – as expected – that samples concentrate around the border of the level set, while areas away from the border receive much less effort. Similar conclusions can be drawn from Figs. 6(b) and 6(c) for the other two functions.

Tables I-III show the results for the falsification volume $V_f$ obtained using: (1) the sum of the violating and remaining volumes; (2) the estimate obtained with the Gaussian process prediction using the $\delta_q$-quantile. Results are reported for each of the three test functions. The standard error for the metric was obtained as a result of 50 macro replications of the algorithm. The Monte Carlo estimation (Monte Carlo row in Tables I-III) is the point estimate for the falsification volume obtained with the same total budget used by the P-X algorithm accounting for macro-replications, i.e., $500 \times 30$ function evaluations. Given the large number of evaluations, we treat the result from the Monte Carlo sampling as an oracle for the true value of the 0-level set. Hence, the closer the P-X mean estimate, the better the performance. We can observe that using the Gaussian process as a means to estimate of the violating volume produces accurate estimates and results robust with respect to the selected $\delta_C$-quantile. This is due to the fact that we achieve particularly high density of sampling in violating regions so that the model variance is also very low within the level set.

Table II shows the results obtained with the finer grid generated with $R = 10, M = 500$ and Table III with $R = 20, M = 500$. Notice that the results are consistent across the three tables with one exception. In Table I, the P-X mean for the Goldstein function is 1.410 while in Tables II and III the mean is about 0.7. With the coarser grid due to the lower value of $M$, some of the partitions in the region $[-1, 0] \times [0.75, 1]$ are not classified yet; thus resulting in a larger estimate of the falsification volume. Hence, the choice of $R, M$ impacts the accuracy and uncertainty associated to the metrics used for classification. Hence, we would like both to be as high as computationally acceptable. While we do not propose an approach for the automatic setting of these parameters, the user should evaluate the highest values based on the available computing power and the per-iterate time that they target. In the remainder of the study, we use the configuration $(M = 500, R = 20)$. 

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
B. Level Set Identification for Cyber-Physical Systems (CPS)

To demonstrate the efficacy and relevance of the PART-X algorithm to engineering applications, we developed an add-on PART-X module for our open source SBTG Python package PSY-TaLiRo [49]. PSY-TaLiRo implements a similar architecture to S-TaLiRo [32] and Breach [37] Matlab tools. Currently, the system requirements can be expressed in a fragment of Timed Propositional Temporal Logic (TPTL) [70] which has polynomial time complexity and is strictly more expressive than Signal or Metric Temporal Logic (STL/MTL) [19]. In addition, our tool can use the Python packages RTAMT [29] and TLtk [30] that provide optimized algorithms for STL robustness monitoring.

Our SBTG tool requires a formal specification (in TPTL or STL), a black-box model of the SUT, and a finite and bounded set for the search parameters. When a vector \( x \) is generated by PART-X (or any other optimizer), the vector \( x \) is separated into a vector that represents static model parameters, e.g., initial conditions and/or other system parameters, and a vector that parameterizes signals. In the latter case, the (finite) vector that corresponds to the signals is interpolated by a user selected function (e.g., splines, piecewise linear, etc.) to produce an input signal for the SUT. Then, the SBTG executes the SUT using the provided parameters and signals, receives the SUT output trajectories, and computes the specification robustness. At that point, the robustness value is returned to the PART-X function, and the process repeats until the PART-X algorithm has reached one of its terminating conditions or the maximum allowed testing budget. Further details on requirements driven SBTG can be found in [1, 19], and [71].

