On Input Generators for Cyber-Physical Systems Falsification

Citation for the original published paper (version of record):
Ramezani, Z., Donzé, A., Fabian, M. et al (2024). On Input Generators for Cyber-Physical Systems Falsification. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 43(4): 1274-1287. http://dx.doi.org/10.1109/TCAD.2023.3333758

On Input Generators for Cyber-Physical Systems Falsification

Zahra Ramezani, Alexandre Donzé, Martin Fabian, and Knut Åkesson

***Abstract*—Falsification is a testing method that aims to increase confidence in the correctness of cyber-physical systems by guiding the search for counterexamples with some optimization algorithm. This method generates input signals for a simulation of the system under test and employs quantitative semantics, which serves as objective functions, to minimize the distance needed to falsify a specification. Various implementations based on different optimization strategies and semantics have been proposed and evaluated in the past. Generally, they assume that an input generator is given. However, this is often not the case in practice and different choices can lead to vastly different outcomes. Therefore, this paper introduces and evaluates various parameterizations of input generators, including pulse, sinusoidal, and piecewise signals with different interpolation techniques. These input generators are compared based on their performance on benchmark examples, as well as coverage measures in the space-time and frequency domains. Input generators facilitate the exploration of numerous different input signals within a single falsification problem, making them especially valuable for industrial practitioners seeking to incorporate falsification into their daily development work.**

***Index Terms*—Cyber-Physical Systems, Testing, Falsification, Simulation-based Optimization, Input Generators**

I. Introduction

Cyber-physical systems (CPSs) [1] are systems that integrate physical components with software to control them, and communication to connect to other systems in the environment. CPSs typically contain a mix of *continuous* and *discrete* dynamics, i.e., they are *hybrid systems*. These systems are often safety-critical, hence their correctness is essential. *Formal verification* and *testing* are commonly used methods for assessing the correctness of CPSs [2].

Since formal verification of hybrid systems is generally an undecidable problem [3], formal verification is seldom a viable option. Indeed, for numerous large-scale industrial systems, no formal model is available, that is, no model exists for which a current tool could produce a mathematical proof of correctness. Nonetheless, models are typically available for numerical simulation, so simulation-based testing (SBT), a form of software testing, is a prevalent approach for CPS verification and validation before integration and the construction of concrete prototypes of the system under test (SUT).

Simulation-based testing of CPS differs from traditional software testing due to the nature of the test cases involved. For traditional software, a test case comprises a combination of parameters for the SUT, whereas, for a CPS, a test case typically involves a combination of parameters and signals, i.e., continuous functions of time. Consequently, the search space is significantly more complex, and specific strategies are required for efficient testing.

Falsification of temporal logic specifications is a popular SBT approach that aims at producing counterexamples of properties expressed in signal temporal logic (STL), a formal language adapted to continuous-time signals.

Optimization-based falsification is performed using *quantitative semantics* that defines an objective function estimating the distance to falsifying a given specification. By searching for inputs that minimize this function, the falsification procedure is guided towards inputs that falsify the specification if those exist. If a falsifying input is found, we have a counterexample. Falsification tries to show the presence of counterexamples but cannot guarantee that counterexamples do not exist. An important challenge with falsification is to reduce the number of tests necessary to find the counterexamples. Three factors affect the efficiency of the falsification procedure: 1) the choice of quantitative semantics, 2) the optimization method used, 3) and how the inputs are generated. The first two factors were studied in previous works. In this paper, we focus on the third. Different quantitative semantics [4], [5] and optimization-based methods [6], [7] have been proposed in the literature. Evaluation results on benchmark examples in [6], [7] indicate that the choice of quantitative semantics is less significant than the optimization method. Optimization-based approaches struggle to handle high-dimensional problems; thus, the number of optimization variables should be low.

Parameterized input generators generate their input signals from a set of parameters. Typically, the parameters represent control points, and interpolation between these points may generate the signal. Input generators can also have other types of parameters, e.g. a sinusoidal generator can be represented with *period* and *amplitude*, but other types of input signals are also possible, including periodic pulses. Defining suitable parameters is a challenging problem where expert knowledge of the SUT is required since system dynamics, especially for large-scale industrial systems, are complex and often unknown. Choosing the number of control points, or the interpolation scheme, is often an arbitrary process, though this can be critical for the success or failure of the falsification. Limiting the number of control points decreases the dimensionality of the problem, making it potentially easier to solve. However, it also constrains inputs such that it may be impossible to generate an input signal able to falsify the specification. Therefore, finding the right balance between the flexibility of the input generation and the dimension of the optimization problem is essential.

In [8], the effect of input generators for falsification is

Z. Ramezani, M. Fabian, and K. Åkesson are with the Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden. E-mail: {rzahra, fabian, knut}@chalmers.se

A. Donzé is with Decyphir SAS, Moirans, France. E-mail:alex@decyphir.com

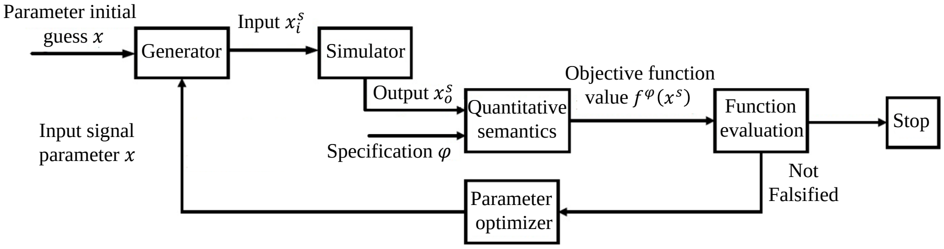

Fig. 1: A flowchart of optimization-based falsification. The input and output signals are denoted by x_i^s and x_o^s, respectively.

evaluated. A simple pulse generator is shown to be successful in falsifying many problems with only a few iterations. In this paper, we extend previous work by systematically evaluating different formulations of input generators.

A test coverage measure can be used as a guide in deciding when to terminate the falsification procedure. In [9], the focus is primarily on state coverage measures. State coverage measures determine what portion of a system's state space is covered by a test suite. State coverage measures can be used to guide test generation algorithms to sample cases from under-explored areas. A newly developed measure of continuous signal coverage was presented in [10], which focuses on the coverage of input signal spaces. Coverage tests of space and time and frequency domains have not been evaluated for falsification to the author's best knowledge.

In this paper, we make the following contributions:

- We introduce several input generators for the falsification of cyber-physical systems;
- We introduce and implement coverage measures for input signals in the space, time, and *frequency* domains;
- We evaluate and compare experimentally the coverage characteristics of our input generators;
- We provide a comprehensive evaluation of benchmark problems comparing the performance of the different input generators using different optimization strategies.

This paper is organized as follows: In Section II, we discuss related work in the domains of both traditional software testing and CPS testing including falsification. The general falsification problem is introduced in Section III. Section IV introduces input generators for falsification. Section V presents coverage measures in space, time, and frequency domains. Section VI evaluates the performance of the suggested methods on benchmark problems. Finally, the paper is summarized in Section VII.

II. Related Work

a) Software and CPS testing: CPS falsification is a form of software testing where test cases are signals and parameters. Although the topic of this paper is to study the effect of parameterization in falsification, i.e., going from a set of signals to a set of parameters, once a given parameterization is chosen, one can argue that the problem becomes quite similar to software testing. Traditionally, test suites for software testing and cyber-physical systems testing, in particular, are developed manually. Even with a comprehensive test suite, bugs can easily slip through unless the suite is very large. Moreover, by increasing the software size and complexity in the CPSs, automated testing as a complement to manual testing is needed. Random testing [11], model-based testing (MBT) [12], combinatorial testing (CT) [13], and search-based testing [14] are automated testing methods that can be used to generate test cases to validate a SUT automatically, which leads to reduce both effort and time costs. Random testing together with input generators is used in the tool QuickCheck [15], which has been applied to the testing of large software systems. QuickCheck was also extended to handle CPSs in [16]. A potential shortcoming of random test case generation is that counterexamples might be complicated; thus, a shrinking procedure was also introduced in [16] to simplify a counterexample after it has been found. Model-based tests aim to make the intended behavior explicit using behavior models. Software testing can be reduced in cost and increased in effectiveness with combinatorial testing (CT) [17]. Input interactions and sequence effects linked to user interaction can go undefined during development, and CT may be able to address them. In testing, coverage is an important notion that, in principle, should guide designers as to when "enough" testing was performed. It is also mostly a manual process. In [18], a set of industrial falsification benchmarks is introduced where a subset of the requirements to "falsify" are actually coverage requirements, i.e., if a test suite manages to falsify them, it means a specific set of behaviors of the SUT was necessarily explored by the tests. Those requirements were defined by the testers. Defining a more general notion of coverage for a CPS domain requires both a tractable way of computing the coverage and getting a meaningful interpretation out of it. This is difficult due to the dimensionality of the input signals and their continuous nature. Star-discrepancy such as in [9] is mathematically precise but hard to compute in general and interpret, which is why we introduced the simpler cell partitioning approach, which can be interpreted small projections. In that sense, our work is mostly inspired by combinatorial testing, which aims at producing test suites covering, e.g., all pairs of possible values, etc. Combining CT methods with our measures for producing test suites with good coverage properties is a subject for future work.

