EPiC Series in Computing
Volume 90, 2022, Pages 204–221
Proceedings of 9th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH22)

ARCH-COMP 2022 Category Report: Falsification with Ubounded Resources*

Gidon Ernst[1], Paolo Arcaini[2], Georgios Fainekos[3], Federico Formica[4], Jun Inoue[5], Tanmay Khandait[6], Mohammad Mahdi Mahboob[4], Claudio Menghi[4], Giulia Pedrielli[6], Masaki Waga[7], Yoriyuki Yamagata[5], and Zhenya Zhang[8]

[1] Ludwig-Maximilians-University (LMU), Munich, Germany `gidon.ernst@lmu.de`
[2] National Institute of Informatics (NII), Tokyo, Japan `arcaini@nii.ac.jp`
[3] Toyota Research Institute of North America `georgios.fainekos@toyota.com`
[4] McMaster University, Canada `{formicaf,mahbom2,menghic}@mcmaster.ca`
[5] National Institute of Advanced Industrial Science and Technology (AIST), Osaka, Japan `{jun.inoue,yoriyuki.yamagata}@aist.go.jp`
[6] Arizona State University (ASU), Tempe, USA `{tkhandai,gpedriel}@asu.edu`
[7] Kyoto University, Japan `mwaga@fos.kuis.kyoto-u.ac.jp`
[8] Kyushu University, Japan `zhang@ait.kyushu-u.ac.jp`

Abstract

This report presents the results from the 2022 friendly competition in the ARCH workshop for the falsification of temporal logic specifications over Cyber-Physical Systems. We briefly describe the competition settings, which have been inherited and adapted from the previous years, give background on the participating teams and tools, and discuss the selected benchmarks. In this year's competition, in addition to the result validation introduced in the previous year, we change the experimental settings for a better account of the difficulty of benchmarks and for a better comparability between the tools.

Data: https://gitlab.com/goranf/ARCH-COMP, https://dx.doi.org/10.5281/zenodo.7359624

1 Introduction

We report on one category of the friendly competition associated with the ARCH 2022 workshop. The goal of the competition is to compare the state-of-the-art of tools for testing and verification of various types of hybrid systems. The competition is organized in different categories, with different specifications (computing reachable regions, checking temporal properties) and varying dynamics in the system models (such as linear/non-linear and hybrid).

This report concerns the *falsification category*, which targets the black-/greybox analysis of executable models with respect to requirements expressed in temporal logic with time bounds,

*The falsification category was coordinated by the first author. The remaining authors represent all participants who have contributed results and/or text to this report and they are listed alphabetically.

encoded in *Metric Temporal Logic (MTL)* [23] or *Signal Temporal Logic (STL)* [24]. The task is to search for initial system configurations and time-varying inputs subject to given constraints that steer the system into a violation status with respect to the temporal requirements. Typical approaches are simulation-based and employ quantitative metrics [16, 17] of how close a given input is to violating a requirement ("robustness semantics"). Research in this area has produced a variety of techniques, mature tools, and practical applications; these are described in overview survey articles [4, 7]. For past instalments of this competition 2017–2021 see [8, 9, 13, 12, 11]. The benchmark set developed by this competition series can be seen as a baseline for research in the area (cf. [10]), and we encourage authors to compare to the results presented here.

The competition of 2022 followed the structure of previous years: Once the benchmarks are agreed on, the participants run the experiments on their own machines and submit the results including concrete input traces that witness falsification. We therefore continue validating these counterexamples found by different tools to ensure the correctness of the results. Besides, there are four notable changes in the competition this year:

- The wind turbine model has been removed from the competition as, in previous editions, has shown to be too easy to falsify (i.e., all tools were able to falsify it in most of the trials; moreover, some tools falsified it with very few simulations). Moreover, doubts have been raised whether the current parameterization of the wind input constitutes a meaningful falsification challenge.
- We do not impose anymore a maximal number of simulations that can be run in one falsification trial (in 2021 competition, the maximum number was set to 300). The rationale of this choice is that we want to obtain a more accurate assessment of the difficulty of the benchmarks.
- We reduced the number of trials from 50 to 10, in order to keep the overall experiment time low. We had to do this for allowing a higher number of simulations in each trial (see previous point). We believe that 10 trials should still provide some reasonable statistics over potential results.
- We also aim to report separately the time taken by simulation in a falsification trial and the total time, such that the ratio of the two to factors out differences in the machine setup between the participants. The aim is to understand the cost of different falsification algorithms and implementations, regardless of the simulation time. However, the collected data was not consistent and has therefore been left out in this report.

ARCH-COMP 2022 Prize: We are very happy to announce that the Ψ-TaLiRo team has been awarded the ARCH-COMP prize. The jury, consisting of group leaders and workshop participants, appreciates the technical achievements embodied in PSY-TaLiRo and furthermore recognises the long and continuous stream of contributions that the team has made to the community. The prize goes to the participants Tanmay Khandait, who in particular contributed a lot to the organization of this year's competition, Giulia Pedrielli, and Georgios Fainekos and recognizes contributions to Ψ-TaLiRo by Quinn Thibeault, Jacob Anderson, and Aniruddh Chandratre.

Data Availability. The models and validation results produced by this competition are available through the shared GitLab repository at `https://gitlab.com/goranf/ARCH-COMP`, notably in the subfolders `models/FALS` and `2022/FALS`. An archive containing the traces submitted for validation will be made available at `https://dx.doi.org/10.5281/zenodo.7359624`. This archive contains the result of validation and instructions to re-validate the results.

2 Benchmark Definitions

2.1 Input Parameterization

Arbitrary piece-wise continuous input signals (Instance 1). This option leaves the input specification up to the participants. The search space is, in principle, the entire set of piece-wise continuous input signals (i.e., which permit discontinuities), where the values for each individual dimensions are from a given range. Additional constraints that were suggested are finite-number of discontinuity and finite variability for all continuous parts of inputs. Further, each benchmark may impose further constraints. Participants may instruct their tools to search a subset of the entire search space, notably to achieve finite parametrization, and then to apply an interpolation scheme to synthesize the input signal.

However, the participants agreed that such a choice must be "reasonable" and should be justified from the problem's specification without introducing external knowledge about potential solutions. Moreover, more general parametrizations that are shared across requirements and benchmark models were preferable. Due to the diversity of benchmarks, it was decided to evaluate the proposed solutions using common sense.

Constrained input signals (Instance 2). This option precisely fixes the format of the input signal, potentially allowing discontinuities. An example input signal would be piecewise constant with k equally spaced control points, with ranges for each dimension of the input, disabling interpolation at Simulink input ports so that tools don't need to up-sample their inputs. The arguments in favor of that are increased comparability of results. As possible downside was mentioned that optimization-based tools (S-TaLiRo and Breach) are just compared with respect to their optimization algorithm. Nevertheless such a comparison is still meaningful in particular with the other, fundamentally different approaches to falsification that have entered the competition since.

