EPiC Series in Computing
Volume 80, 2021, Pages 195–202
8th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21)

Temporal Logic Falsification of Cyber-Physical Systems using Input Pulse Generators *

Zahra Ramezani[1], Alexandre Donzé[2], Martin Fabian[1], and Knut Åkesson[1]

[1] Chalmers University of Technology, Gothenburg, Sweden
{rzahra, fabian, knut}@chalmers.se
[2] Decyphir SAS, Moirans, France
alex@decyphir.com

Abstract

Falsification is a testing method for cyber-physical systems where numerical optimization is used to find counterexamples of a given specification that the system must fulfill. The falsification process uses quantitative semantics that play the role of objective functions to minimize the distance to falsifying the specification. Falsification has gained attention due to its versatile applicability, and much work exists on various ways of implementing the falsification process, often focusing on which optimization algorithm to use, or more recently, the semantics for the formal requirements. In this work, we look at some practical aspects of input generation, i.e., the mapping from parameters used as optimization variables to signals that form the actual test cases for the system. This choice is critical but often overlooked. It is assumed that problem experts can guide how to parameterize inputs; however, this assumption is often too optimistic in practice. We observe that pulse generation is a surprisingly good first option that can falsify many common benchmarks after only a few simulations while requiring only a few parameters per signal.

1 Introduction

Cyber-physical systems (CPSs) are often safety-critical systems; hence, assessing their correctness is essential. Testing is a commonly used approach to uncover faults in a *system under test* (SUT). Though testing cannot prove the absence of faults, it can nonetheless raise the confidence of a CPS behaving according to specification and is, therefore, an industrially interesting and useful endeavor.

Falsification of temporal logic properties is a testing method that tries to find *counterexamples* for specified behavior of CPSs. Falsification can be used when formal specifications exist, and the system can be simulated. Commonly, falsification uses optimization-based methods, where quantitative semantics associated with the specifications define the objective function. The objective function gives a measure of the distance to the specification being falsified. The selection of new test input cases uses the objective function values from previous simulations. Then, the test cases are modified so that the new test case is likely to have a lower objective value.

*This work was supported by the Swedish Research Council (VR) project SyTeC VR 2016-06204 and from the Swedish Governmental Agency for Innovation Systems (VINNOVA) under project TESTRON 2015-04893, and was partly supported by the Wallenberg AI, Autonomous Systems and Software program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Falsification of temporal logic properties is a black-box approach where only the input-output behavior of the SUT can be observed. Different parameterized input generators can be used. Typically, the parameters represent control points, and some interpolation between these points generates the input. Defining suitable input 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 no counterexample might fulfill the given constraints on the inputs, making the problem impossible to solve. Therefore, finding the right balance between the flexibility of the input generation and the dimension of the optimization problem is essential.

This question was part of the discussions leading to the first set of falsification benchmarks for the ARCH friendly competition, as reported in [7]. To ensure fair comparison between falsification algorithms, each benchmark was proposed with two instances: one with a fixed parameterization (given ranges and number of control points) for comparison purposes and one with a “free” parameterization, for which the only constraint was that input signals had to be piece-wise continuous signals.

In [11] different falsification methods were evaluated on the ARCH benchmark problem. One problem, the steam condenser (*SC*) [13], is of particular interest here. For the fixed paramerization with predetermined control points, no method succeeded in solving it whereas for the free parametrization, only [13], that used a simulated annealing global search in combination with an optimal control based local search on the infinite-dimensional input space, managed to successfully falsify the problem. However, this is not a black-box method and for many industrial problems we are restricted to black-box approaches. Since only a non black-box based method was able to falsify it, the *SC* problem was considered one of the hardest problems of the suite. However, examination of the falsifying input trace [13] revealed that it was essentially periodical, switching between the extreme values of the input parameter range, and we were able to falsify by using a simple pulse generator and optimizing over only the pulse period.

In [11] and others, it was already noted that extreme points, or corners, were particularly effective at solving many problem instances - quite intuitively, all those exhibiting some monotonicity with respect to the inputs. What the *SC* problem suggested is that periodic signals are a family of input generators with the potential for quick falsification of problems with loosely specified inputs. As a consequence, we extended the experimentation to the benchmarks [7] and [12] using the pulse generator in Breach [4] and found that we were able to solve a surprisingly high number of problems, as reported in this paper. Breach uses signal temporal logic (STL) [9] to formulate the specifications.