In order to demonstrate the PART-X algorithm and our SBTG tool, we selected the F-16 benchmark (version 88ABW-2020-2188) [31] from the ARCH competition [48]. The F16 benchmark provides both a Matlab/Simulink and a Python version of a simplified F16 Ground Control Avoidance System (GCAS). The GCAS system uses 16 continuous variables and piece-wise non-linear differential equations to perform autonomous maneuvers to avoid hitting the ground. Our benchmark instance defines three static inputs \( \phi, \theta \) and \( \psi \), which are the initial roll, pitch and yaw angles of the aircraft, and there are no time-varying inputs. The three angles can range in the intervals \([0.2 \pi, 0.2833 \pi], [-0.5 \pi, -0.54 \pi], \) and \([0.25 \pi, 0.375 \pi] \), respectively. For this demonstration case study, we focused only on the altitude of the aircraft over the time horizon \([0, 15]\) after the GCAS has been activated. That is, the F-16 is initialized at an altitude and in a pose where a collision with the ground already is or will become possible and, hence, GCAS is activated to prevent the collision. The safety requirement in STL is

\[
\text{Always}_{0,15}(\text{altitude} > 0)
\]

which simply states that the altitude should always be above 0 during the first 15 sec.

PART-X was applied to the F16 GCAS model with the initial altitude set to various altitude values between 2300 ft. and 2400 ft., which are listed in Table IV. These experiments were run on a server with an Intel(R) Xeon(R) Platinum 8260 @ 2.40GHz CPU with 128G RAM running Ubuntu 20.04.2 LTS. The F16 GCAS model is known to be falsifiable at altitudes 2300 ft and it appears to be non-falsifiable at altitude 2400 ft. For altitude 2400 ft, we say “appears” since formal verification tools cannot yet provide deterministic safety guarantees for a system like the F-16 GCAS. However, we executed 250,000 simulations using uniform random sampling and confirmed that with high-confidence no simulation violated the safety requirement at 2400 ft. This conclusion is also confirmed by PART-X (see row for altitude 2400 ft in Table IV).

PART-X was run with initialization budget \( n_0 = 30 \), unclassified subregions, per-subregion budget \( n_{B0} = 10 \), classified subregions sampling budget \( n_c = 100 \), maximum budget \( T = 5000 \), number of Monte Carlo iterations \( R = 20 \), number of evaluations per iterations \( M = 500 \), number of cuts \( B = 2 \), and classification level \( \delta_C = 0.05 \). The minimum volume condition was set to \( \delta_0 = 0.001 \). Table IV shows the results of running PART-X on the F-16 benchmark. Each benchmark instance was run 50 times since PART-X is a stochastic algorithm. Here, \( FR \) represents the falsification rate, i.e., out of 50 macro-replications, how many times PART-X was able to sample at least one falsification. Finally, \( \min_{p_0} \) is the minimum robustness value sampled across all the evaluations by the algorithm. The estimation of the \( q \)-quantile falsification volume is reported under \( V_f^q \).

One immediate observation from Table IV is that the quantile falsification volume for altitude \( H_0 = 2300 \) ft is conclusively indicating the presence of falsifying regions (non-zero volume and LCB is positive). The falsification regions can be identified in the partitions returned by PART-X. This information could be used to focus any system debugging efforts to these regions of the operating space where we have evidence that falsifying behaviors exist. More information can be extracted from these regions about the system behavior by using tools such as [72], and model debugging could be performed with frameworks such as [73]. Alternatively, and if possible, the system operating guidelines could be updated to reflect that the system should not operate in certain regions. Along the same lines, the assumptions of assurance cases could be updated to reflect the safe operating regions.

The strength of PART-X becomes apparent for altitudes \( H_0 = 2338.4 \) ft, or higher. First, at altitude \( H_0 = 2400 \), PART-X conclusively determines that there are no falsifications. At altitude \( H_0 = 2338.4 \), it can be observed that for the same overall test budget PART-X identified falsifications when the Monte-Carlo (uniform sampling) estimate failed to produce falsifications. Near the falsification boundary, i.e, for altitudes 2338.4 and 2338.5, it can be observed that falsification volume statistics are the same indicating that even if the system is safe, it is not robustly safe. At this point, it is up to the test engineer or the system evaluator to decide whether the volume (i.e., probability of falsification) can be tolerated for the target application. If the system evaluator determines that higher confidence is needed in the results, then the sampling process can be resumed. The benefit of our approach is that the partitions contain information about which areas are the most frequently falsified. This can help focus the debugging efforts on the most critical regions of the operating space.
likely to falsify and, hence, further sampling can be biased towards these partitions.