2.2 Models and Requirements

A brief description of the benchmark models follows, the formal requirements are shown in Table 1 as STL/MTL formulas. The conjunctive requirements introduced in this year are AT6abc, NNx, and CCx, as marked in the table.

Automatic Transmission (AT). This model of an automatic transmission encompasses a controller that selects a gear 1 to 4 depending on two inputs (throttle, brake) and the current engine load, rotations per minute ω, and car speed v. It is a standard falsification benchmark derived from a model by Mathworks and has been proposed for falsification in [20].

Input specification: $0 \leq \mathit{throttle} \leq 100$ and $0 \leq \mathit{brake} \leq 325$ (both can be active at the same time). Constrained input signals (instance 2) permit discontinuities at most every 5 time units. Requirements are specific versions of those in [20] where the parameters have been chosen to be somewhat difficult.

Fuel Control of an Automotive Powertrain (AFC). The model is described in [22] and has been used in two previous instalments of this competition [8, 9]. The specific limits used in the requirements are chosen such that falsification is possible but reasonably hard.

The constrained input signal (instance 2) fixes the throttle θ to be piecewise constant with 10 uniform segments over a time horizon of 50 with two modes (normal and power corresponding to

Table 1: Requirement formulas for the benchmarks

Key	STL formula	Remarks/Constraints
AT1	$\square_{[0,20]} v < 120$	
AT2	$\square_{[0,10]} \omega < 4750$	
AT51	$\square_{[0,30]}((\neg g1 \wedge \circ\ g1) \rightarrow \circ\ \square_{[2,2.5]} g1)$	where $\circ\ \phi \equiv \Diamond_{[0.001,0.1]}\ \phi$
AT52	$\square_{[0,30]}((\neg g2 \wedge \circ\ g2) \rightarrow \circ\ \square_{[0,2.5]} g2)$	
AT53	$\square_{[0,30]}((\neg g3 \wedge \circ\ g3) \rightarrow \circ\ \square_{[0,2.5]} g3)$	
AT54	$\square_{[0,30]}((\neg g4 \wedge \circ\ g4) \rightarrow \circ\ \square_{[0,2.5]} g4)$	
AT6a	$(\square_{[0,30]} \omega < 3000) \rightarrow (\square_{[0,4]} v < 35)$	
AT6b	$(\square_{[0,30]} \omega < 3000) \rightarrow (\square_{[0,8]} v < 50)$	
AT6c	$(\square_{[0,30]} \omega < 3000) \rightarrow (\square_{[0,20]} v < 65)$	
AT6abc	AT6a $\wedge$ AT6b $\wedge$ AT6c	cojunctive requirement
AFC27	$\square_{[11,50]}((\mathit{rise} \vee \mathit{fall}) \rightarrow (\square_{[1,5]} \lvert\mu\rvert < \beta))$	$0 \leq \theta < 61.2$ (normal mode)
AFC29	$\square_{[11,50]} \lvert\mu\rvert < \gamma$	$0 \leq \theta < 61.2$ (normal mode)
AFC33	$\square_{[11,50]} \lvert\mu\rvert < \gamma$	$61.2 \leq \theta \leq 81.2$ (power mode)
	where $\beta = 0.008$, $\gamma = 0.007$	
	$\mathit{rise} = (\theta < 8.8) \wedge (\Diamond_{[0,0.05]}(\theta > 40.0))$	
	$\mathit{fall} = (\theta > 40.0) \wedge (\Diamond_{[0,0.05]}(\theta < 8.8))$	
NN	$\square_{[1,37]}(\lvert Pos - Ref\rvert > \alpha + \beta\lvert Ref\rvert \rightarrow \Diamond_{[0,2]}\square_{[0,1]}\neg(\alpha + \beta\lvert Ref\rvert \leq \lvert Pos - Ref\rvert))$	
	where $\alpha = 0.005$ and $\beta = 0.03$	
NNx	$\Diamond_{[0,1]}(Pos > 3.2) \wedge \Diamond_{[1,1.5]}(\square_{[0,0.5]}(1.75 < Pos < 2.25)) \wedge \square_{[2,3]}(1.825 < Pos < 2.175)$	conjunctive requiremet $1.95 \leq Ref \leq 2.05$
WT1	$\square_{[30,630]} \theta \leq 14.2$	
WT2	$\square_{[30,630]} 21000 \leq M_{g,d} \leq 47500$	
WT3	$\square_{[30,630]} \Omega \leq 14.3$	
WT4	$\square_{[30,630]} \Diamond_{[0,5]} \lvert\theta - \theta_d\rvert \leq 1.6$	
CC1	$\square_{[0,100]} y_5 - y_4 \leq 40$	
CC2	$\square_{[0,70]} \Diamond_{[0,30]} y_5 - y_4 \geq 15$	
CC3	$\square_{[0,80]}((\square_{[0,20]} y_2 - y_1 \leq 20) \vee (\Diamond_{[0,20]} y_5 - y_4 \geq 40))$	
CC4	$\square_{[0,65]} \Diamond_{[0,30]} \square_{[0,20]} y_5 - y_4 \geq 8$	
CC5	$\square_{[0,72]} \Diamond_{[0,8]}((\square_{[0,5]} y_2 - y_1 \geq 9) \rightarrow (\square_{[5,20]} y_5 - y_4 \geq 9))$	
CCx	$\bigwedge_{i=1..4} \square_{[0,50]}(y_{i+1} - y_i > 7.5)$	conjunctive requirement
F16	$\square_{[0,15]} \mathit{altitude} > 0$	
SC	$\square_{[30,35]}(87 \leq \mathit{pressure} \wedge \mathit{pressure} \leq 87.5)$	

feedback and feedforward control), and the engine speed ω to be constant with $900 \leq \omega < 1100$ to capture the input profile outlined in [22] and to match the previous competitions. For this reason, we do not consider the unconstrained (instance 1) input specification. Faults are disabled (e.g. by setting `fault_time` > 50).

Neural-network Controller (NN). This benchmark is based on MathWork's neural network controller for a system that levitates a magnet above an electromagnet at a reference position.[1] It has been used previously as a falsification demonstration in the distribution of Breach. The model has one input, a reference value *Ref* for the position, where $1 \leq Ref$ and $Ref \leq 3$. It outputs the current position of the levitating magnet *Pos*. The input specification for instance 1 requires discontinuities to be at least 3 time units apart, whereas instance 2 specifies an input signal with exactly three constant segments. The time horizon for the problem is 40. The requirement ensures that after changes to the reference, the actual position eventually stabilizes around that value with small error.

Chasing cars (CC). The model is derived from Hu et al. [21] which presents a simple model of an automatic chasing car. Chasing cars (CC) model consists of five cars, in which the first car is driven by inputs (*throttle* and *brake*), and other four are driven by Hu et al.'s algorithm. The output of the system is the location of five cars y_1, y_2, y_3, y_4, y_5. The properties to be falsified are constructed artificially, to investigate the impact of complexity of the formulas to falsification. The input specifications for instance 1 allows any piecewise continuous signals while the input specification for instance 2 constraints inputs to piecewise constant signals with control points for each 5 seconds, i.e., 20 segments.