2 Falsification Problem and Pulse Inputs

Given a system $S : U \to X$ that maps an input signal $u \in U$ to an output signal $x \in X$, and a specification φ for x, the falsification problem is the problem of finding u such that x violates φ. The usual approach to solve this problem consists in defining a quantitative semantics $\rho(x)$ for φ such that if $\rho(x) < 0$ then x violates φ, defining a parameterization $p \to u(p)$, and minimizing ρ over some range of p until it becomes negative. In this paper, we consider the special case where inputs are defined as periodic square waves and evaluate their ability to solve a suite of existing falsification benchmark problems. A periodic square wave pulse can be defined by five parameters $p = (period, base, amplitude, delay, width)$ as shown in Figure 1.

Assuming the simulation time T, if *period* and *delay* are relatively small wrt T, we get regular square shaped inputs. However, we can also obtain signals of different types, e.g.,

- when *period* $\geq 2T$ and *width* ≥ 0.5, we get **constant signals**,
- when *period* $= 2T$ and *delay* varies in $[0, T]$ we get **single step inputs**.

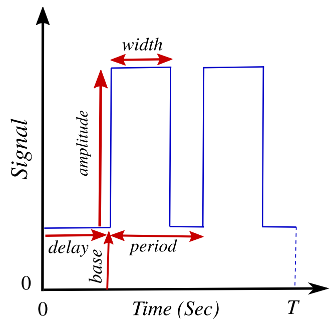

Figure 1: Pulse generator parameterization.

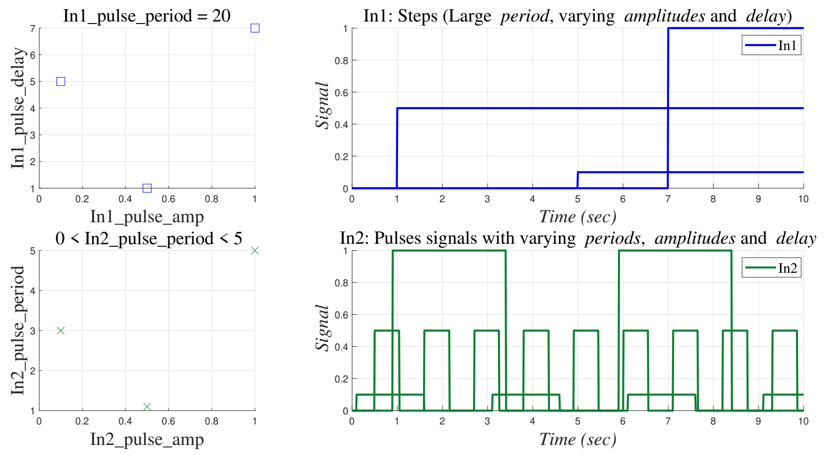

Figure 2: Parameter instances and their corresponding signals using the pulse generator.

A few examples of generated signals are shown in Figure 2.

3 Benchmark Problems

The following benchmark problems are considered. For details about the specifications, see [7, 12]:

- Automatic Transmission (*AT*): Two inputs, throttle in [0, 100], and brake in [0, 325];
- Automatic Transmission (*AT′*): Same as *AT*, but with different specifications. Throttle in [0, 100], and brake in [0, 500];
- Chasing Cars (*CC*): Two inputs, throttle and brake, both defined in [0, 1];
- Wind Turbine (*WT*): The input is the wind speed in [8, 16];
- $\Delta - \Sigma$ Modulator: Single input, three different ranges: [-0.35, 0.35], [-0.40, 0.40], [-0.45, 0.45]; with three initial conditions x_1^{init}, x_2^{init}, x_3^{init} in the range [-0.1, 0.1];
- Switched System (*SS*): Two inputs defined in the range [-1, 1]; Three different values are considered for parameter *thresh*: 0.7, 0.8, 0.9;

- Neural Network (*NN*): One input (reference) chosen in [1, 3];
- Fuel Control (*AFC*): Two inputs, throttle in [0, 61.2], and engine speed in [900, 1000];
- Steam Condenser (*SC*): One input with possible values in [3.99, 4.01].

4 Experimental Setup and Results

Given the parameter ranges for the pulse generator, the following strategies are evaluated:

- A Hybrid-Corner-Random (HCR) method [11] using corner and uniform random samples;
- The Nelder-Mead (NM) method [10], where 100 random points are evaluated before starting the simplex iterations based on a sorted list of the objective function values;
- The Line-search Falsification (LSF) method of [11].