Figure 7 presents some results for the F-16 benchmark for a falsifying case at altitude 2338.4 ft. Notice in Fig. 7(a) that the sampling is denser in the regions where falsifications are more likely (low robustness values). Another immediate observation in 7(b) is that the majority of the induced partitions are deemed to be safe (green), whereas the partitions where falsifications may occur (but with low probability) are classified as undetermined (blue). Figs. 7(c) presents the predicted robustness values over points sampled uniformly at random over the search space.

### C. Falsification Performance

In order to evaluate the performance of PART-X as a falsification tool, we compared it against the state-of-the-art CPS
falsification tools. In particular, we selected the Automatic Transmission (AT) Simulink model benchmark [74], which has been used in the ARCH Falsification Competition [48] since 2019.

The AT benchmark is a Simulink model of an automatic transmission controller with hybrid dynamics (both continuous and discrete behavior). The model is deterministic, i.e., it does not contain components with stochastic behavior, e.g., sensor noise, but it considers the driver inputs as bounded disturbances to search over for requirement violations. There are two driver inputs to the system: the throttle and brake. The brake input also enables modeling variable load to the engine, e.g., going uphill or downhill. The physical component of the system has two continuous-time state variables which are also its outputs: the speed of the engine $\omega$ (RPM) and the speed of the vehicle $v$ (mph). The discrete behavior of the model is due to several non-smooth components, e.g., lookup tables and signal saturation, but the main discrete behavior is enabled by 2 state machines modeling the current gear and the gear shifting logic. When the discrete system behavior needs to be analyzed, the model also can return the current gear and shifting logic state.

Initially, the vehicle is at rest at time 0, i.e., the model always initializes with velocity $v = 0$ and engine rotation $\omega = 0$. Hence, the system outputs depend only on the input signals. The throttle takes values between 0 (fully closed) to 100 (fully open), while the brake ranges in the interval $[0, 352]$. For all the benchmark problems reported in Table V, the input signals are parameterized using the $\phi$chip interpolation function with 7 control points for the throttle and 3 control points for the brake. Hence, the resulting search space is ten dimensional.

The results are shown in Table V, where we compare the performance of the falsification tools from ARCH falsification competition 2021 with PART-X on the AT benchmark [48]. It can be observed that the PART-X falsification performance is in par with the performance of specialized falsification tools with the added benefit that PART-X returns results with probabilistic guarantees. In other words, from the practitioners perspective, it may be preferable to use PART-X even when probabilistic guarantees are not needed in the initial system design iterations. The added benefit of PART-X is that it identifies the regions in the search space which are more likely to contain falsifying behaviors. It is worth mentioning that since no other falsification tool provides probabilistic guarantees, the ARCH falsification competition does not contain benchmark instances which are not falsifiable.

VI. CONCLUSION

We present, for the first time, the PART-X algorithm for the automatic generation of test cases with the objective of estimating the zero level set of the robustness landscape of a Cyber-Physical System (CPS) induced by its safety requirements. Specifically, given an input space, a simulation tool (or test harness), a requirement, and a significance level, PART-X attempts to classify the input space into violating, satisfying, and undecided subregions. The algorithm relies on a collection of Gaussian processes, each defined over a single subregion, used to estimate the unknown robustness function in unsampled areas of the input. We provide the falsification volume to be interpreted as the probability that a falsifying input exists. In our main results, we show how to probabilistically bound the classification error at each iteration of the algorithm; thus, providing a bound on the error of the estimation of the falsification volume. The numerical results demonstrate the ability of PART-X to estimate zero level sets when applied to general nonlinear non-convex functions as well as when applied to the F16 and AT benchmarks. The next step in our research is to extend our results from deterministic to stochastic systems as well as to explore scalability to higher dimensions.

REFERENCES

[1] J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito, and K. Butts, “Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques,” IEEE Control Syst. Mag., vol. 36, no. 6, pp. 45–64, Dec. 2016.