Aircraft Ground Collision Avoidance System (F16). The model has been derived from the one presented in [19]. The F16 aircraft and its inner-loop controller for Ground Collision avoidance have been modeled using 16 continuous variables with piece-wise nonlinear differential equations. Autonomous maneuvers are performed in an outer-loop controller that uses a finite-state machine with guards involving the continuous variables. The system is required to always avoid hitting the ground during its maneuver starting from all the initial conditions for roll, pitch, and yaw in the range $[0.2\pi, 0.2833\pi] \times [-0.4\pi, -0.35\pi] \times [-0.375\pi, -0.125\pi]$.[2] Since the benchmark has no time-varying input, there is no distinction between instance 1 and instance 2. The requirement is checked for a time horizon equal to 15.

Steam condenser with Recurrent Neural Network Controller (SC). The model is presented in [35]. It is a dynamic model of an steam condenser based on energy balance and cooling water mass balance controlled with a Recurrent Neural network in feedback. The time horizon for the problem is 35 seconds. The input to the system can vary in the range $[3.99, 4.01]$. For instance 2, the input signal should be piecewise constant with 20 evenly spaced segments.

3 Participants

We briefly describe in alphabetical order all participating tools, the respective main ideas of the underlying approaches, followed by details on how each tool was set up for the competition.

[1] https://au.mathworks.com/help/deeplearning/ug/design-narma-l2-neural-controller-in-simulink.html

[2] The report from 2019 [13] erroneously specifies: $[0.2\pi, 0.2833\pi] \times [-0.5\pi, -0.54\pi] \times [0.25\pi, 0.375\pi]$, however, previous results were in fact obtained with the correct range.

3.1 ARIsTEO

Description. ARIsTEO [28] is a Matlab toolbox for test case generation against system specifications presented in STL and it is developed on the top of S-TaLiRo. ARIsTEO is designed to targeting a large and practically-important category of CPS models, known as *compute-intensive* CPS (CI-CPS) models, where a single simulation of the model may take hours to complete. ARIsTEO embeds black-box testing into an iterative approximation-refinement loop. At the start, some sampled inputs and outputs of the model under test are used to generate a surrogate model that is faster to execute and can be subjected to black-box testing. Any failure-revealing test identified for the surrogate model is checked on the original model. If spurious, the test results are used to refine the surrogate model to be tested again. Otherwise, the test reveals a valid failure. ARIsTEO is publicly available under the General Public License (GPL).[3]

Setup. ARIsTEO provides the same interface and parameters as S-TaLiRo, while providing additional configuration options. We had used an ARX model (ARX-2) with order $na = 2$, $nb = 2$, and $nk = 2$[4] as structure for the surrogate model used in the approximation-refinement loop of ARIsTEO. For models with multiple inputs and outputs the dimension of the matrix na, nb and nk is changed depending on the number of inputs and outputs. We used the default configuration of S-TaLiRo for searching failure-revealing revealing tests on the surrogate model. We considered the same parametrization of S-TaLiRo for the input signals. The original Simulink model was executed once to learn the initial surrogate model. The cut-off values for the number of simulations of the original model and for the number of simulations of the surrogate model (per trial) were set to 300. The results of ARIsTEO can further improve by (i) using configurations for the surrogate model that provide more accurate approximations of the original models and more effectively guide the search toward faulty inputs; and (ii) using the SOAR option of S-TaLiRo that significantly improved the results of S-TaLiRo compared with the last edition of this competition.

3.2 FalCAuN

Description. FalCAuN [34] is an experimental tool for testing a Simulink model using black-box checking [32], an automated testing method based on active automata learning and model checking. In FalCAuN, the input and the output signals of the Simulink model are discretized in time and values, and the model is abstracted into a black-box Mealy machine. FalCAuN learns the Mealy machine and conducts model checking to find a counterexample. FalCAuN is designed to efficiently falsify a Simulink model against multiple specifications by reusing the learned Mealy machine. FalCAuN is publicly available under General Public License (GPL) v3[5].

We utilize the *discrete-time* semantics of STL, which is essentially the same as the semantics of LTL. Because of such discretization, the control points must be fine enough to capture the timing constraints in the STL formula. For example, in order to capture the timing constraint $\Diamond_{[0,0.05]}$, the duration between the control points must be at most 0.05. We note that due to the use of the discrete-time semantics, the signal reported as a counterexample by FalCAuN may not falsify the model in terms of the continuous-time semantics.

[3] https://github.com/SNTSVV/ARIsTEO

[4] https://nl.mathworks.com/help/ident/ref/arx.html

[5] https://github.com/MasWag/FalCAuN

Setup. For the signal discretization, we have the following hyperparameters: the (constant) duration of the intervals between samples, the input signal values at the control points, and the thresholds of the output signal values for the discretization. We used the shortest duration between the control points such that the LTL encoding of the STL formula is small enough for the back-end model checker LTSMin. The duration ranges from 1.0 to 10.0 time units. In most benchmarks, we let the input signal values be the maximum and the minimum of the range. The exceptions are as follows.

- In AT6a, AT6b, and AT6c, the throttle can be 50 in addition to 0 and 100.
- In AT2 instance 2, the throttle and the brake can be 5 and 6 evenly spaced values, respectively, i.e., the value of the throttle can be one of 0, 25, 50, 75, and 100.
- In SCa, the input can be 4.00 in addition to 3.99 and 4.01.

In most benchmarks, we let the threshold of the output signal values be the thresholds in the STL formula. The exceptions are as follows.

- In AT1 and AT2 instance 2, we have the common thresholds: 120 for the speed and 4750 for the RPM.
- In AT6a, AT6b, and AT6c, we have the common thresholds: 35, 50, and 65 for the speed and 3000 for the RPM.
- In CC1, CC2, and CC3, we have the common thresholds: 15 and 40 for $y_5 - y_4$ and 20 for $y_2 - y_1$.

3.3 falsify

Description. falsify is an experimental program which solves falsification problems of safety properties by reinforcement learning [36]. falsify uses a *grey-box* method, that is, it learns system behavior by observing system outputs during simulation. falsify is currently implemented by a deep reinforcement learning algorithm *Asynchronous Advantage Actor-Critic* (A3C) [29].

Setup. The input specification uses piecewise constant function with discontinuities spaced in even intervals ΔT. $\Delta T = 1$ for all models except for SC in which $\Delta T = 0.1$ is used. The choice for the SC model was $\Delta T = 0.1$ model because Instance 2 uses $\Delta T = 1.75$, which is near to $\Delta T = 1$.