Coverage is not necessarily a reliable criterion for testing, though. One reason is that bugs, the falsified points, are typically not evenly distributed. Instead, they tend to cluster. Thus, even with good coverage, falsification performance might not be satisfactory, as is shown and discussed in this paper. For

CPSs, coverage-guided test criteria [9] can be used to assess the quality of a test suite. Various coverage criteria can be used to evaluate the extent to which a test suite has explored a system's behavior. These coverage-guided test criteria can aid in deciding when to continue generating more test cases. However, they generally do not help in determining when to cease generating test cases since satisfying a coverage test criterion does not guarantee the absence of counterexamples. According to [19], test suites that achieve the modified condition/decision coverage (MC/DC) over implementations are generally larger and more effective than test suites that achieve MC/DC over functionally equivalent but structurally simpler implementations. The MC/DC and fault-finding effectiveness of test suites generated over simpler implementations are significantly lower when applied to complex implementations. A simpler implementation still achieves high MC/DC, while a complex implementation still achieves high fault finding. In [20], however, it was shown that MC/DC was not always a meaningful measure for hybrid systems. Also in general, MC/DC is a white-box method, while the falsification process done in our paper is a black-box method.

b) Input Generation: Few research works focusing on the input generators are found in the literature. In [21], an input generator that uses words accepted by a timed automaton (TA) is presented, and it is shown how it can be used to develop complex periodic behaviors. This is more general than the pulse generator discussed in [8], but it also requires a lot more effort to model the TA for input generation. In [22], the proposed approach starts with constant signals and adds new control points incrementally. Their results on dynamic parametrization were promising but still preliminary and tested on a limited number of case studies Interestingly, they mention coverage as future work.

In [9], a method based on randomly exploring trees (RRT), adapted from path-planning problems, is presented. The problem of choosing the number of control points in the benchmark competition [23] was addressed by considering two input instances for each example. For the first instance, a fixed parameterization, given ranges and number of control points, is considered; for the second instance, "free" parameterization, the only constraint is that the input signal has to be piecewise continuous and within a given range.

In [7], different falsification methods were evaluated on benchmark problems. The steam condenser (SC) problem introduced in [24] was not falsified by any tool or method. However, it is possible to falsify the system, as shown in [24], by using an optimal control-based approach. The counterexample was a periodical input, switching between the extreme values of the allowed input parameter range. This counterexample was an inspiration to introduce the pulse input generators in [8].

One of the advantages of a pulse generator is that with a few parameters, it covers a wide range of typical testing signals: from bang-bang inputs, high frequency pulses up to steps, and constant signals when the period is larger than half the simulation time. This likely explains why the proposed pulse generator falsifies all the evaluated benchmark problems, including the SC problem. However, the proposed pulse generator in [8] left some unanswered questions. First, which parameters should be free and which should be fixed? Secondly, how to parameterize the pulse generator so it can conveniently generate relevant input signals?

III. Falsification

The falsification procedure for the optimization-based falsification approach is shown in Fig. 1. An n-dimensional vector, x, is used to parameterize the input signals, restricting each element to be within a defined range. Given the input parameters x, the *Generator*, i.e., the input generator, generates an input trace for each input signal, x_i^s, which describes a discrete sequence of input values $x_i^s[k]$. This paper focuses on this part where different input generators can be parameterized with a different number of control points and interpolation between them or completely different parameterizations like pulse and sinusoidal generator. Each element in the sequence is indexed by k, where k ranges from the start to the end of the simulation, and the full sequence is denoted by x_i^s. The *Simulator* simulates the SUT with x_i^s as input and generates x_o^s as output.

The combination of the input vector x_i^s and the output vector x_o^s, called x^s, together with the specification φ are used by the objective function $f^{\varphi}(x^s)$ to evaluate whether φ is falsified or not. If the specification is not falsified, the quantitative semantics gives a value representing how convincingly the test passed. On the other hand, if the specification is falsified, the current set of input and output traces is a counterexample, and the falsification procedure can terminate.

Metric Interval Temporal Logic (MITL) [25] and Signal Temporal Logic (STL) [26] can be used to express specifications with explicit timing intervals, which is well suited for CPS requirements. In this paper, STL is used. Two different quantitative semantics are defined for STL, *Max*, and *Additive*. Both of these can be expressed in terms of valued Booleans (VBools) [16]. In [6], [27], the strengths and weaknesses of different semantics have been compared from practical experiments that result in suggesting multiple quantitative semantics in [28]. Several toolboxes are available for simulation-based falsification of CPSs, like S-TaLiRo [29] and Breach [30]. These tools are both MATLAB/Simulink toolboxes that can be used to falsify large-scale industrial systems. S-TaLiRo finds counterexamples using specifications usually expressed in MITL, and Breach performs falsification using STL. This paper uses Breach together with *Additive* to evaluate the proposed input generators.

IV. Input Generators

In this section, we present a modified pulse generator that is parameterized differently and only uses the minimum and maximum values for each input signal, together with the simulation time. Hence, it helps test engineers in setting up falsification problems without requiring in-depth knowledge about the SUT. This paper aims to also evaluate the effectiveness of other input generations. Hence two groups of input generators will be assumed, sinusoidal generator and piecewise inputs.

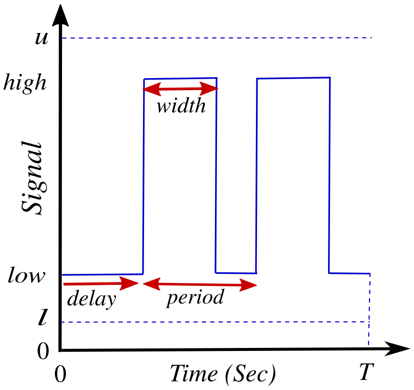

Fig. 2: A pulse input generator parameterization.

A. Input Generator Parameterization

Each input to the SUT is defined in a range $[l, u]$. In this work, three different input generators are considered, pulse, sinusoidal generator, and piecewise. In the following, we discuss the parameterization of these input signals.

1) Pulse Input Generator Parameterization: A periodic square wave with an initial *delay* is shown in Fig. 2. This signal is parameterized by $(\mathit{period}', \mathit{width}', \mathit{delay}', \mathit{high}', \mathit{low}')$. T is the simulation time. The pulse generator presented in Figure 2 is parametrized as follows: $\mathit{period} = \mathit{period}' \cdot T,$, $\mathit{width} = \mathit{width}' \cdot \mathit{period}$, $\mathit{delay} = \mathit{delay}' \cdot T$, $\mathit{low} = l + \mathit{low}' \cdot (u - l)$, $\mathit{high} = \mathit{low} + \mathit{high}' \cdot (u - \mathit{low})$ where The control points $\mathit{width}', \mathit{delay}', \mathit{low}', \mathit{high}'$ are all in the range $[0\ 1]$. period' can vary in two different ranges depending on the use of delay'. If period' and delay' are used together, $\mathit{period}' \in [0\ 1]$; otherwise $\mathit{period}' \in [0\ 2]$. This assumption is because if $\mathit{delay}' = 1$, it can make a constant signal equal to low. On the other hand, when the delay' is not included, $\mathit{period}' = 2$ will result in a step signal.

2) Sinusoidal Input Generator Parameterization: Five parameters $(\mathit{freq}', \mathit{decay}', \mathit{high}', \mathit{low}', \mathit{delay}')$ can be used to generate a sinusoidal generator signal, as shown in Fig. 3. The *period* refers to the time from one peak to the next. The right graph shows an exponentially decaying sine wave. In the implementation of sinusoidal generator, a frequency parameter (*freq*), is used such that $\mathit{period} = \frac{2\pi}{\mathit{freq}}$.

To generate the sinusoidal generator signal to the left in Fig. 3, we parameterize it as follows: $\mathit{freq} = \mathit{freq}'/(2 \cdot T_s)$, $\mathit{decay} = \mathit{decay}'/T$, $\mathit{delay} = \mathit{delay}' \cdot T$, $\mathit{low} = l + \mathit{low}' \cdot (u - l)$, $\mathit{high} = \mathit{low} + \mathit{high}' \cdot (u - \mathit{low})$, where T_s is the sampling time which is considered to be 0.01 s. The parameters $\mathit{freq}', \mathit{delay}', \mathit{low}', \mathit{high}'$ are in the range $[0\ 1]$, while decay' is in $[-1,\ 1]$.

3) Piecewise Input: A piecewise input signal has multiple segments, which can have different ranges. Two parameters can be defined for a piecewise input signal: control points and the interpolation scheme used between them. Fig. 4 shows three piecewise input signals where the intervals are $(0, t_1)$, (t_1, t_2) and (t_2, T). For this signal, five control points are needed, two control points in the time domain and three in the signal, i.e., the amplitude domain. Time intervals define how long the intervals should be. Different interpolations between the points are possible, for example, *previous*, *linear*, and *pchip*. *Previous* and *linear* interpolation are straightforward, but *pchip* uses piecewise cubic polynomials. If the values of each of the three sub-segments between $(0, t_1)$, (t_1, t_2) and (t_2, T) are the same, a *constant* input signal is generated, no matter the values of t_1 and t_2. On the other hand, if $t_1 = t_2 = 0$, a step input signal is generated. It should be noted here that in the literature, piecewise with *previous* interpolation is called a piecewise constant function. As a result, it is called *previous* in this paper to distinguish it from the *constant* input signal that has only one value in the signal domain.

4) Evaluating parameter combinations: In [31], an evaluation of the importance of different parameters for the efficiency of the falsification procedure using a pulse input generator is done. The evaluation uses benchmark problems [32], [28] with TuRBO as the optimization algorithm and Thompson Sampling [33] as an acquisition function (i.e., how exploration and exploitation of the search space are done in Bayesian optimization [34]). In the report, all benchmark examples were first evaluated with varying one input parameter and keeping the other parameters at their default values. This was followed by an evaluation of combinations of input parameters. We considered combinations of input signals where one parameter was successful in falsifying at least one specification, regardless of the success rate.