[2] T. Nahhal and T. Dang, “Test coverage for continuous and hybrid systems,” in Proc. CAV, vol. 4590. Cham, Switzerland: Springer, 2007, pp. 449–462.

[3] T. Dreossi et al., “Efficient guiding strategies for testing of temporal properties of hybrid systems,” in Proc. 7th Int. Symp. NASA Formal Methods (NFM), vol. 9058. Cham, Switzerland: Springer, 2015, pp. 127–142.

[4] Z. Zhang, G. Ernst, S. Sedwards, P. Arcaini, and I. Hasuo, “Two-layered falsification of hybrid systems guided by Monte Carlo tree search,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 37, no. 11, pp. 2894–2905, Nov. 2018.

[5] E. Plaku, L. E. Kavraki, and M. Y. Vardi, “Falsification of LTL safety properties in hybrid systems,” in Tools and Algorithms for the Construction and Analysis of Systems (TACAS), vol. 5505. Cham, Switzerland: Springer, 2009, pp. 368–382.

[6] H. Abbas, G. Fainekos, S. Sankaranarayanan, F. Ivančić, and A. Gupta, “Probabilistic temporal logic falsification of cyber-physical systems,” ACM Trans. Embedded Comput. Syst., vol. 12, no. 2s, pp. 1–30, May 2013.

[7] J. Deshmukh, M. Horvat, X. Jin, R. Majumdar, and V. S. Prabh, “Testing cyber-physical systems through Bayesian optimization,” ACM Trans. Embedded Comput. Syst., vol. 16, no. 5s, pp. 1–18, Oct. 2017.
[47] L. Mathesen, S. Yaghoubi, G. Pedrielli, and G. Fainekos, “Falsification of cyber-physical systems with multiple conjunctive temporal logic specifi-
cations using gradient based local search in space and time,” IFAC-
IFIP Control System Testing and Verification (CSTV), vol. 1, no. 1, pp. 439–446.
[48] G. Ernst et al., “Arch-comp 2021 category report: Falsification of
hybrid systems,” in Proc. 36th ACM Int. Conf. Hybrid Systems: Com-
put. Control. New York, NY, USA: Association for Computing Machinery, Apr. 2021, pp. 730–7313.
[49] G. Ernst et al., “Falsification of cyber-physical systems with Gaussian process regression,” in Proc. 15th ACM Int. Conf. Hybrid Systems: Com-
pus. Control. New York, NY, USA: Association for Computing Machinery, Apr. 2012, pp. 125–134, doi:10.1145/2185632.2185653.
[50] T. Akazaki, “Falsification of conditional safety properties for cyber-
 physical systems with Gaussian process regression,” in Runtime Veri-
fication, Y. Falcone and C. Sánchez, Eds. Cham, Switzerland: Springer, 2016, pp. 439–446.
[51] S. Silvetti, A. Policriti, and L. Borrioluschi, “An active learning approach to the falsification of black box cyber-physical systems,” in Integrated Formal Methods (Lecture Notes in Computer Science), vol. 10510, N. Polikarpova and S. A. Schneider, Eds. Turin, Italy: Springer, 2017, pp. 3–17, doi:10.1007/978-3-319-66845-1_1.
[52] R. Feldt and S. Pouloud, “Broadening the search in search-based software testing: It need not be evolutionary,” in Proc. 16th ACM Int. Workshop Search-Based Softw. Test. (SBST), May 2015, pp. 1–7.
[53] Z. Zhang and P. Arcaini, “Gaussian process-based confidence estimation for hybrid system falsification,” in Formal Methods, M. Huisman, C. Păsăreanu, and N. Zhan, Eds. Cham, Switzerland: Springer, 2021, pp. 330–348.
[54] P. I. Frazier, “Bayesian optimization,” in Recent Advances in Opti-
mization and Modeling of Contemporary Problems. INFORMS, 2018, pp. 255–278.
[55] S. Ghosh, F. Berkenkamp, G. Ranade, S. Qadeer, and A. Kapoor, “Verifying controllers against adversarial examples with Bayesian opti-
mization,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2018, pp. 1–7, doi:10.1109/ICRA.2018.8461904.
[56] L. Mathesen, G. Pedrielli, S. H. Ng, and Z. Bzilbinsky, “Stochastic optimization with adaptive restart: A framework for integrated local and global learning,” J. Global Optim., vol. 79, no. 1, pp. 87–110, Jan. 2021.
[57] L. Mathesen, S. Yaghoubi, G. Pedrielli, and G. Fainekos, “Falsification of cyber-physical systems with robustness uncertainty quantification through stochastic optimization with adaptive restart,” in Proc. IEEE 15th Int. Conf. Autom. Sci. Eng. (CASE), Aug. 2019, pp. 991–997.
[58] L. Mathesen, G. Pedrielli, and G. Fainekos, “Efficient optimization-based falsification of cyber-physical systems with multiple conjunctive requirements,” in Proc. IEEE 17th Int. Conf. Autom. Sci. Eng. (CASE), Aug. 2021, pp. 732–737.
[59] G. Ernst et al., “Arch-comp 2021 category report: Falsification with validation of results,” in Proc. 8th Int. Workshop Appl. Verification Continuous Hybrid Syst. (ARCH21), in EPiC Series in Computing, vol. 80, G. Frehse and M. Aichern, Eds. 2021, pp. 133–152 [Online]. Available: https://easychair.org/publications/paper/F4kF
A. Dokhanchi, B. Hoxha, and G. Fainekos, “On-line monitoring for cyber-physical systems,” in Proc. Formal Methods Industrial Critical Systems (FMICS), 2021, pp. 223–231.