3.4 FalStar

Description. FalStar [14] is an a falsification tool that explores the idea to construct falsifying inputs incrementally in time, thereby exploiting potential time-causal dependencies in the problem. The algorithm used in this competition is called adaptive Las-Vegas tree search (aLVTS). The main idea is to try "simple" inputs first, i.e., to scale gracefully with the intuitive combinatorial hardness of the benchmark problem [15]. This is achieved by sampling from input domains with gradually finer temporal and spatial resolution, starting with extreme values. The approach can be regarded as a more reasonable baseline over random sampling, which takes the structure of the problem into account, but which neither relies on sophisticated optimization methods nor insight into the models themselves. The code is publicly available under the BSD license.[6]

Setup. The search space for instance 1 included piecewise constant inputs (the only parameterization currently supported), ranging from 2 upto 4 control points at which discontinuities

[6]https://github.com/ERATOMMSD/falstar

are allowed (resp. upto 3 for NN). In this configuration FalStar benefits from a low number of control points and is more likely to try inputs with fewer control points first. For the AT benchmarks it was clear beforehand that this choice suffices to falsify all benchmarks, and the setting was then kept for the remaining experiments. Instance 2 input signals are parameterized with the number of control points as specified by the respective benchmarks. The F16 benchmark does not have a time-varying input and therefore the strategy proposed in [14] is not applicable. Instead, global optimization with Nelder/Mead is used on this benchmark, which is effective.

3.5 ForeSee

Description. In falsification, the *scale problem* can occur when the signals used in the specification have different scales (e.g., rpm and speed): namely, the contribution of a signal could be *masked* by another one when computing robustness. ForeSee [37] (FORmula Exploitation by Sequence trEE) tackles this problem by introducing a new robustness definition, called *QB-Robustness*, which combines quantitative robustness and classical Boolean satisfaction. QB-Robustness does not require comparing (i.e., by minimum or maximum) robustness values of different sub-formulas, so possibly avoiding the scale problem. However, in order to be computed, QB-Robustness requires the selection of a sequence of sub-formulas along the syntax tree of the specification for which to compute the quantitative robustness. Different sub-formulas sequences can be more or less effective in mitigating the scale problem.

ForeSee implements a falsification strategy based on a Monte Carlo Tree Search over the structure of the formal specification: first, by tree traversal, it identifies the sub-formulas sequence; then, on the leaves, it performs numerical hill-climbing optimization, with the aim of falsifying the selected sub-formulas. ForeSee is the spiritual successor of FalStar/MCTS from [13, 12]. It is publicly available under GNU General Public License (GPL) v3.[7]

Setup. Since ForeSee is implemented on the basis of Breach, it provides the same interface of Breach, namely, users can characterize the shape of input signals with a number of options, including piecewise constant, piecewise linear, pulse, etc. In this report, we regulate the shape of input signals with piecewise constant, parametrized by the number of *control points*.

In the current implementation of ForeSee, only CMA-ES [2] is provided as the optimizer; this is due to our insight in the performances of different optimizers, in which CMA-ES outperforms other optimizers. However, involving other optimizers is not difficult for ForeSee, and will be considered in the future releases.

Since ForeSee technically relies on Monte Carlo Tree Search (MCTS), the hyperparameters in MCTS need to be properly selected. As a default setting, we use 0.2 as the scalar in the UCB1 algorithm, that takes a balance of *exploration* and *exploitation*; and we set 10 generations as the budget for the playout phase of MCTS.

3.6 S-TaLiRo

Description. S-TaLiRo [1] is a Matlab toolbox for monitoring and test case generation against system specifications presented in STL (or MTL). The test cases are automatically generated using optimization techniques guided by formal requirements in STL in order to find falsifying systems behaviors. The tool supports different optimization algorithms. In past competitions, the Stochastic Optimization with Adaptive Restarts (SOAR) [27] framework was used for all the benchmarks except for choosing instance 1 type inputs in Steam Condenser model. In

[7] https://github.com/choshina/ForeSee

that benchmark, Simulated annealing global search was combined with a local optimal control based search [35]. For the 2021 competition, we have used the minSOAR framework [26, 25] for the newly proposed conjunctive benchmarks. The minSOAR framework can be thought of as an extension to SOAR which can handle multiple conjunctive requirements by modeling information from each resulting robustness component. S-TaLiRo is publicly available on-line under General Public License (GPL) [8].

Setup. In S-TaLiRo, input signals are parameterized in two ways: the number of control points for the input signal, and the time location of those control points during simulation. The number of control points for each input signal is given by the user forming an optimization problem with search space dimension the same as the number of control points. An option is provided to the user to add to the search space the timing of the control points, but this option is not used in the competition. For this competition, the control point time locations are evenly spaced over the duration of the simulation for all the benchmarks except for the SC problem instance 1.

For the transmission model the [*throttle*, *brake*] control points are interpolated with the *pchip* function, with $[7, 3]$ as the number of control points in specifications 1-6 and $[4, 2]$ for 7-9 to reduce the dimensionality of the search space. For the Neural model, we use 13 control points to yield piecewise constant signals of 3.33 seconds apart. The Wind Turbine used the default model input of 126 control points interpolated linearly. For the SC model, Simulated Annealing (SA) global search was utilized in combination with an optimal control based local search on the infinite dimensional input space. The SA global search utilizes piecewise constant inputs with 12 possibly uneven time durations.

3.7 Ψ-TaLiRo

Description. ΨTaLiRo [33], the python version of S-TaLiRo, is an open-source toolbox for temporal logic robustness-guided falsification of Cyber-Physical Systems (CPS). This toolbox, which is completely modular, helps in the generation of test cases for falsification of system under test using a common interface for temporal logic monitors. While the toolbox provides inbuilt optimizers (DA, Uniform Random etc.), one can also develop optimizers and pass it ΨTaLiRo. The tool box is publicly available on-line under General Public License (GPL) [9]. For this competition, we provide results with two algorithms

1. **LS-emiBO** aims to model embed the dependencies between the different components in a conjunctive requirement. This algorithm works by considering a component chosen from a classifier built between the points and component with minimum robustness, and then sampling a point that maximizes the Expected Improvement (EI) function. It is important to note that this algorithm turns into a simple Bayesian Optimization if we do not have conjunctive requirements.
2. **Part-X** adaptively partitions the search space to enclose the falsifying points. The algorithm uses local Gaussian process estimates in order to adaptively branch and sample within the input space. The partitioning approach not only helps us identify the zero level-set, but also to circumvent issues that rise due to the fact that the robustness is discontinuous. In fact, the only assumption we need on the robustness function is that it is a locally continuous function [31].