This results in Table I, which contains the most successful falsification combinations of different input parameters. To compare the performance of each input signal, we choose one, three, and five input parameters for the optimization. The reason for not evaluating all possible combinations of parameters is first due to time constraints. Secondly, for the piecewise input signal, we cannot have two or four control points. Because the number of control points must be an odd number. On the other hand, with one control point, a *constant* input signal is generated. In Section. VI, we evaluate the pulse signal with width' for one input parameter, $\mathit{low}', \mathit{width}', \mathit{period}'$ for three input parameters, and all parameters when having five input parameters. Similarly, the sinusoidal generator signal will be evaluated with freq' for one input parameter, and $\mathit{low}', \mathit{freq}', \mathit{decay}'$ when using three input parameters. They arrived at this setup after evaluation. Finally, the piecewise signal will have only one control point in the signal domain when only one parameter is allowed for the optimization. With three parameters, piecewise will have one variable in the time domain and two parameters that vary in the signal domain.

V. Coverage Measure

In this section, we introduce a method to measure how well a finite set of inputs covers the set of all possible inputs.

A. Space Coverage

We start by describing how we measure the coverage of a dense rectangular domain $\mathcal{D}$ by a finite, discrete set of parameter vectors. This is essentially done by binning the vectors into a set of cells (similar to binning [35]), and then counting the proportion of non-empty cells over the total

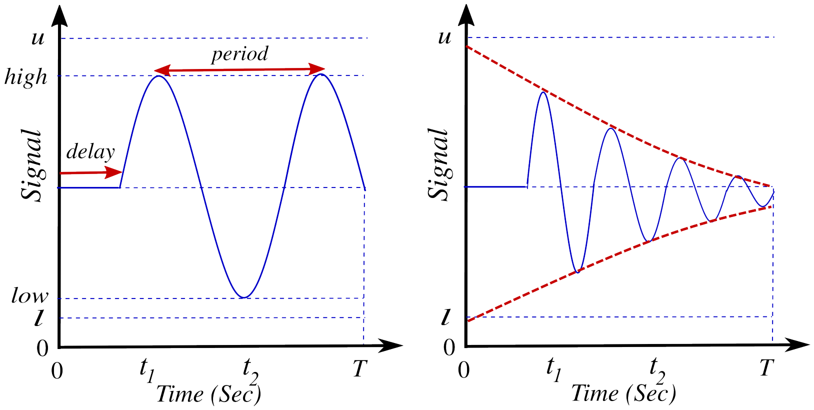

Fig. 3: Sinusoidal input parameterization: The left graph shows how a sinusoidal generator can be generated with $(freq', decay', high', low', delay')$. The right graph shows an exponentially decaying sine wave.

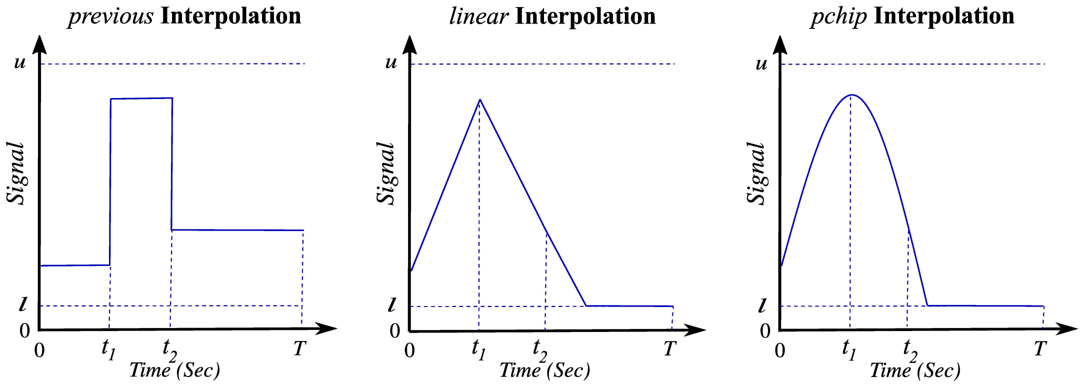

Fig. 4: Three piecewise input signals with five control points and with *previous*, *linear*, and *pchip* interpolation.

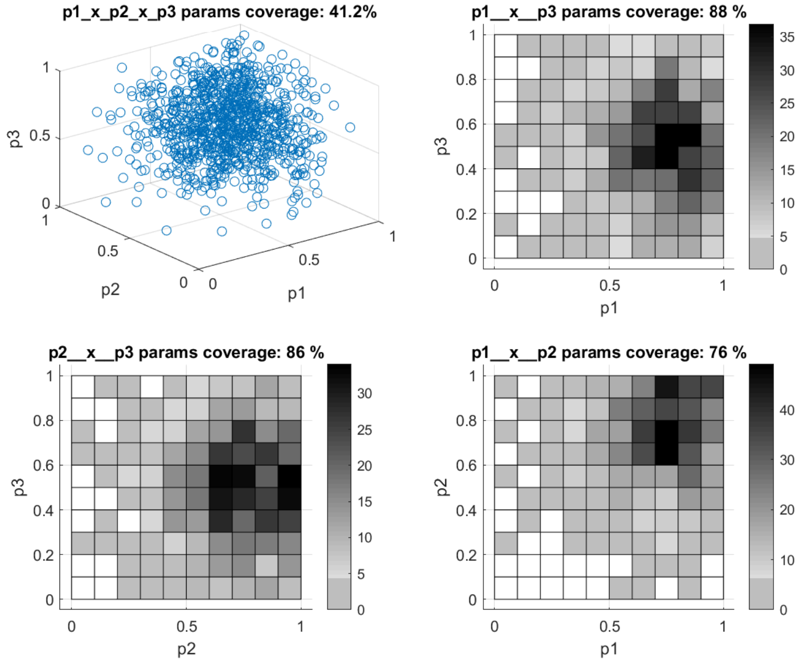

Fig. 5: Space coverage example with $\mathcal{D} = [0, 1)^3$, and $\mathcal{P}$ composed of 500 vectors with a normal distribution centered around $(0.75, 0.75, 0.5)$.

TABLE I: Most successful parameter combinations for falsification. The first value shows how many specifications, out of 40, are falsified in at least one run. The second value shows the number of successful falsifications in each run, where there is a maximum of 200 possible falsifications since we have 40 specifications, and each falsification run is repeated five times.

Num. of Inputs	One	Two	Three	Four	Five
Pulse Input combinations	$width'$	$low', width'$	$low', period', width'$	$low', period', width', high'$ $low', period', width', delay'$	All Parameters
Num. of falsified spec.	30;150	38;188	38;184	40;189	38;181

number of cells. More formally, we define a set of cells $\mathcal{C} = \{C_i, i = 1, 2, \cdots, n_c\}$ such that $\mathcal{C}$ is a partition of $\mathcal{D}$ as

$$\mathcal{D} = \bigcup_{i=1}^{n_c} C_i \text{ with } C_i \cap C_j = \emptyset \text{ if } i \neq j.$$

Given a finite set $\mathcal{P} = \{p_1, \cdots, p_N\}$ of vectors in $\mathcal{D}$, we define $\mathcal{C}_\mathcal{P} = \{C \in \mathcal{C}, \exists p \in \mathcal{P} \cap C\}$. Then

$$cov_\mathcal{D}(\mathcal{C}, \mathcal{P}) = \frac{card(\mathcal{C}_\mathcal{P})}{card(\mathcal{C})} = \frac{card(\mathcal{C}_\mathcal{P})}{n_c}, \qquad (1)$$

where $card$ returns the cardinality of a set, i.e., the number of its elements. This definition is quite general and flexible as it depends on how $\mathcal{C}$ is defined, and its computation and interpretation are mostly straightforward. It actually makes little sense in general to compute only one such coverage measure. We illustrate this concept with the example in Fig. 5, where $\mathcal{D} = [0, 1)^3$ and $\mathcal{S}$ is a composed of 500 vectors generated with a normal distribution centered around $(0.75, 0.75, 0.5)$. We report four different coverage measures: in the top left, C_i is of the form

$$[c_i^1, c_i^1 + 0.1) \times [c_i^2, c_i^2 + 0.1) \times [c_i^3, c_i^3 + 0.1)$$

where c_i^1 is in $\{0.1, 0.2, \cdots, 0.9\}$. In other words, each dimension of $\mathcal{D}$ is divided in 10 intervals, resulting in a partition into 1000 cells of size 0.1. The 500 samples result in a 3 D coverage of 41.2%. For the other plots, we consider projected measures on two dimensions. For instance, on the bottom right plot, C_i is of the form

$$[c_i^1, c_i^1 + 0.1) \times [c_i^2, c_i^2 + 0.1).$$

For this measure, the 500 samples result in a coverage of 88%. Note that the two-dimensional plots feature the number of samples in each cell.