G. Ernst, S. Sedward, Z. Zhang, and I. Hasuo, “Fast falsification of hybrid systems using probabilistically adaptive input,” in Quantitative Evaluation of Systems, D. Parker and V. Wolf, Eds. Cham, Switzerland: Springer, 2019, pp. 165–181.

Z. Zhang, D. Lyu, P. Arcaini, L. Ma, I. Hasuo, and J. Zhao, “Effective hybrid system falsification using Monte Carlo tree search guided by QBS-robustness,” in Proc. 33rd Int. Conf. (CAV), Berlin, Germany: Springer-Verlag, Jul. 2021, pp. 595–618, doi: 10.1007/978-3-030-81685-8_29.

S. Shekhar and T. Javidi, “Multiscale Gaussian process level set estimation,” in Proc. Mach. Learn. Res., K. Chaudhuri and M. Sugiyama, Eds. vol. 89, Apr. 2019, pp. 3283–3291.

X. Qin, Y. Xia, A. Zutshi, C. Fan, and J. V. Deshmukh, “Statistical verification of cyber-physical systems using surrogate models and conformal inference,” in Proc. ACM/IEEE 13th Int. Conf. Cyber-Phys. Syst. (ICCPP), May 2022, pp. 116–126.

T. Takisaka, Z. Zhang, P. Arcaini, and I. Hasuo, “A robustness-based confidence measure for hybrid system falsification,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 42, no. 5, pp. 1718–1731, May 2023.

H.-M. Kim, B. K. Mallick, and C. C. Holmes, “Analyzing nonstationary spatial data using piecewise Gaussian processes,” J. Amer. Statist. Assoc., vol. 113, no. 523, pp. 1094–1111, Jul. 2018, doi: 10.1080/01621459.2017.1307116.

Y. Prasetio, “Simulation-based optimization for complex stochastic systems,” Ph.D. thesis, Dept. Ind. Syst. Eng., Univ. Washington, Washington, DC, USA, 2011.

Z. B. Zabinsky and H. Huang, “A partition-based optimization approach for level set approximation: Probabilistic branch and bound,” in Women in Industrial and Systems Engineering: Key Advances and Perspectives on Emerging Topics, A. E. Smith, Ed. Berlin, Germany: Springer Nature, 2020, pp. 113–155.

H. Chipman, E. George, and R. McCulloch, “Bayesian treed models,” Mach. Learn., vol. 48, pp. 299–320, Jul. 2002.

D. G. T. Denison, N. M. Adams, C. C. Holmes, and D. J. Hand, “Bayesian partition modelling,” Comput. Statist. Data Anal., vol. 38, no. 4, pp. 475–485, Feb. 2002.