[8] https://sites.google.com/a/asu.edu/s-taliro
[9] https://gitlab.com/sbtg/psy-taliro

Setup. In ΨTaLiRo, input signals to blackbox models are parameterized with control locations and their corresponding time stamps, which then leads to the formation of an optimization problem with the dimensionality depending upon the number of control points. The input signals along with their corresponding time stamps are interpolated depending on the instance. For this competition, all signals have evenly spaced control points and are interpolated using pchip for instance 1 and piecewise constant for instance 2, and we utilize RTAMT [30] for robustness calculation. The LS-emiBO optimizer samples 20 points from the search space and then sequentially sample points until a falsification is found or the maximum budget of 2000 evaluations is reached. Finally, the Part-X optimizer, which provides probabilistic guarantees, starts with an initialization budget $n_0 = 20$, per-subregion budget for unclassified subregions with $n_{\text{BO}} = 20$, classified subregions budget $n_c = 50$, maximum budget $T = 2000$, number of Monte Carlo iterations $R = 20$, number of evaluations per iterations $M = 500$, number of cuts $B = 2$, and classification percentile $\delta_u = 0.05$. Also, we used $\delta_v = 0.001$ to identify dimensions that should not be branched. We provide the probabilistic guarantees in table 2

4 Evaluation & Validation

Falsification tools were instructed to run each individual requirement 10 times, to account for the stochastic nature of most algorithms. We report the falsification rate, i.e., the number of trials where a falsifying input was found, as well as the median and mean of the number of simulations required to find such input (not including the unsuccessful runs in the aggregate). There was no cut-off for the number of simulations imposed on the experiments (last years had a maximum of 300).

The results were provided by the participants and have therefore been obtained on multiple platforms with varying resources and different MATLAB/Simulink versions. In contrast to prior years, all results have been obtained this year with the new competition settings (cf. section 1).

We continue the effort to valudate results, which has been established in 2021. The overarching goal is to ensure that the comparison reported here is meaningful, and the approach taken accounts for several potential sources of error, both for technical reasons or because of human error. The hypothetical case of cheating participants was not regarded likely, and we emphasize upfront that no indication whatsoever for dishonest behavior was found. Rather, the goal is to establish a higher standard of quality of evaluation results, that can ultimately benefit any future work in simulation-based falsification: Just like the benchmark set established by this community gets adopted by experiments in the literature, validation of results using an independent reference checker should become standard, too. To that end, for each falsification trial per requirement, we collected the following information:

- information about which benchmark (model + requirement identifier)
- the initial conditions and time-series input signal resulting from that trial
- whether the signal is expected to falsify the requirement
- if available, a robustness value derived from running the input through the model
- optionally, the corresponding output signal, and further information such as time stamps or wall-clock times

In the following, we will refer to the this information as the "reported" result. Technical details on the reporting format are documented here: `https://gitlab.com/gernst/ARCH-COMP/-/blob/FALS/2021/FALS/Validation.md`. Multiple questions were identified as concerns for validation, albeit not all points were achieved or fully taken into account:

1. does the reported input signal adhere to the valid ranges of input for that particular model, as described in section 2?
2. does the reported input signal adhere to the constraints of the respective instance?
3. is the reported verdict correct, i.e., whether running this signal through the model produces a falsifying trace?
4. is the reported robustness value consistent with the verdict (values strictly < 0 indicate a falsification), and how accurately can the robustness be reproduced?
5. how accurately can the reported output be reproduced, if given?
6. are the values reported in tables 3 and 4 correct?

In the end, we checked for 1., 3., and 4. Regarding 2., it is not quite straight-forward to check the shape of the signal, so we did not attempt it. Regarding 5., not all participants did report output signals, so we omitted this aspect from further investigation. For 6., there was simply not sufficient consistent data, as many participants chose to run a fraction only of the experiments represent in the tables.

4.1 Comparative Results

The results for unconstrained piecewise-continuous input signals (instance 1) are shown in table 3. For a better comparison of the performance of the tools, a common ground is piecewise constant input signals (instance 2) with a concrete specification of the number of discontinuities allowed. The corresponding results are shown in table 4. Empty cells indicate lack of data for various reasons, such as missing tool support for a particular benchmark feature, or simply that the respective participants did not take the time to set up and/or run these experiments.

The results depend on the choices for the search space, which we briefly discuss for each participating tool as described in section 3. For a more detailed discussion of the outcomes, see [13, 12, 11].

Some results of `FalCAuN` could not be confirmed, because the abstraction it uses is too coarse, which is to some degree expected. Validation for FalStar is not meaningful, since it is based on exactly the same experimental setup. Otherwise, we are happy that all results could be validated. Nevertheless, we emphasize that validation *was* worthwhile, as we did catch cases of mismatching Simulink model versions and minor issues with formalization of requirements.

4.2 Falsification with Probabilistic Guarantees

This year we also propose to introduce tools that can provide probabilistic guarantees for falsification of the system under test to understand if we can provide any conclusion about the system under test along with falsifying it. This becomes an even important question to address when no falsification are found. Providing probabilistic guarantees can help in assessing the safety of the system while also providing the quality of test samples generated.

To that end, we introduce the Part-X [31] algorithm in this competition that can help in providing the probabilistic guarantees of the system under test. Along with the standard results for benchmarks, Part-X also provide results on the lower and upper confidence bounds of normalized falsification volume at 95% confidence. The results are shown in table 2 for both intances. We refrain from an in-depth analysis of these results for now.

Definition 1 (Normalized Falsification Volume)**.** *Let $\{\sigma_k\}$ be the subregions produced by the* Part-X *algorithm up to the k^{th} iteration each with associated surrogate model $\hat{Y}_k^{\gamma}(\mathbf{x}), \mathbf{x} \in \sigma_k$*

Table 2: Results for piecewise continuous input signals (instance 1) and constrained input signals (instance 2). *FR*: falsification rate, ✓: validated falsification rate, $\overline{S}$: mean number of simulations, $\widetilde{S}$: median (rounded down) number of simulations, *LCB*: Lower Confidence Bound at 95% confidence, *UCB*: Upper Confidence Bound at 95% confidence *R*: Non-simulation time ratio (%). Bold entries indicate that some results could not be validated.