The quantitative semantics used is the *Additive* semantics [2]. Each falsification is set to do a maximum of 1000 simulations. For each problem (system+specification), 20 runs are done to account for the falsification algorithms randomness. In the tables, two values are presented, the relative success rate of the falsification in percentage, and within parentheses the average number of simulations (rounded) per successful falsification. In the following we consider different parameterizations for the pulse generation.

Table 1: Results for all problems using only *period* as the varying input parameter and the HCR method. Other parameters are constant: *delay* = 0, *width* = 0.5, *base* and *amplitude* such that the pulse goes from the minimum to the maximum of the signal range.

Systems	**Specifications**								
	φ_1^{AT}	φ_2^{AT}	φ_3^{AT}	φ_4^{AT}	φ_5^{AT}	φ_6^{AT}	φ_7^{AT}	φ_8^{AT}	φ_9^{AT}
AT	**100** (5)	**100** (2)	**100** (23)	**100** (10)	**100** (6)	**100** (17)	0 (-)	0 (-)	0 (-)
	$\varphi_1^{AT'}$ $T = 20$	$\varphi_1^{AT'}$ $T = 30$	$\varphi_1^{AT'}$ $T = 40$	$\varphi_2^{AT'}$ $T = 10$	$\varphi_3^{AT'}$ $T = 4.5$	$\varphi_3^{AT'}$ $T = 5$	$\varphi_4^{AT'}$ $T = 1$	$\varphi_4^{AT'}$ $T = 2$	$\varphi_5^{AT'}$ $T = 1$
AT'	**100** (1)	**100** (1)	**100** (1)	**100** (2)	**100** (79)	**100** (23)	**100** (1)	**100** (1)	**100** (5)
	$\varphi_5^{AT'}$ $T = 2$	$\varphi_6^{AT'}$ $T = 10$	$\varphi_6^{AT'}$ $T = 12$	$\varphi_7^{AT'}$	$\varphi_8^{AT'}$ $\bar{\omega} = 3000$	$\varphi_8^{AT'}$ $\bar{\omega} = 3500$			
	100 (5)	0 (-)	0 (-)	0 (-)	0 (-)	0 (-)			
	φ_1^{CC}	φ_2^{CC}	φ_3^{CC}	φ_4^{CC}	φ_5^{CC}				
CC	**100** (5)	**100** (1)	**100** (5)	0 (-)	**100** (31)				
	φ_1^{WT}	φ_2^{WT}	φ_3^{WT}	φ_4^{WT}					
WT	**100** (4)	**100** (2)	**100** (4)	**100** (119)					
	$\varphi_1^{\Delta-\Sigma}$ $[-0.35, 0.35]$	$\varphi_1^{\Delta-\Sigma}$ $[-0.40, 0.40]$	$\varphi_1^{\Delta-\Sigma}$ $[-0.45, 0.45]$						
$\Delta - \Sigma$	**100** (24)	**100** (5)	**100** (6)						
	φ_1^{SS} (*thresh* = 0.7)	φ_1^{SS} (*thresh* = 0.8)	φ_1^{SS} (*thresh* = 0.9)						
SS	**100** (2)	**100** (2)	**100** (2)						
	φ_1^{NN}	φ_2^{NN}							
NN	**100** (9)	**100** (10)							
	φ_1^{AFC}	φ_2^{AFC}							
AFC	**100** (2)	**100** (2)							
	φ_1^{SC}								
SC	**100** (221)								

4.1 Input Parameter: *period*

We first consider *period* as the only varying parameter, in the interval $[0, 2T]$ where T is the simulation time, with the HCR method. The results are shown in Table 1. The first observation from Table 1 is that many specifications are falsified easily with a small number of simulations using HCR, i.e., a combination of extreme and random values. Using an optimization method or considering more varying input parameters are not required for those specifications. Furthermore, the pulse generator could falsify problems that were considered hard with other approaches [11]. For instance, *SC* was one problem that no method in [11] could falsify, whereas with HCR and a pulse generator, Table 1 shows that it is falsifiable with a reasonable number of simulations. Similarly, for falsifying the second specification of the *NN* problem, φ_2^{NN}, and the first specification of $\Delta - \Sigma$ in [11] optimization methods were required, while here HCR works well.