B. Space and Time Coverage

In this work, we want to provide a coverage measure not only for samples but, more importantly, for signals as functions of time. Our proposition is a direct extension of the measure above by considering time as an additional dimension in which the signals are sampled. Consider a set of signals $\mathcal{S} = \{x_1, \cdots, x_N\}$, a domain $\mathcal{D}$ and a bounded time domain $\mathcal{T}$ such that $x_i(t) \in \mathcal{D}$ for all i. We define a finite set of cells $\mathcal{C}^\tau = \{C_i^\tau, i = 1, 2, \cdots, n_c\}$ such that

$$\mathcal{T} \times \mathcal{D} = \bigcup_{i=1}^{n_c} C_i^\tau \text{ with } C_i^\tau \cap C_j^\tau = \emptyset \text{ if } i \neq j.$$

We define now $\mathcal{C}_\mathcal{S} = \{C^\tau \in \mathcal{C}^\tau, \exists x \in S, t \in \mathcal{T}, x(t) \in C^\tau\}$. The coverage measure for $\mathcal{T}, \mathcal{D}, \mathcal{C}^\tau$ and $\mathcal{S}$ is then defined almost identically to (1) by

$$cov_{\mathcal{T}\times\mathcal{D}}(\mathcal{C}^\tau, \mathcal{S}) = \frac{card(\mathcal{C}_\mathcal{P})}{card(\mathcal{C}^\tau)}, \qquad (2)$$

Similarly to the above, the measures depend heavily on the choice of $\mathcal{C}^\tau$. This is again best explained with an illustrative example. In Fig. 6, we considered a two-dimensional input signal where each dimension is sinusoidal of different frequencies. The set $\mathcal{S} = \{\text{trace1}, \text{trace2}\}$ has only two samples (or traces) taking values in the $sig1$, $sig2$ dimensions. They are plotted in the top left plot. The top right figure shows coverage projected on time and $sig1$ dimensions, for which a 37 % coverage is achieved. The bottom left figure shows coverage projected on $time$ and $sig2$ dimensions. Because trace2 oscillates rapidly, it achieves 100 % coverage. Note that time is, of course, the main and important difference with the "space" coverage measure. However, our definition does not require it to be explicitly included in the cell decomposition. The bottom right plot of Fig. 5 shows the projected untimed coverage in dimensions $sig1$ and $sig2$.

C. Frequency Domain Coverage

Finally, we introduce a notion of frequency domain coverage for input signals. Assuming again a finite set $\mathcal{S}$ of $n-$dimensional signals and a bounded set of frequencies $\hat{\mathcal{D}} \subset \mathbb{R}^n$ for which we define a cell partitioning $\hat{\mathcal{C}} = \{\hat{C}_i, i = 1, 2, \cdots, n_c\}$ such that

$$\hat{\mathcal{D}} = \bigcup_{i=1}^{n_c} \hat{C}_i \text{ with } \hat{C}_i \cap \hat{C}_j = \emptyset \text{ if } i \neq j.$$

The general idea is to find which sets of frequencies are covered by the set of signals $\mathcal{S}$. For this, we use the Fourier transform [36], which for x in $\mathcal{S}$ we denote $\hat{x}$, and define $\hat{\mathcal{S}}$ as:

$$\hat{\mathcal{S}} = \{\nu \in \mathcal{D},\ \exists x \in \mathcal{S},\ |\hat{x}(\nu)| > \varepsilon(\hat{x}, \mathcal{S})\}$$

where $\varepsilon(\hat{x}, \mathcal{S}) > 0$ is designed as a threshold for the norm of $\hat{x}$ to be considered significant for coverage measurement purposes. From there, we can define the coverage in the frequency domain as:

$$\hat{cov}_\mathcal{D}(\hat{\mathcal{D}}, \mathcal{S}) = cov_\mathcal{D}(\hat{\mathcal{D}}, \hat{\mathcal{S}})$$

where $cov_\mathcal{D}(\hat{\mathcal{D}}, \hat{\mathcal{S}})$ is defined as in (1). We illustrate this definition on Fig. 7.

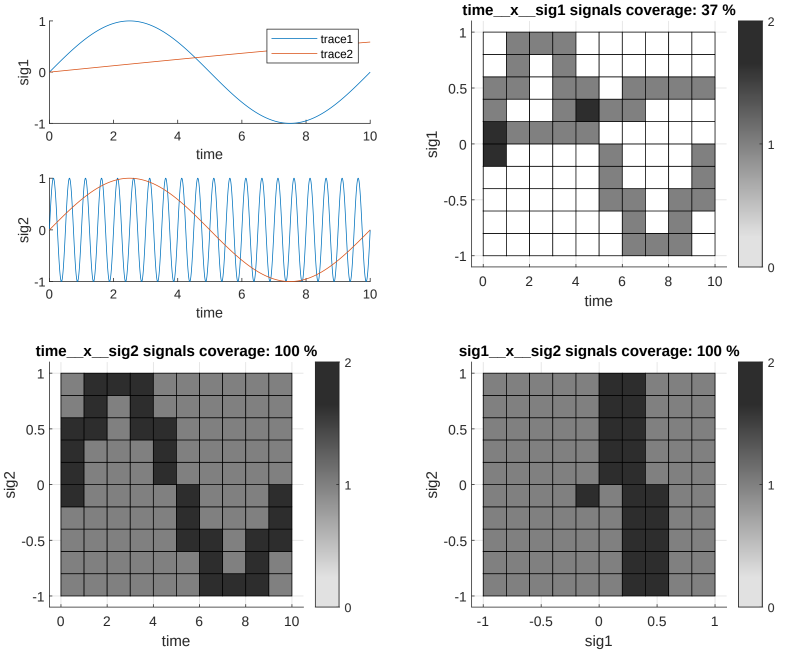

Fig. 6: Coverage of two traces of two dimensional signals, where each dimension is sinusoidal of different frequencies (top left). We show coverage projected on time and one dimension (top-left and bottom-right) as well as untimed two-dimensional coverage (bottom right).

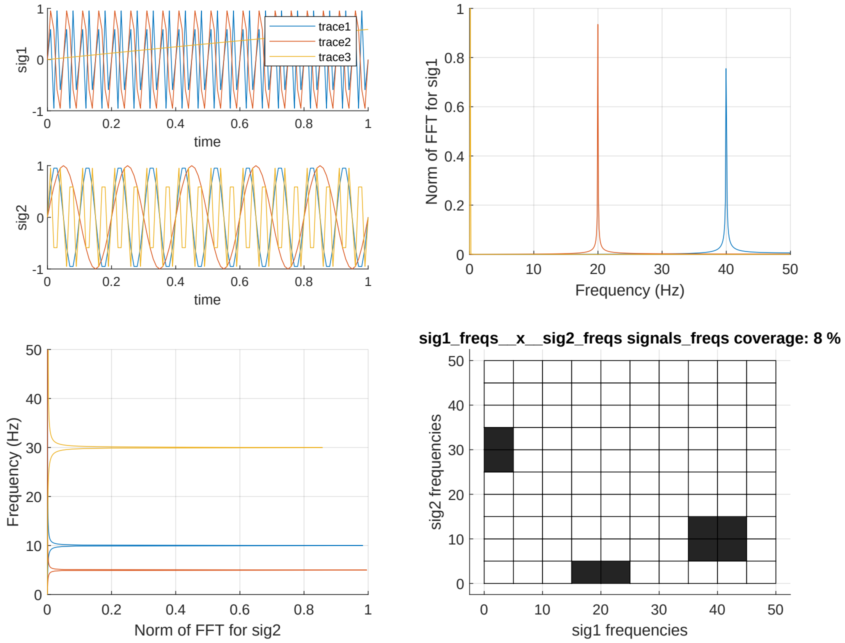

Fig. 7: Frequency coverage of three traces of a two-dimensional sinusoidal input signal. The Fourier transforms identify the fundamental frequencies, which results in small regions of the frequency domain being covered (bottom right).

D. Implementation and Experimental Evaluation

We implemented the coverage measures in the tool Breach[1] [30]. We briefly sketch the complexity of their computation. The approach avoids dimensional complexity using a sparse coding of the cells and a hash table. First, signals are discretized in time, and for each value, we check which cell it belongs to, which is done in linear time in dimension n, then check if the cell was already visited by another sample (done in constant time using a hash table). Computing the number of occupied cells over the total number of cells is then trivial and done in constant time. Assuming N signals of dimension n, discretized in time with the time step δ_t giving $k = T/\delta_t$ samples, the complexity is then in $\mathcal{O}(Nnk)$. Computing the frequency domain coverage measures requires additional Fast Fourier Transforms computations, which are done in $\mathcal{O}(k \log k)$[37], which gives a complexity of $\mathcal{O}(Nnk + Nnk \log k)$, so that the overall cost is actually dominated by FFT computations.

We evaluated the coverage measures presented in Section V on the different signal generators by sampling up to 1000 uniform random signals based on the parameterizations described above. The results are presented in Fig. 8.

The sinusoidal generator demonstrates the best performance in both time-space and frequency coverage. Piecewise generators have good performance for time-space coverage but cannot achieve more than 10% frequency domain coverage. Pulse generators provide average performance in both time-space and frequency domain coverage. Recall that when using only one variable, the pulse generator is allowed only two values (min and max), and the period is fixed, which explains the low performance for both time-space and frequency domain coverage.

E. Remarks and Discussion

In [8], it was found that the pulse signal generator performed remarkably well on falsification benchmarks. We speculated that by changing the parameters of the generator, the signal could efficiently represent different families of signals that are typical falsification candidates, such as steps, but also signals at variable frequencies. This motivated us to explore the coverage properties of different families of signals.

Our choice of coverage measure was motivated by simplicity and interpretability rather than more precise mathematical concepts. For instance, our measure heavily depends on the choice of the cells $\mathcal{C}$. By changing from 10 cells to 20 cells per dimension, the coverage measure of a given same set of signals can easily be halved or more. This may not be desirable from a mathematical point of view. However, assuming a careful and conscious choice of each dimension and discretization, with a simple default, normalized choice, the coverage number we obtain is always easily interpretable. Moreover, since projection on a few dimensions is easy, we believe that it makes it versatile to obtain specific insights.