Tool	**Ψ-TaLiRo**							**Ψ-TaLiRo**						
Approach	Part-X							Part-X						
Instance	1							2						
Property	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*LCB*	*UCB*	*R*	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*LCB*	*UCB*	*R*
AT1	10	**9**	34.9	28.5	0.00E+00	7.03E-04	70.7	10	10	30.5	25.5	0.00E+00	5.58E-04	85.7
AT2	10	10	6.7	5.5	9.45E-02	1.80E-01	52.8	10	10	6.5	5.0	1.16E-01	2.77E-01	50.6
AT51	0	0	–	–	0.00E+00	0.00E+00	93.9	10	10	13.3	11.5	2.22E-01	5.86E-01	64.3
AT52	10	10	5.6	2.0	1.81E-01	9.02E-01	62.5	10	10	66.5	53.5	0.00E+00	0.00E+00	93.5
AT53	10	10	15.7	15.5	2.45E-02	4.26E-01	59.7	10	10	2.2	2.0	8.38E-01	1.00E+00	57.0
AT54	3	3	862.6	–	0.00E+00	3.60E-05	91.0	10	10	85.0	65.0	0.00E+00	7.68E-02	76.2
AT6a	10	10	134.3	51.5	1.18E-01	2.47E-01	58.2	10	10	153.7	72.0	5.75E-02	1.94E-01	53.1
AT6b	10	10	212.2	150.0	9.45E-02	2.88E-01	57.8	10	10	307.9	111.5	3.40E-02	1.97E-01	56.4
AT6c	10	**9**	200.5	138.0	9.94E-02	2.86E-01	58.1	10	10	334.4	249.5	4.34E-02	1.98E-01	59.3
AT6abc	10	**9**	126.1	50.0	1.02E-01	2.67E-01	68.7	10	10	106.9	67.5	5.72E-02	2.06E-01	69.4
CC1	10	10	19.0	16.5	2.71E-01	8.31E-01	69.2	10	10	17.6	21.0	2.83E-01	8.86E-01	68.7
CC2	10	10	23.9	12.0	4.82E-01	1.00E+00	68.6	10	10	17.8	12.0	2.27E-01	1.00E+00	66.3
CC3	10	**9**	23.1	24.0	1.28E-01	4.58E-01	69.9	10	10	13.5	12.0	1.18E-01	1.00E+00	69.5
CC4	0	0	–	–	0.00E+00	0.00E+00	95.3	0	0	–	–	0.00E+00	0.00E+00	94.5
CC5	10	10	45.8	29.0	3.83E-02	7.10E-01	79.4	10	10	29.9	22.5	2.09E-01	5.90E-01	73.8
CCx	9	9	681.9	703.0	0.00E+00	0.00E+00	96.0	10	10	607.1	156.0	0.00E+00	0.00E+00	96.2
NN	10	10	15.2	16.0	4.84E-01	8.80E-01	83.5	10	10	145.8	89.5	0.00E+00	1.36E-01	87.3
NNx	–	–	–	–	–	–	–	10	10	190.7	40.0	0.00E+00	1.20E-02	66.4
SC	0	–	–	–	0.00E+00	0.00E+00	78.9	0	0	–	–	0.00E+00	2.70E-05	45.5
F16	0	–	–	–	0.00E+00	0.00E+00	39.2	–	–	–	–	–	–	–
AFC27	–	–	–	–	–	–	–	10	0	34.3	27.0	5.90E-01	7.27E-01	89.4
AFC29	–	–	–	–	–	–	–	10	0	12.1	11.0	2.31E-01	5.36E-01	87.9
AFC33	–	–	–	–	–	–	–	0	0	–	–	0.00E+00	0.00E+00	96.1

and V be the total volume of the search space, then the normalized falsification volume calculated for δ_C-quantile is:

$$V^{\delta_C}|(\mathbf{x},\mathbf{y}) = \int_{\bigcup \sigma_k} \frac{I_{q_{\delta_C}<0}}{V}. \tag{1}$$

Table 3: Results for piecewise continuous input signals (instance 1). *FR*: falsification rate, ✓: validated falsification rate, $\overline{S}$: mean number of simulations, $\widetilde{S}$: median (rounded down) number of simulations, *LCB*: Lower Confidence Bound at 95% confidence, *UCB*: Upper Confidence Bound at 95% confidence, *R*: $(\frac{SimulationTime}{TotalTime}) * 100$ (%).

Tool:	**Uniform Random**				**ARIsTEO**				**FalCAuN**				**falsify**				**FalStar**				**ForeSee**				**Ψ-TaLiRo**				
Approach:					ARX-2								A3C				aLVTS								LSemiBO				
Benchmark	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*
AT1	0	0	–	–	0		–	–	10	10	3177.8	3166.5					10	10	26.2	18.5	10	10	337.6	347.5	10	8	106.3	105.5	72.7
AT2	10	10	7.6	5.0	10	10	5.4	2.0	7/7	7/7	1992.0	1442.0					10	10	3.5	2.0	10	10	107.5	33.5	10	10	15.5	12.5	57.5
AT51	1	1	923.0	923.0	0	0	-	-					10	10	4.5	3.0	10	10	65.2	22.0	10	10	19.3	10.5	1	1	22.0	22	92.0
AT52	10	10	4.1	2.0	10	10	1.7	1.5					10	10	1.1	1.0	10	10	252.6	240.5	10	10	65.1	36.5	10	10	3.2	2.5	61.7
AT53	10	10	18.6	15.0	10	10	12.8	8.5					10	10	1.2	1.0	10	10	99.6	81.5	10	10	4.0	2.0	10	10	28.0	21.0	60.2
AT54	4	4	1121.3	1056.0	0	0	-	-					10	10	1.4	1.0	10	10	86.3	52.5	10	10	24.8	23.5	7	5	715.3	554.0	90.6
AT6a	10	10	74.4	41.5	10	10	93.4	70.0	10	**2**	860.0	786.0					10	10	74.4	79.0	10	10	128.0	136.0	10	10	76.6	89.0	58.1
AT6b	10	10	251.3	189.0	4	4	112.3	127	10	**0**	750.9	653.5	3	3	166.3	186.0	10	10	92.0	81.0	10	10	227.9	211.0	7	7	426.1	224.0	58.2
AT6c	10	10	185.2	86.0	10	10	124.2	111.5	10	**0**	1227.1	1014.0	0	0	–	–	10	10	620.0	953.4	10	10	156.9	128.0	9	9	328.2	112.0	53.3
AT6abc	10	10	58.8	33.5	10	10	73.4	30.5									10	10	41.0	601.5					10	10	34.2	29.5	37.0
NN	10	10	38.6	27.5	1	1	299.0	299.0					9	9	19.7	13.0	10	10	185.5	126.0	10	10	51.3	46.0	10	10	36.4	35.5	84.6
NN$_{\beta = 0.04}$													5	5	72.4	74													
NNx					0	0	–	–									5	5	787.0	813.0	10	10	1.1	1.0					
NL CC1	10	10	10.4	9.5	10	10	24.8	17.5	10	10	206.1	214.5	10	10	42.0	28.5	10	10	2.9	1.5					10	10	13.1	8.5	71.5
CC2	10	10	15.4	15.0	10	10	14.6	9.0	10	**0**	119.0	119.0	4	4	43.3	10.0	10	10	3.8	1.5					10	10	16.4	11.0	65.5
CC3	10	10	77.9	54.5	10	10	59.5	35.0	10	**3**	224.0	218.0	10	10	35.0	33.0	10	10	6.2	4.5					10	10	21.5	15.0	71.8
CC4	0	0	–	–	0	0	–	–					0	0	–	–	2	2	1256.5	1256.5					1	1	1253.0	1253.0	93.6
CC5	10	10	28.5	14.5	10	10	18.8	18.0					3	3	5.7	5	10	10	60.5	32.0					10	10	47.3	39.0	84.8
CCx	9	9	667.8	469.0	4	4	134.0	74.0									9	9	962.4	1000.0					10	10	210.6	70.0	20.8
F16	0	0	–	–																									
SC	0	–	–	–	0		–	–																					

Table 4: Results for constrained input signals (instance 2). *FR*: falsification rate, ✓: validated falsification rate, $\overline{S}$: mean number of simulations, $\widetilde{S}$: median (rounded down) number of simulations, *LCB*: Lower Confidence Bound at 95% confidence, *UCB*: Upper Confidence Bound at 95% confidence, *R*: $(\frac{SimulationTime}{TotalTime}) * 100$ (%).