Table 2: Results for *SC* with *period* as the only varying input parameter and using NM and LSF.

Systems	Optimization Methods	Specifications
SC	Hybrid-Corner-Random	**100** (221)
	Nelder-Mead	**100** (81)
	Line-search Falsification	**100** (96)

Although using HCR works for *SC*, more simulations are needed than for the other falsified problems in Table 1. Hence, using optimization methods might decrease the number of simulations needed to falsify *SC*. For this purpose, in Table 2 HCR is compared with NM and LSF for *SC*. As can be seen, the optimization-based methods have better performance and manage to falsify *SC* with fewer simulations.

4.2 Input Parameters: *period* and *amplitude*

As earlier said, having few parameters is preferable, although it might affect the falsification success rate. When a system has more than one input, the system depends on more parameters which might make the specification harder to falsify. Hence, to falsify the specifications that could not be falsified in Table 1, it might be an advantage also to consider more parameters of the input signal.

Table 3 shows the results for two input parameters, *period* and *amplitude*, for the systems that could not be falsified in Table 1. The *amplitude* varies within the ranges defined for each problem in Section 3, where the base is set the lower bound of input ranges.

Table 3: Results for some problems with *period* and *amplitude* as input parameters using HCR and optimization-based methods; *delay* = 0, *width* = 0.5.

Systems	Optimization Methods	Specifications				
AT′		$\varphi_6^{AT'}$ $T = 10$	$\varphi_6^{AT'}$ $T = 12$	$\varphi_7^{AT'}$	$\varphi_8^{AT'}$ $\bar{\omega} = 3000$	$\varphi_8^{AT'}$ $\bar{\omega} = 3500$
	Hybrid-Corner-Random	95 (426)	**100** (53)	0 (-)	**100** (223)	0 (-)
	Nelder-Mead	**100** (224)	**100** (50)	0 (-)	75 (322)	0 (-)
	Line-search Falsification	**100** (218)	**100** (38)	0 (-)	**100** (186)	0 (-)
AT		φ_7^{AT}	φ_8^{AT}	φ_9^{AT}		
	Hybrid-Corner-Random	**100** (33)	**100** (43)	**100** (33)		
	Nelder-Mead	**100** (31)	**100** (30)	**100** (22)		
	Line-search Falsification	**100** (12)	**100** (16)	**100** (9)		
CC		φ_4^{CC}				
	Hybrid-Corner-Random	0 (-)				
	Nelder-Mead	0 (-)				
	Line-search Falsification	0 (-)				

As seen in Table 3, more specifications can be falsified, $\varphi_6^{AT'}$ both with $T = 10$ and $T = 12$, $\varphi_8^{AT'}$ with $\bar{\omega} = 3000$, and all specifications of the *AT* problem, φ_i^{AT}, $i = 7, 8, 9$. Only three specifications $\varphi_7^{AT'}$, $\varphi_8^{AT'}(\bar{\omega} = 3500)$, and φ_4^{CC} remain to be falsified. These are falsified in [11], although φ_4^{CC} is on the harder side. For all problems in Table 3, LSF performs the best, falsifying the specifications with a success rate of 100% with fewer simulations than the other two methods. The difference is most visible for $\varphi_6^{AT'}$ with $T = 10$ where both optimization methods can falsify with a 100% success rate and about half as many simulations as HCR; for $\varphi_8^{AT'}(\bar{\omega} = 3000)$, NM does not work as well as HCR and LSF.

Table 4: Results for the specification $\varphi_8^{AT'}(\bar{\omega} = 3500)$ with inverted pulse generators and *period* using HCR, NM, and LSF.

Systems	Optimization Methods	Specifications
$\varphi_8^{AT'}$ ($\bar{\omega} = 3500$)	Hybrid-Corner-Random	**100** (71)
	Nelder-Mead	**100** (68)
	Line-search Falsification	**100** (104)

Table 5: Experimental falsification results using HCR, NM and LSF

Systems	Optimization Methods	Specifications
AT'	Hybrid-Corner-Random	45 (480)
	Nelder-Mead	55 (215)
	Line-search Falsification	90 (285)
CC	Hybrid-Corner-Random	95 (189)
	Nelder-Mead	**100** (186)
	Line-search Falsification	**100** (218)

(a) Results for $\varphi_7^{AT'}$ and φ_4^{CC} with *period*, *amplitude*, and *width* as free parameters.