R. B. Gramacy and H. K. H. Lee, “Bayesian treed Gaussian process models with an application to computer modeling,” Taylor Francis, Tech. Rep., 2005, doi: 10.1080/01621459.2005.1050689.

R. D. Diwakaran, S. Sankaranarayanan, and A. Trivedi, “Analyzing neighborhoods of falsifying traces in cyber-physical systems,” in Proc. ACM/IEEE 8th Int. Conf. Cyber-Physical Syst. (ICCPs), Apr. 2017, pp. 109–120.

E. Bartocci, N. Manjunath, I. Mariani, C. Mateis, and D. Nicković, “CPSDebug: Automatic failure explanation in CPS models,” Int. J. Softw. Tools Technol. Transf., vol. 23, no. 5, pp. 783–796, Oct. 2021.

B. Hoxha, H. Abbas, and G. Fainekos, “Benchmarks for temporal logic requirements for autonomous systems,” in Proc. 1st 2nd Int. Workshop Appl. Verification Continuous Hybrid Syst., in EPiC Series in Computing, G. Frehse and M. Altthoff, Eds. vol. 34, 2015, pp. 25–30.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Hao Huang is an Assistant Professor with the Department of Industrial Engineering and Management, Yuan Ze University, Taoyuan, Taiwan. His research interests include simulation optimization, data analysis, and resource allocation applications.

Mauricio Castillo-Effen (Member, IEEE) received the B.Sc. degree in electrical engineering majoring in controls and automation from the University of Applied Sciences of Hannover, Germany, the M.Sc. degree in electrical engineering/controls from the University of San Simón (with TU Delft), and the Ph.D. degree in electrical engineering from the University of South Florida.

He is a fellow at the Lockheed Martin’s Advanced Technology Laboratories (LM ATL) leading the research area of trustworthy autonomy and artificial intelligence which focuses on the development and fielding of complex decision making in high criticality aerospace applications. Previously, he was a Lead Engineer at Robotics, Aviation, and V&V, General Electric’s Global Research Center. He also teaches robotics and controls as an Adjunct and Invited Faculty Member at multiple universities around the world (Rensselaer Polytechnic Institution, University of South Florida, and Bolivian Private University). He is an inventor of more than 20 published patents in the areas of aviation, robotics, and autonomy. His current interests include the application of formal techniques and systems engineering for the agile development and certification of aerospace systems incorporating complex software and artificial intelligence.

Dr. Castillo-Effen is an associate editor, a reviewer, and a committee member in multiple events organized by IEEE, AIAA, and AAAI.

Georgios Fainekos (Member, IEEE) received the Diploma degree (B.Sc. and M.Sc.) in mechanical engineering from the National Technical University of Athens (NTUA) in 2001, and the M.Sc. and Ph.D. degrees in computer and information science from the University of Pennsylvania (UPenn) in 2004 and 2008, respectively.

He is a Senior Principal Scientist at Toyota Motor North America Research and Development. Before joining Toyota, he was an Associate Professor at the School of Computing and Augmented Intelligence (SCAI), Arizona State University (ASU). After graduating from UPenn, he was a Post-Doctoral Researcher at NEC Laboratories America in the System Analysis and Verification Group. His technical expertise is in applied logic, formal verification, testing, control theory, artificial intelligence, and optimization. His research has applications to automotive systems, medical devices, autonomous (ground and aerial) vehicles, and human–robot interaction (HRI).

Dr. Fainekos received the NSF CAREER Award and the ASU SCIDSE Best Researcher Junior Faculty Award in 2013. His teaching activities were recognized with the 2021 and 2019 Top 5% Teaching Awards by the Fulton Schools of Engineering, ASU. His research has received several awards (e.g., IEEE ITSC 2019 and ACM HSCC RE 2019) and nominations (e.g., IEEE CASE 2021, IEEE/ACM MEMOCODE 2019, and IEEE ICRA 2007).