Tool:	**Uniform Random**				**ARIsTEO**				**FalCAuN**				**falsify**				**FalStar**				**ForeSee**				**Ψ-TaLiRo**				
Approach:					ARX-2								A3C				aLVTS								LSemiBO				
Benchmark	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*
AT1	0		–	–	0	0	–	–	10	10	150.0	142.5													10	10	105.4	105.0	86.8
AT2	10	10	18.8	13.5	10	10	12.9	8.0	8/8	8/8	1228.9	1029.5													10	10	11.6	11.0	50.3
AT51	10	10	20.5	16.5	10	10	19.0	10.0					10	10	15.5	15.5									10	10	13.7	8.5	69.3
AT52	10	10	74.1	65.0	9	9	74.7	46.0					10	10	4.7	5.0									10	10	79.1	95.0	92.9
AT53	10	10	1.5	1.0	10	10	1.4	1.0					10	10	2.4	1.5									10	10	2.7	2.0	57.1
AT54	10	10	47.9	42.0	10	10	44.0	40.0					10	10	33.4	29.0									10	10	37.7	32.0	79.6
AT6a	10	10	156.6	138.0	7	7	65.1	62.0																	9	9	255.0	226.0	61.7
AT6b	10	10	472.2	588.0	5	5	91.2	74.0					9	9	107.3	76.0									9	9	580.0	522.0	49.4
AT6c	10	10	326.8	176.0	7	7	175	141.0	10	**0**	239.6	215.0	0	0	–	–									8	8	361.5	168.0	54.7
AT6abc	10	10	149.0	125.5	9	9	84.1	81.0																	10	10	240.5	74.0	7.8
AFC27	10	0	–	–	9	–	21.9	25.0					10	6	1.9	1.0									10	0	113.2	109.5	97.8
AFC29	10	0	25.1	19.0	10	–	3.4	3.5					10	5	1.0	1.0									10	0	19.6	19.0	100.0
AFC33	10	0	–	–	0	–	–	–					10	0	1.0	1.0									0	0	–	–	35.0
NN	10	10	277.2	158.5	10	10	117	82.5													10	10	74.9	88.5	10	10	155.5	100.0	88.8
$NN_{\beta = 0.04}$													10/20	9/20	107.3	76													
NNx	10	10	712.7	488.0	0	0	–	–													10	10	1.0	1.0	10	10	46.8	48.0	37.2
CC1	10	10	16.4	9.5	10	10	9.1	8.0					10	10	19.9	16.5					10	10	27.6	28.5	10	10	10.8	8.0	73.9
CC2	10	10	12.4	13.0	10	10	10.8	9.0					9	9	8.4	7.0					1	1	148.0	148.0	10	10	9.6	7.0	68.3
CC3	10	10	19.6	21.0	10	10	12.8	13.5					10	10	8.2	5.0					10	10	16.4	10.0	10	10	11.7	8.0	70.5
CC4	0	0	–	–	0	0	–	–					1	0	15.5	15.0					10	**9**	586.3	575.0	3	**0**	1608.0	1580.0	96.9
CC5	10	10	37.4	22.0	10	10	21.1	11.5					4	4	16.5	16.0					10	10	95.2	30.0	10	10	28.3	27.0	75.3
CCx	7	7	614.0	291.0	3	3	97	103.0																	10	10	240.5	74.0	35.4
SC	0	0	–	–	100.0	0		–																					

5 Conclusion and Outlook

The benchmark set established by this competition appears to gain traction in the falsification literature, underpinning the results and experiments that are being published. Publishing such results as part of the competition report sets a reference and gold-standard against which such efforts can compare. It is therefore great to again have new participants this year.

Clearly, the approaches implemented in the participating tools have become even more diverse, and a comparison on even footing and for a full set of results becomes more difficult. Nevertheless, the results reported in tables 3 and 4 already give a good overview of the strengths and weaknesses of the respective approaches. It has become apparent that no "best" approach can be singled out.

The effort to validate the falsifying inputs generated by the tools proved to be more involved than initially thought. A non-negligible part of that effort was to establish the technical basis in terms of exchange formats and a validator tool. There are plenty of sources of error—by the human and computational—which each of its own warrants dedicated investigation. While many big and systematic issues could be resolved, some more subtle ones remain.

Overall, our findings stress the importance of validating experimental data, especially in a well-defined comparative setting. This experience is shared with other competitions like SV-COMP (which has validation since 2016 [5]), Test-Comp (which had independent coverage evaluation from the start in 2019 [6]), and many other competitions (for an overview, see [3]).

Several issues remain: The goal of a larger benchmark set with more complex and higher-dimensional models is still not achieved. Moreover, there is still no end-to-end process for obtaining results and validating these in a more automated way, such that the entire competition can—at least in principle—be repeated independently.

The competition on Search-Based Software Testing [18] (SBST) has a Cyber-physical systems track with a lane-keeping assistant benchmark. Overall, six tools have participated in the SBST competition. We will aim at an exchange of ideas and lessons learned to bring the two currently disjoint communities closer together.

Finally, we aim at facilitating the process for new participants to enter the friendly ARCH-COMP on falsification. To that end, it would be great to have a guide that outlines the steps necessary and where respective information can be found.

Acknowledgments Many thanks to the organizers of the ARCH workshop 2022 for hosting this competition and for providing a supportive and friendly environment. The organizer thanks all participants for their time and patience during investigation of the discrepancies found during validation. P. Arcaini is supported by MIRAI Engineerable AI Project (No. JPMJMI20B8), JST; and ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST, Funding Reference number: 10.13039/501100009024. The ASU team (S-TaLiRo + SOAR) was partially supported by DARPA FA8750-20-C-0507, NSF CNS 1932068, NSF CMMI 2046588, NSF CNS 2000792, NSF CMMI 1829238, NSF IIP-1361926 and the NSF I/UCRC Center for Embedded Systems. Z. Zhang is supported by JSPS KAKENHI Grant No. 20H04168, 19K24348, 19H04086, JST-Mirai Program Grant No. JPMJMI20B8, Japan. F. Formica, M. M. Mahboob, and C. Menghi are supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number RGPIN-2022-04622]. M. Waga is partially supported by JST ACT-X Grant Number JPMJAX200U and JST CREST Grant Number JPMJCR2012, Japan.