Systems	Optimization Methods	Specifications
CC	Hybrid-Corner-Random	95 (316)
	Nelder-Mead	**100** (197)
	Line-search Falsification	**100** (102)

(b) Results for φ_4^{CC} with *period*, *width*, and *delay* as free parameters.

We did some evaluation for the three remaining specifications with 0% success rate and realized that if we considered different assumptions, it would be possible to falsify at least one of them. For AT', recall that two inputs are defined, throttle in [0, 100] and brake in [0, 500]. Our idea was to favor situations where throttle was high when the brake was low and vice versa. For this, we set the *base* for throttle to 100 and *amplitude* to negative values from -100 to 0. This way, pulses of throttle are most of the time inverted with respect to pulses of the brake signal. The result with inverted pulse generators and also considering *period* as varying input for AT' is presented for $\varphi_8^{AT'}(\bar{\omega} = 3500)$ in Table 4. As can be seen, the specification $\varphi_8^{AT'}(\bar{\omega} = 3500)$ is falsified using inverted pulse generators with *period* and *amplitude*, but here NM performs the best.

4.3 Input Parameters: *period*, *amplitude*, and *width*

A third pulse parameter, *width* can be added as varying input parameter. The *width* can vary in the range $[0, 1]$ relative to the period. Table 5a, *period*, *amplitude* and *width* were used as variables to falsify $\varphi_7^{AT'}$ and φ_4^{CC}. For $\varphi_7^{AT'}$ LSF performs significantly better than the two other approaches. For φ_4^{CC}, the optimization-based approaches are able to successfully falsify the specification and perform slightly better than HCR.

4.4 Input Parameters: *period*, *width*, and *delay*

Finally, in Table 5b the effects of adding *delay* as free parameters is evaluated. The specification φ_4^{CC} is evaluated with a pulse generator with *period*, *width*, and *delay* as free parameters. It is assumed that *width* is within [0, 1], and *delay* in [0, 0.2T] where the upper delay bound is selected based on the simulation time, 20%. Both of the optimization approaches require fewer simulations on average than HCR.

4.5 Discussion and related work

From the evaluation in the previous section, we note that all evaluated benchmark problems in this paper could be falsified using the input pulse generator. It is certainly possible to formulate specifications that cannot be falsified using a pulse generator but the surprisingly strong performance on this set of benchmark problems shows that the approach has the potential for quick falsification of problems with loosely specified inputs. From our evaluation, we also observe that, for the harder problems, using an optimization-based approach rather than using HCR improves the performance. Optimization-based approaches are sensitive to the dimensionality of the optimization problem, though, so it is desirable to keep the number of optimization variables low, making the pulse generator an interesting way to parameterize the inputs with relatively few parameters. In this work, we started our evaluation by first considering only one free input parameter, then relaxing the problem by allowing more input parameters to be decided during the optimization part of the falsification. The preferred order to add parameters to the optimization problems, how many falsification attempts to run before relaxing the input and add more free parameters, etc, is a topic for future work. Another open problem is how to best deal with multiple input signals. For one requirement, we had the knowledge that inputs should be synchronous but inverted – can this be generalized?

Maybe the efficiency and versatility of pulse generation for falsification should not come as a surprise since its domain contains many typically stressful stimuli in the time domain, such as steps, and in the frequency domain with periodic signals, which makes it possible to uncover resonance effects such as the one for the steam condenser. On the other hand, counterexamples that require visiting specific regions of the state space in some specific order are intuitively going to be harder if not impossible to find with pulse inputs. This is illustrated by $\varphi_7^{AT'}$, which is falsified only if the gear signal follows a given sequence with timing constraints and was found to be the hardest to falsify here. Preliminary experiments with version 2.0 of the ATwSS benchmark [6] showed that requirements augmented with artificial input signals have also proven difficult to falsify with pulses. One reason is that the input dimensionality is increased from less than 3 to 13 signals.

Other works have considered input generation with varying control points or parameterization. In [3], the authors start with constant signals and add new control points incrementally. Other authors have implemented methods adapted from path-planning algorithms, such as randomly exploring trees [5]. More recent methods come from the (reinforcement) learning domain and also construct tree like structures [8] so that inputs are evaluated partially and on-the-fly. These methods are suited for the type of problems mentioned above requiring targeting a small region of the input signal space, but they might fail to converge toward counterexamples living in some high-frequency regions, simply because this will require adding too many control points. In [1], the authors introduce an input generator using the words accepted by a time automaton (TA) and show in particular that it can address the generation of complex periodic behaviors. This is much more general than the simple pulse generator that we consider in this paper and requires more effort in modeling the TA for input generation.