We exploited this versatility to define frequency domain coverage. Basically, the Fourier transform that we used is one way of performing projection on a given feature before measuring coverage. The frequency domain analysis that we obtained is simple but enough to provide insights into our sets of signals. We tested our frequency domain coverage measure with the sinusoidal generator and could verify, as was expected, that this generator is optimal, i.e., reaches 100% coverage, as can be seen in Fig. 7 in the bottom graph. This motivated us to include this generator in our evaluation for falsification, which we do in the following section.

VI. Experimental Evaluation

In this section, we evaluate the input generators on the ARCH benchmark problems of [23] with three additional benchmarks from [28]. For the ARCH benchmark problems, the problems are denoted *AT*, *CC*, *NN*, *AFC*, and *SC*. The benchmarks from [28] are AT', the $\Delta - \Sigma$ modulator, and *SS*. Note that the AT' (Automatic Transmission) problem has different specifications and input ranges from the original *AT* problem formulated in [28].

A. Evaluated Optimization Methods

We compare the performance of the input generators using methods found in the literature: the hybrid corner-random (HCR) strategy and line-search falsification (LSF) [7], TuRBO [38], and πBO [39].

HCR, an optimization-free method, is included in the evaluation because in [7], it is shown to be surprisingly efficient in successfully falsifying many benchmark problems while being straightforward to implement.

A direct search method, line-search falsification (LSF), from [7] is also included because it showed to be an efficient method for falsification. This method combines random exploration with local search by randomly generating lines in the n-dimensional parameter space and optimizing over the line segments. In this work, we use Option 4 of this method which works with lines extending beyond the boundaries of input ranges and thus has a higher chance of resulting in corner values; see [7] for more details.

Two approaches based on Bayesian Optimization (BO) [40], TuRBO [38] with Lower Confidence Bound [41] as acquisition function and πBO [39], were in [42] shown to have unique capabilities for falsification.

B. Experimental Setup

The HCR method starts with corners and then switches to uniform random (UR) samples. The number of corners is fixed and depends on the dimensionality of the optimization problem, i.e., the number of input parameters n. When the number of corners, i.e., 2^n corner points, is exhausted, HCR continues using only random samples. It switches between corners and UR until the maximum number of simulations, 1000 here, is reached or a falsified point is found. For the LSF method, the value of maximum iterations to work with a single line is set to 3. As BO methods require a set of initial samples to start the process, we set the initial number of samples to $2 \cdot n$.

The results for all input generators, across different numbers of parameters and optimization methods, are presented in

[1] https://github.com/decyphir/breach

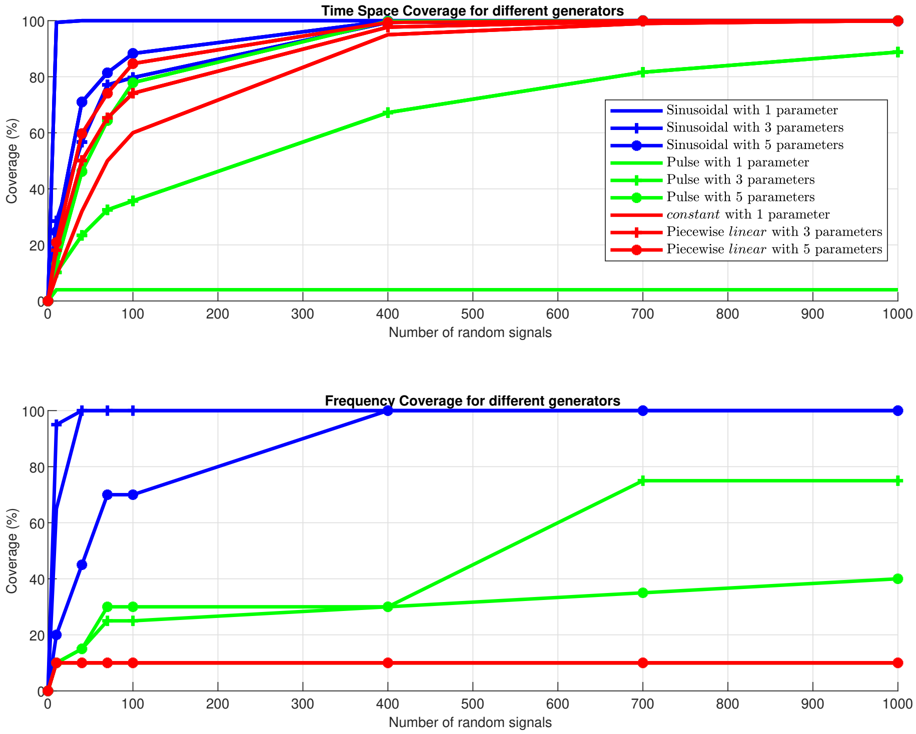

Fig. 8: Coverage performance measures for different signal generators. For piecewise generators, we only show the results for linear interpolation for clarity. For time space coverage, the result is similar for the three interpolation methods (slightly better for *pchip*, and slightly worse for *previous*) and identical for frequency coverage for all interpolation methods and the number of parameters.

TABLE II: Results for all evaluated input generators with different parameters and optimization methods. The first value shows how many specifications, out of 44, are falsified in at least one run. The second value shows the number of successful falsifications in each run, where there is a maximum of 220 possible falsifications since we have 44 specifications, and each falsification run is repeated five times. The last number is the average number of simulations (rounded) per successful falsification, where the maximum number of simulations is 1000.

Num. of Inp. Param.		One				Three				Five			
Inp. Gen.	Interpolation	HCR	LSF	TuRBO	πBO	HCR	LSF	TuRBO	πBO	HCR	LSF	TuRBO	πBO
Pulse	-	33;165;17	33;165;12	33;165;6	33;165;7	37;185;70	42;207;50	43;214;29	43;204;49	39;195;43	42;202;39	44;209;26	41;202;46
Sinusoid	-	27;135;178	31;140;34	29;125;62	28;139;24	37;185;74	40;193;33	36;175;54	40;194;20	37;185;15	40;195;27	40;196;16	39;195;19
Piece-Wise	*constant*	28;140;21	30;150;25	29;142;43	30;147;55	-	-	-	-	-	-	-	-
	previous	-	-	-	-	40;200;41	43;213;63	42;205;50	43;211;61	39;195;43	43;213;47	42;208;36	43;209;56
	linear	-	-	-	-	32;160;36	37;180;57	36;180;39	36;178;62	36;180;84	43;205;51	42;203;50	43;195;75
	pchip	-	-	-	-	32;160;34	35;173;32	35;170;30	36;170;55	38;190;82	43;209;59	42;210;49	43;204;59

Table. II. In this table, the first column denotes the input generator and the second column indicates the interpolation scheme used for the piecewise signal. The subsequent four columns display the evaluation results for a single input using HCR, LSF, TuRBO, and πBO, respectively. The following columns show the results for three and five input parameters. Each falsification is capped at a maximum of 1000 simulations. Each evaluation is repeated 5 times to account for the inherent randomness of the algorithms. The evaluation utilizes 44 different problem specifications. For each method, two values are presented: the first value represents how many specifications (out of 44) that have been falsified. Given that there are five falsification runs for each input parameter value and specification, a total of $5 \cdot 44 = 220$ falsifications are possible. The second value, provided after the semicolon, indicates the number of specifications falsified in each run out of the potential 220.

C. Results Analysis

From the result in Table II, we observe that most of the evaluated problems can be falsified with only one parameter for the optimization. In fact, a maximum of 33 out of 44 problems can be falsified using the pulse generator. The number of falsified specifications increases when the number of parameters for each input signal to 3 and 5. With a single input parameter, the pulse generator demonstrates superior performance compared to other input generators. It manages to falsify 33 specifications in each run amounting to a total of 165, irrespective of the optimization method used. When using sinusoidal generator and piecewise generator, the LSF method exhibits the best performance. By increasing the number of parameters for optimization to three, more specifications can be falsified, regardless of the choice of input generators. In this scenario, pulse generator demonstrates superior performance when paired with TuRBO, falsifying 43 specifications in all runs except one in 4 out of 5 trials. When comparing the performance of different interpolations of piecewise generator, *previous* outperforms others, especially when used with LSF, and three parameters are included. Increasing the number of parameters to five for each signal slightly enhances the falsification success rate. This improvement is particularly significant for *linear* and *pchip*. For instance, when using LSF with *linear* interpolation, 43 specifications are falsified with five parameters, compared to 37 with three parameters. Similarly, the number of falsified specifications jumps from 36 to 42 and from 36 to 43 when using TuRBO and πBO, respectively. In general, the best performance across all optimization methods is achieved when five parameters are included. This is evident when using TuRBO with the pulse generator signal results in falsifying all 44 specifications in at least one trial.

Although increasing the number of parameters for optimization generally helps to falsify more specifications, this is not universally true. For instance, when using pulse generator with three parameters, πBO falsifies 43 specifications with a total of 204 successful falsifications per run. On the other hand, πBO with five parameters only falsifies 41 specifications over 202 runs, a decrease from 43. This indicates that increasing the number of parameters can make some specifications more difficult to falsify, requiring more simulations. To show some specific cases, Table III lists the results for three specifications of benchmark problems, φ_1^{SC}, $\varphi_7^{AT'}$ and φ_4^{CC}, that are hard to falsify, as was discussed in [7], [8]. In this table, two values are presented for each specification; the first is the relative success rate of falsification in percent. Since there are 5 falsification runs for each parameter value and specification, the success rate will be a multiple of 20%. The second value, inside parentheses, is the average number of simulations (rounded) per successful falsification.