5 Conclusion

This paper focuses on the practical aspect of input generation to falsify cyber-physical systems using a simple structured input pulse generator. Different input types can be generated using a pulse generator, such as constant inputs, step inputs, and uniform time steps. The approach is suitable for black-box falsification where no assumptions are made on the SUT other than that the inputs can be changed, the system simulated, and the outputs observed.

The proposed method has been evaluated on standard benchmark problems, and the pulse generator was able to falsify all the benchmark problems. Based on the presented results, we can see that for the more challenging falsification problems using an optimization-based approach to decide on the parameters for the pulse generation, in general, reduces the number of simulations needed for a successful falsification.

For future work, it would be interesting to investigate how to add free parameters systematically and under which conditions to include additional parameters. The benchmark problems contain a relatively low number of input signals while industrial problems typically have a large number of input signals; evaluating the approach on industrial problems is future work.

References

[1] Benoît Barbot, Nicolas Basset, Thao Dang, Alexandre Donzé, James Kapinski, and Tomoya Yamaguchi. Falsification of Cyber-Physical Systems with Constrained Signal Spaces. In *NASA Formal Methods*, pages 420–439, Moffett Field, United States, May 2020.

[2] Koen Claessen, Nicholas Smallbone, Johan Eddeland, Zahra Ramezani, and Knut Åkesson. Using valued booleans to find simpler counterexamples in random testing of cyber-physical systems. *IFAC-PapersOnLine*, 51(7):408–415, 2018.

[3] Jyotirmoy Deshmukh, Xiaoqing Jin, James Kapinski, and Oded Maler. Stochastic Local Search for Falsification of Hybrid Systems. In Bernd Finkbeiner, Geguang Pu, and Lijun Zhang, editors, *Automated Technology for Verification and Analysis*, Lecture Notes in Computer Science, pages 500–517, Cham, 2015. Springer International Publishing.

[4] Alexandre Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *International Conference on Computer Aided Verification*, pages 167–170. Springer, 2010.

[5] Tommaso Dreossi, Thao Dang, Alexandre Donzé, James Kapinski, Xiaoqing Jin, and Jyotirmoy V. Deshmukh. Efficient Guiding Strategies for Testing of Temporal Properties of Hybrid Systems. In *NASA Formal Methods*, volume 9058, pages 127–142. Springer International, Cham, 2015.

[6] Johan Lidén Eddeland, Alexandre Donzé, Sajed Miremadi, and Knut Åkesson. Industrial temporal logic specifications for falsification of cyber-physical systems. In Goran Frehse and Matthias Althoff, editors, *ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)*, volume 74 of *EPiC Series in Computing*, pages 267–274, 2020.

[7] Gidon Ernst, Paolo Arcaini, Alexandre Donzé, Georgios Fainekos, Logan Mathesen, Giulia Pedrielli, Shakiba Yaghoubi, Yoriyuki Yamagata, and Zhenya Zhang. ARCH-COMP 2019 category report: Falsification. In *ARCH@ CPSIoTWeek*, pages 129–140, 2019.

[8] Gidon Ernst, Sean Sedwards, Zhenya Zhang, and Ichiro Hasuo. Fast Falsification of Hybrid Systems Using Probabilistically Adaptive Input. In David Parker and Verena Wolf, editors, *Quantitative Evaluation of Systems*, Lecture Notes in Computer Science, pages 165–181, Cham, 2019. Springer International Publishing.

[9] Oded Maler, Dejan Nickovic, and Amir Pnueli. *Checking Temporal Properties of Discrete, Timed and Continuous Behaviors*, pages 475–505. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[10] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[11] Zahra Ramezani, Koen Claessen, Nicholas Smallbone, Martin Fabian, and Knut Åkesson. Testing cyber-physical systems using a line-search falsification method. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[12] Zahra Ramezani, Johan Lidén Eddeland, Koen Claessen, Martin Fabian, and Knut Åkesson. Multiple objective functions for falsification of cyber-physical systems. *IFAC-PapersOnLine*, 53(4):417–422, 2020.

[13] Shakiba Yaghoubi and Georgios Fainekos. Gray-box adversarial testing for control systems with machine learning components. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 179–184, 2019.