The problem φ_1^{SC} was the motivation to introduce the pulse generator in [8]. As shown in [8], φ_1^{SC} is falsified using pulse generator with only the *period* parameter, regardless of the used optimization methods. As can be seen in Table III, TuRBO is the best optimization method which falsifies φ_1^{SC} with both three and five parameters. Adding more parameters for optimization made the specification harder to falsify; for example, πBO falsifies it with three parameters in one run and not at all with five parameters.

The specification $\varphi_7^{AT'}$ is falsified only if the gear signal follows a given sequence with timing constraints. As can be seen, piecewise generator with different interpolations is more successful in falsifying this problem with all optimization-based methods. In general, increasing the number of parameters from 3 to 5, opposite to φ_1^{SC}, helps to falsify this example with less number of simulations for some optimization. For example, LSF's success rate is increased to 100% from 80% using *previous*. As can also be seen for this specification, how to interpolate among the control points is an important factor, which results in better performance of *linear* in general using three parameters.

If the number of parameters is increased, the performance of the optimization methods is reduced for φ_4^{CC}. For example, when using pulse generator with one parameter regardless of the optimization method, it is possible to falsify this specification. On the other hand, having three and five parameters makes this specification hard to falsify. Three parameters for piecewise generator demonstrate better performance than with only one parameter. Compared to five parameters, having three parameters using piecewise generator performs better.

From the above discussion, comparing the performance of input generators, sinusoidal did not perform well for these three specifications. On the other hand, pulse generator falsifies all three hard benchmark problems using TuRBO, as shown in Table II. Comparing the performance of the optimization methods, TuRBO outperforms the other optimization methods. On the other hand, πBO shows a reasonable performance for these three specifications, probably because of the moderate number of dimensions. As a result of the forgetting factor [39], if the wrong falsified area is injected into this method, πBO still converges to the falsified area. LSF shows worse performance compared to the BO methods for these three specifications, as was discussed in [42]. HCR does not show good performance here because these problems are hard to falsify, and an optimization-based method is beneficial.

The three evaluated specifications in Table III are belonging to the subsets of benchmark problems where the falsification efficiency is significantly changed when increasing the number of optimization parameters. This means that increasing the number of inputs for optimization might help to falsify a specification, or it might worsen the performance. Also, for the evaluated problems, the performance of falsification depends on which input signal is used.

Three cactus plots are shown in figures 9– 11 to compare different input generators and optimization methods for all 44 evaluated specifications with one, three, and five parameters, respectively. In these figures, each optimization method is shown with a specific marker: HCR (○), LSF (□), TuRBO (◇), πBO (⋆). Different colors are used to show different input parameters as pulse generator (green), sinusoidal generator (blue), and piecewise generator with *previous* (magenta), *linear* (red), and *pchip* (black). The constant signal is shown with red color in Fig. 9.

As depicted in Fig. 9, pulse generator outperforms other methods regardless of the optimization method used. Similarly, the pulse generator with three parameters when optimized with TuRBO falsifies more specifications using fewer simulations, as seen in Fig. 10. Moreover, employing all five parameters with pulse generator as the input generator and TuRBO as the optimization method enables the falsification of all evaluated benchmark problems in at least one of the falsification runs, see Fig. 11.

TABLE III: Evaluation results for all input generators and optimization methods for three hard specifications to falsify (φ_1^{SC}, $\varphi_7^{AT'}$, φ_4^{CC}). The first value is the relative success rate of falsification in percent. The second value, inside parentheses, is the average number of simulations (rounded) per successful falsification.

Spec.	Num. of Inp. Param.		One				Three				Five			
	Inp. Gen.	Interpolation	HCR	LSF	TuRBO	πBO	HCR	LSF	TuRBO	πBO	HCR	LSF	TuRBO	πBO
φ_1^{SC}	Pulse	-	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	100 (126)	20 (184)	0 (-)	0 (-)	60 (432)	0 (-)
	Sinusoid	-	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)
	Piecewise	*constant*	0 (-)	0 (-)	0 (-)	0 (-)	-	-	-	-	-	-	-	-
		previous	-	-	-	-	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)
		linear	-	-	-	-	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)
		pchip	-	-	-	-	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)
$\varphi_7^{AT'}$	Pulse	-	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	80 (418)	100 (183)	100 (375)
	Sinusoid	-	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	20 (633)	0 (-)
	Piecewise	*constant*	0 (-)	0 (-)	0 (-)	0 (-)	-	-	-	-	-	-	-	-
		previous	-	-	-	-	0 (-)	80 (538)	100 (131)	100 (242)	0 (-)	100 (462)	100 (172)	80 (338)
		linear	-	-	-	-	100 (251)	100 (226)	100 (212)	100 (219)	0 (-)	100 (454)	100 (103)	60 (390)
		pchip	-	-	-	-	100 (251)	100 (182)	100 (84)	60 (339)	100 (844)	100 (296)	100 (240)	100 (317)
φ_4^{CC}	Pulse	-	100 (393)	100 (129)	100 (47)	100 (134)	0 (-)	80 (125)	100 (216)	80 (261)	0 (-)	0 (-)	60 (457)	40 (515)
	Sinusoid	-	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)
	Piecewise	*constant*	0 (-)	0 (-)	0 (-)	0 (-)	-	-	-	-	-	-	-	-
		previous	-	-	-	-	0 (-)	80 (603)	0 (-)	40 (612)	0 (-)	60 (373)	0 (-)	40 (551)
		linear	-	-	-	-	100 (232)	100 (319)	100 (148)	100 (70)	0 (-)	100 (498)	100 (454)	60 (307)
		pchip	-	-	-	-	100 (232)	100 (229)	80 (188)	100 (75)	0 (-)	100 (472)	100 (308)	40 (103)

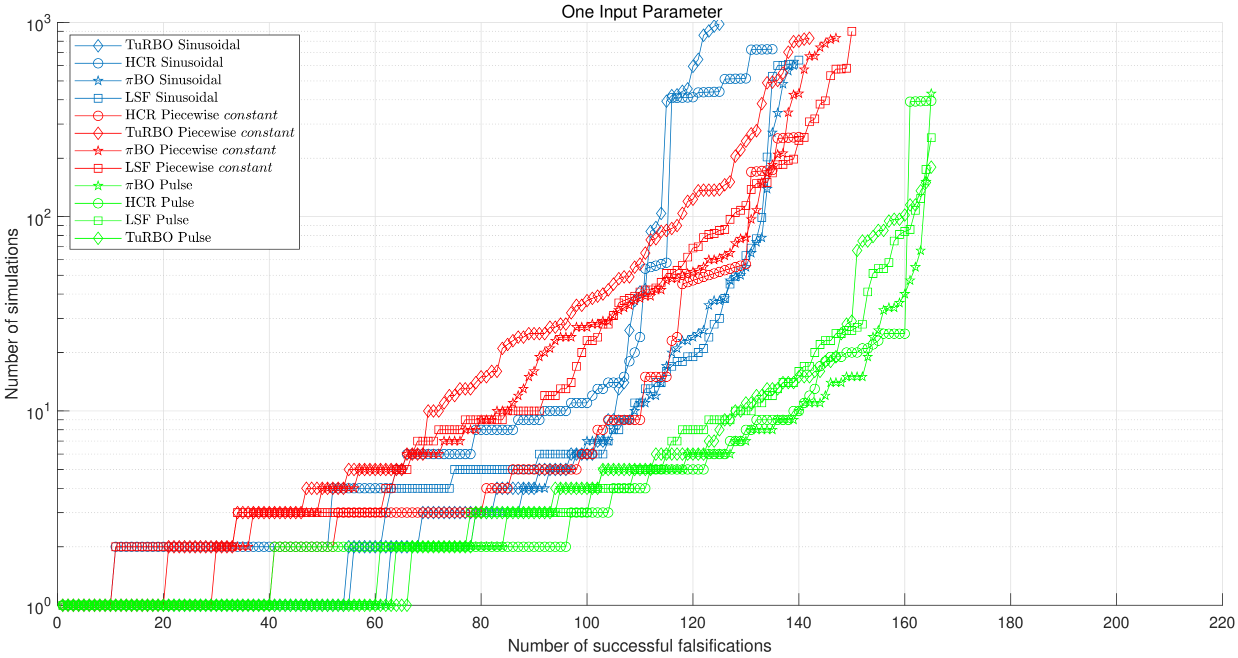

Fig. 9: A cactus plot showing the performance of different input generators using one input parameter and optimization methods on all benchmark problems. The x-axis gives the number of successful falsifications completed for the number of simulations (y-axis, logarithmic scale), a maximum of 1000 simulations.

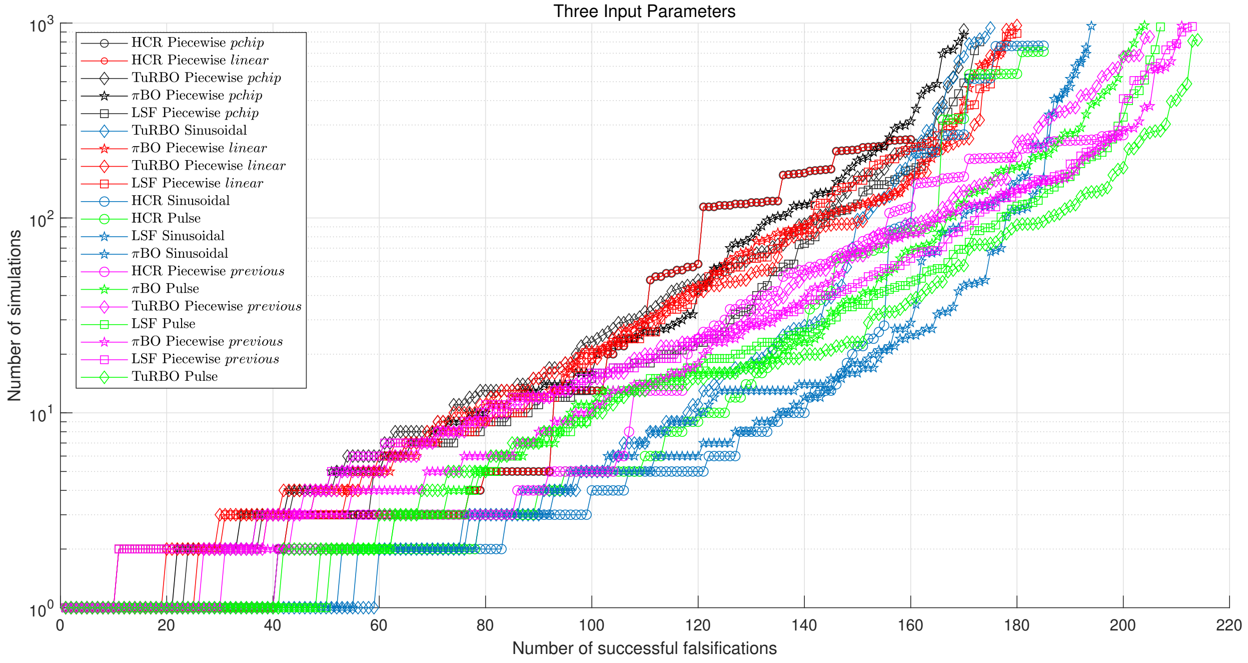

Fig. 10: A cactus plot showing the performance of different input generators using three input parameters and optimization methods on the benchmark problems. The x-axis gives the number of successful falsifications completed for the number of simulations (y-axis, logarithmic scale), a maximum of 1000 simulations.

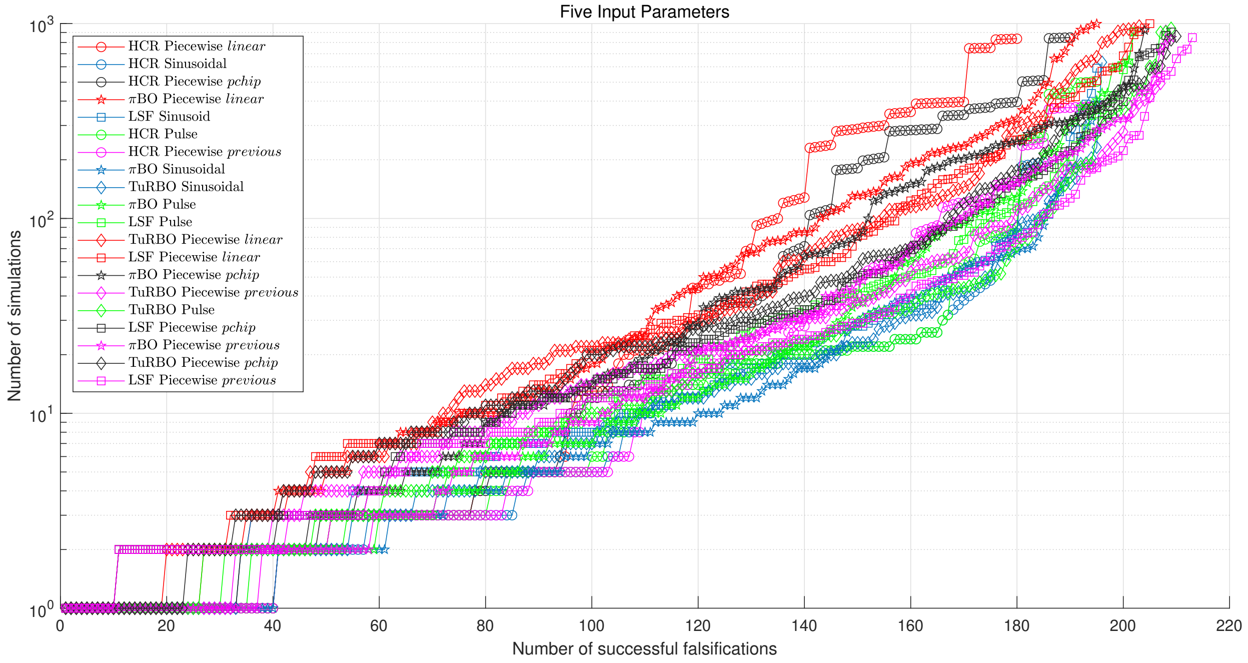

Fig. 11: A cactus plot showing the performance of different input generators using five input parameters and optimization methods on all benchmark problems. The x-axis gives the number of successful falsifications completed for the number of simulations (y-axis, logarithmic scale), a maximum of 1000 simulations.

VII. Conclusion

This paper explores the impact of various input generators on the falsification of cyber-physical systems. This topic is relevant to practitioners and has not been thoroughly explored within the falsification community. The assessed input signal generators encompass pulse, sinusoidal, and piecewise signals. These inputs are appropriate for black-box falsification, in which no assumptions are made about the SUT, apart from the ability to modify inputs, simulate the system, and observe outputs. The various input generators are assessed on benchmark problems, and coverage measures of the different input generators are presented in both the space-time and frequency domains. To evaluate the efficiency of the distinct input generators, we employ different search methods, including optimization-free and optimization-based approaches. Pulse generators exhibit average performance in space-time and frequency domain coverage. They were able to falsify all benchmark problems using TuRBO and having all five input parameters as decision variables. The sinusoidal generator excelled in covering space and frequency but demonstrated lower efficiency than other input generators for falsification. A probable reason is their overly regular nature, whereas typical falsification signals exhibit discontinuous behavior, such as steps and "bang-ban" patterns, which pulses effectively generate. This is further exemplified by the piecewise constant input generator (*previous*), which demonstrates strong performance for falsification and space-time coverage. However, its frequency domain coverage is comparatively weak, which hindered its success in some problems. The evaluation results indicate that numerous specifications are falsified using only a single input parameter. Increasing the number of parameters proves beneficial for falsifying certain specifications, while adversely affecting efficiency for others. Additionally, the evaluation reveals that employing a learning-based approach, specifically Bayesian optimization, reduces the number of simulations required for the successful falsification of more challenging problems. For future research, it would be compelling to examine the proposed input generators on industrial-scale problems with a large number of input signals, such as the benchmarks presented in [43]. Moreover, several directions warrant exploration concerning the coverage measures we introduced:

- By calculating coverage measures during falsification, we could potentially monitor the exploration versus exploitation behavior of the solver and influence it in either direction;
- Is it possible to "invert" the coverage measure of signals to design efficient test suites, that is, test suites with a minimal number of sets achieving a specified level of coverage? The concept would be to extend ideas from combinatorial testing to our framework [13];
- Investigate the theoretical and empirical connections with other coverage measures, such as star-discrepancy for samples [9], and more broadly, epsilon nets coverage for signals and functions [44].

Finally, our results on input generation and parameterization and the coverage measures we introduced can be incorporated in metaheuristics for falsification such as in [10], [21]. However, the design and evaluation of such automatic strategies should be done carefully, as another extensive experimental study in [45] has shown that adaptive strategies outperform simple but effective ones, such as those based on corners and random exploration (e.g., HCR in our work), only in relatively rare, difficult cases.

Acknowledgment

This work was supported by the Swedish Research Council (VR) project SyTeC VR 2016-06204 and by the Swedish Governmental Agency for Innovation Systems (VINNOVA) under project TESTRON 2015-04893. It was also partly funded by the Wallenberg AI, Autonomous Systems, and Software program (WASP), sponsored by the Knut and Alice Wallenberg Foundation. The evaluations were conducted utilizing resources at the High Performance Computing Center North (HPC2N) of Umeå University, a Swedish national center for scientific and parallel computing. The authors express their gratitude to K. Šehić, L. Nardi, K. Claessen, and N. Smallbone for their valuable input to this work.

References

[1] R. Alur, *Principles of cyber-physical systems*. MIT press, 2015.

[2] S. Mitra, *Verifying Cyber-Physical Systems: A Path to Safe Autonomy*. MIT Press, 2021.

[3] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" in *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, 1995, pp. 373–382.

[4] V. Raman, A. Donzé, D. Sadigh, R. M. Murray, and S. A. Seshia, "Reactive synthesis from signal temporal logic specifications," in *Proceedings of the 18th Int. Conf. on hybrid systems: Computation and control*. ACM, 2015, pp. 239–248.

[5] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262 – 4291, 2009.

[6] J. L. Eddeland, S. Miremadi, and K. Åkesson, "Evaluating optimization solvers and robust semantics for simulation-based falsification," in *ARCH20. 7th Int. Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)*, ser. EPiC Series in Computing, G. Frehse and M. Althoff, Eds., vol. 74. EasyChair, 2020, pp. 259–266.

[7] Z. Ramezani, K. Claessen, N. Smallbone, M. Fabian, and K. Åkesson, "Testing cyber–physical systems using a line-search falsification method," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 8, pp. 2393–2406, 2022.

[8] Z. Ramezani, A. Donzé, M. Fabian, and K. Åkesson, "Temporal logic falsification of cyber-physical systems using input pulse generators," *EPiC Series in Computing*, vol. 80, pp. 195–202, 2021.

[9] T. Dreossi, T. Dang, A. Donzé, J. Kapinski, X. Jin, and J. V. Deshmukh, "Efficient guiding strategies for testing of temporal properties of hybrid systems," in *NASA Formal Methods Symposium*. Springer, 2015, pp. 127–142.

[10] A. Adimoolam, T. Dang, A. Donzé, J. Kapinski, and X. Jin, "Classification and coverage-based falsification for embedded control systems," in *Int. Conf. on Computer Aided Verification*. Springer, 2017, pp. 483–503.

[11] R. Hamlet, "Random testing," *Encyclopedia of software Engineering*, vol. 2, pp. 971–978, 1994.

[12] A. Pretschner, "Model-based testing," in *Proceedings. 27th Int. Conf. on Software Engineering, 2005. ICSE 2005.* IEEE, 2005, pp. 722–723.

[13] D. R. Kuhn, R. N. Kacker, and Y. Lei, *Introduction to combinatorial testing*. CRC press, 2013.

[14] P. McMinn, "Search-based software testing: Past, present and future," in *2011 IEEE Fourth Int. Conf. on Software Testing, Verification and Validation Workshops*. IEEE, 2011, pp. 153–163.

[15] K. Claessen and J. Hughes, "QuickCheck: a lightweight tool for random testing of Haskell programs," in *Proceedings of the fifth ACM SIGPLAN Int. Conf. on Functional programming*, 2000, pp. 268–279.

[16] K. Claessen, N. Smallbone, J. Eddeland, Z. Ramezani, and K. Åkesson, "Using valued booleans to find simpler counterexamples in random testing of cyber-physical systems," *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 408–415, 2018.

[17] D. R. Kuhn, R. N. Kacker, Y. Lei *et al.*, "Practical combinatorial testing," *NIST special Publication*, vol. 800, no. 142, p. 142, 2010.

[18] J. L. Eddeland, S. Miremadi, and K. Åkesson, "Evaluating optimization solvers and robust semantics for simulation-based falsification," in *ARCH20. 7th Int. Workshop on Applied Verification of Continuous and Hybrid Systems*, 2020.

[19] G. Gay, M. Staats, M. Whalen, and M. P. E. Heimdahl, "The risks of coverage-directed test case generation," *IEEE Transactions on Software Engineering*, vol. 41, no. 8, pp. 803–819, 2015.

[20] J. Eddeland, J. G. Cepeda, R. Fransen, S. Miremadi, M. Fabian, and K. Åkesson, "Automated mode coverage analysis for cyber-physical systems using hybrid automata," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 9260–9265, 2017, 20th IFAC World Congress.

[21] B. Barbot, N. Basset, T. Dang, A. Donzé, J. Kapinski, and T. Yamaguchi, "Falsification of cyber-physical systems with constrained signal spaces," in *NASA Formal Methods Symposium*. Springer, 2020, pp. 420–439.

[22] J. Deshmukh, X. Jin, J. Kapinski, and O. Maler, "Stochastic local search for falsification of hybrid systems," in *Int. Symposium on Automated Technology for Verification and Analysis*. Springer, 2015, pp. 500–517.

[23] G. Ernst, P. Arcaini, I. Bennani, A. Chandratre, A. Donzé, G. Fainekos, G. Frehse, K. Gaaloul, J. Inoue, T. Khandait *et al.*, "ARCH-COMP 2021 category report: Falsification with validation of results." in *ARCH@ ADHS*, 2021, pp. 133–152.

[24] S. Yaghoubi and G. Fainekos, "Gray-box adversarial testing for control systems with machine learning components," in *Proceedings of the 22nd ACM Int. Conf. on Hybrid Systems: Computation and Control*, 2019, pp. 179–184.

[25] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-Time Systems*, vol. 2, no. 4, pp. 255–299, 1990.

[26] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Hei2004, pp. 152–166.

[27] Z. Ramezani, N. Smallbone, M. Fabian, and K. Åkesson, "Evaluating two semantics for falsification using an autonomous driving example," in *2019 IEEE 17th Int. Conf. on Industrial Informatics*, vol. 1, 2019, pp. 386–391.

[28] Z. Ramezani, J. L. Eddeland, K. Claessen, M. Fabian, and K. Åkesson, "Multiple objective functions for falsification of cyber-physical systems," *IFAC-PapersOnLine*, vol. 53, no. 4, pp. 417–422, 2020.

[29] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-TaLiR: A tool for temporal logic falsification for hybrid systems," in *Tools and Algorithms for the Construction and Analysis of Systems*, P. A. Abdulla and K. R. M. Leino, Eds. Springer Berlin Heidelberg, 2011, pp. 254–257.

[30] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *Computer Aided Verification*, T. Touili, B. Cook, and P. Jackson, Eds. Springer Berlin Heidelberg, 2010, pp. 167–170.

[31] Ramezani, Zahra, and Åkesson, Knut, "Technical report: The effect of input parameters on falsification of cyber-physical systems," *arXiv preprint arXiv:2209.07131*, 2022.

[32] G. Ernst, P. Arcaini, A. Donzé, G. Fainekos, L. Mathesen, G. Pedrielli, S. Yaghoubi, Y. Yamagata, and Z. Zhang, "ARCH-COMP 2019 Category Report: Falsification," in *ARCH19. 6th Int. Workshop on Applied Verification of Continuous and Hybrid Systems*, vol. 61. EasyChair, 2019, pp. 129–140.

[33] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[34] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[35] M. Wand, "Fast computation of multivariate kernel estimators," *Journal of Computational and Graphical Statistics*, vol. 3, no. 4, pp. 433–445, 1994.

[36] R. N. Bracewell and R. N. Bracewell, *The Fourier transform and its applications*. McGraw-Hill New York, 1986, vol. 31999.

[37] K. R. Rao, D. N. Kim, and J.-J. Hwang, *Fast Fourier Transform - Algorithms and Applications*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[38] D. Eriksson, M. Pearce, J. Gardner, R. D. Turner, and M. Poloczek, "Scalable global optimization via local bayesian optimization," *Advances in Neural Information Processing Systems*, vol. 32, pp. 5496–5507, 2019.

[39] Hvarfner, Carl and Stoll, Danny and Souza, Artur and Lindauer, Marius and Hutter, Frank and Nardi, Luigi, "πBO: Augmenting Acquisition Functions with User Beliefs for Bayesian Optimization," in *Tenth Int. Conf. of Learning Representations, ICLR 2022*, 2022, pp. 1–30.

[40] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *CoRR*, vol. abs/1708.06374, 2017.

[41] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proceedings of the 27th Int. Conf. on Int. Conf. on Machine Learning*, ser. ICML'10. Madison, WI, USA: Omnipress, 2010, p. 1015–1022.

[42] Ramezani, Zahra, and Šehić, Kenan and Nardi, Luigi and Åkesson, Knut, "Falsification of cyber-physical systems using Bayesian optimization," *arxiv.org/abs/2209.06735*, 2022.

[43] J. L. Eddeland, A. Donzé, S. Miremadi, and K. Åkesson, "Industrial temporal logic specifications for falsification of cyber-physical systems," in *ARCH20. 7th Int. Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)*, ser. EPiC Series in Computing, G. Frehse and M. Althoff, Eds., vol. 74. EasyChair, 2020, pp. 267–274.

[44] H. Brönnimann and M. T. Goodrich, "Almost optimal set covers in finite VC-dimension," *Discrete & Computational Geometry*, vol. 14, no. 4, pp. 463–479, Dec. 1995.

[45] J. L. Eddeland, A. Donzé, and K. Åkesson, "Multi-Requirement Testing Using Focused Falsification," in *HSCC '22: 25th ACM Int. Conf. on Hybrid Systems: Computation and Control, Milan, Italy, May 4 - 6, 2022*, E. Bartocci and S. Putot, Eds. ACM, 2022, pp. 4:1–4:11.

Zahra Ramezani received the BSc (2011) and MSc (2013) degrees in Electrical Engineering from Iran University of Science and Technology, Tehran, Iran. She received her PhD at the Department of Electrical Engineering (2022), Chalmers University of Technology, Gothenburg, Sweden. Her research interest is testing of cyber-physical systems.

Alexandre Donzé is the co-founder of Decyphir, building design automation tools for Cyber-Physical Systems. He received a computer science engineering degree from ENSIMAG in Grenoble in 2002 and a Ph.D in Math and Computer Science from Grenoble-Alpes University in 2007. He worked at CMU in Pittsburgh, Verimag Lab in Grenoble, and UC, Berkeley in California. He made several contributions to the field of CPS testing and verification with significant impact both in academia and in the industry.

Martin Fabian is since 2014 full Professor in Automation and Head of the Automation Research group at the Department of Electrical Engineering, Chalmers University of Technology. He received his PhD in Control Engineering from Chalmers University of Technology in 1995. His research interests include formal methods for automation systems in a broad sense, merging the fields of Control Engineering and Computer Science. He has authored more than 200 publications and is co-developer of the formal methods tool Supremica, which implements state-of-the-art algorithms for supervisory control synthesis.

Knut Åkesson (Member) is Professor with the Department of Electrical Engineering at Chalmers University of Technology, Gothenburg, Sweden. His main research is in using rigorous methods for the analysis of cyber-physical systems. Åkesson holds a Master of Science in Computer Science and Technology from Lund Institute of Technology at the University of Lund (1997), Sweden, and PhD in Control Engineering from Chalmers University of Technology (2002), Gothenburg, Sweden.