References

[1] Yashwanth Annpureddy, Che Liu, Georgios Fainekos, and Sriram Sankaranarayanan. S-TaLiRo: A tool for temporal logic falsification for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 254–257. Springer, 2011.

[2] Anne Auger and Nikolaus Hansen. A restart CMA evolution strategy with increasing population size. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005*, pages 1769–1776, 2005.

[3] Ezio Bartocci, Dirk Beyer, Paul E Black, Grigory Fedyukovich, Hubert Garavel, Arnd Hartmanns, Marieke Huisman, Fabrice Kordon, Julian Nagele, Mihaela Sighireanu, et al. Toolympics 2019: An overview of competitions in formal methods. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 3–24. Springer, 2019.

[4] Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donzé, Georgios Fainekos, Oded Maler, Dejan Ničković, and Sriram Sankaranarayanan. Specification-based monitoring of cyber-physical systems: a survey on theory, tools and applications. In *Lectures on Runtime Verification*, pages 135–175. Springer, 2018.

[5] Dirk Beyer. Reliable and reproducible competition results with benchexec and witnesses (report on SV-COMP 2016). In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 887–904. Springer, 2016.

[6] Dirk Beyer. International competition on software testing (Test-Comp). In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 167–175. Springer, 2019.

[7] Anthony Corso, Robert J Moss, Mark Koren, Ritchie Lee, and Mykel J Kochenderfer. A survey of algorithms for black-box safety validation. *arXiv preprint arXiv:2005.02979*, 2020.

[8] Adel Dokhanchi, Shakiba Yaghoubi, Bardh Hoxha, and Georgios Fainekos. ARCH-COMP17 category report: Preliminary results on the falsification benchmarks. In Goran Frehse and Matthias Althoff, editors, *ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 48 of *EPiC Series in Computing*, pages 170–174. EasyChair, 2017.

[9] Adel Dokhanchi, Shakiba Yaghoubi, Bardh Hoxha, Georgios Fainekos, Gidon Ernst, Zhenya Zhang, Paolo Arcaini, Ichiro Hasuo, and Sean Sedwards. ARCH-COMP18 category report: Results on the falsification benchmarks. In Goran Frehse, editor, *ARCH18. 5th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 54 of *EPiC Series in Computing*, pages 104–109. EasyChair, 2018.

[10] Johan Liden Eddeland, Alexandre Donze, Sajed Miremadi, and Knut Akesson. Industrial temporal logic specifications for falsification of cyber-physical systems. In *ARCH@CPSIoTWeek*, 2020.

[11] Gidon Ernst, Paolo Arcaini, Ismail Bennani, Aniruddh Chandratre, Alexandre Donzé, Georgios Fainekos, Goran Frehse, Khouloud Gaaloul, Jun Inoue, Tanmay Khandait, Logan Mathesen, Claudio Menghi, Giulia Pedrielli, Marc Pouzet, Masaki Waga, Shakiba Yaghoubi, Yoriyuki Yamagata, and Zhenya Zhang. ARCH-COMP 2021 category report: Falsification with validation of results. In Goran Frehse and Matthias Althoff, editors, *8th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21)*, volume 80 of *EPiC Series in Computing*, pages 133–152. EasyChair, 2021.

[12] Gidon Ernst, Paolo Arcaini, Ismail Bennani, Alexandre Donzé, Georgios Fainekos, Goran Frehse, Logan Mathesen, Claudio Menghi, Giulia Pedrielli, Marc Pouzet, Shakiba Yaghoubi, Yoriyuki Yamagata, and Zhenya Zhang. ARCH-COMP 2020 category report: Falsification. 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 140–152. EasyChair, 2020.

[13] 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 Goran Frehse and Matthias Althoff, editors, *ARCH19. 6th International*

Workshop on Applied Verification of Continuous and Hybrid Systems, volume 61 of *EPiC Series in Computing*, pages 129–140. EasyChair, 2019.

[14] Gidon Ernst, Sean Sedwards, Zhenya Zhang, and Ichiro Hasuo. Fast falsification of hybrid systems using probabilistically adaptive input. *arXiv preprint arXiv:1812.04159*, 2018.

[15] Gidon Ernst, Sean Sedwards, Zhenya Zhang, and Ichiro Hasuo. Falsification of hybrid systems using adaptive probabilistic search. *ACM Trans. Model. Comput. Simul.*, 31(3), jul 2021.

[16] Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications. In Klaus Havelund, Manuel Núñez, Grigore Roşu, and Burkhart Wolff, editors, *Formal Approaches to Software Testing and Runtime Verification*, LNCS, pages 178–192. Springer, 2006.

[17] Martin Fränzle and Michael R Hansen. A robust interpretation of duration calculus. In *International Colloquium on Theoretical Aspects of Computing*, pages 257–271. Springer, 2005.

[18] Alessio Gambi, Gunel Jahangirova, Vincenzo Riccio, and Fiorella Zampetti. Sbst tool competition 2022. In *2022 IEEE/ACM 15th International Workshop on Search-Based Software Testing (SBST)*, pages 25–32. IEEE, 2022.

[19] Peter Heidlauf, Alexander Collins, Michael Bolender, and Stanley Bak. Verification challenges in F-16 ground collision avoidance and other automated maneuvers. In Goran Frehse, editor, *ARCH18. 5th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 54 of *EPiC Series in Computing*, pages 208–217. EasyChair, 2018.

[20] Bardh Hoxha, Houssam Abbas, and Georgios Fainekos. Benchmarks for temporal logic requirements for automotive systems. In Goran Frehse and Matthias Althoff, editors, *ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*, volume 34 of *EPiC Series in Computing*, pages 25–30. EasyChair, 2015.

[21] Jianghai Hu, John Lygeros, and Shankar Sastry. Towards a theory of stochastic hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 160–173. Springer, 2000.

[22] Xiaoqing Jin, Jyotirmoy V. Deshmukh, James Kapinski, Koichi Ueda, and Ken Butts. Powertrain control verification benchmark. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, HSCC '14, pages 253–262, New York, NY, USA, 2014. ACM.

[23] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, Nov 1990.

[24] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In Yassine Lakhnech and Sergio Yovine, editors, *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[25] Logan Mathesen. *Making Bayesian Optimization Practical in the Context of High Dimensional, Highly Expensive, Black-Box Functions.* PhD thesis, Arizona State University, 2021.

[26] Logan Mathesen, Giulia Pedrielli, and Georgios Fainekos. Efficient optimization-based falsification of cyber-physical systems with multiple conjunctive requirements. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 732–737, 2021.

[27] Logan Mathesen, Giulia Pedrielli, Szu Hui Ng, and Zelda B Zabinsky. Stochastic optimization with adaptive restart: A framework for integrated local and global learning. *Journal of Global Optimization*, 79(1):87–110, 2021.

[28] Claudio Menghi, Shiva Nejati, Lionel Briand, and Yago Isasi Parache. Approximation-refinement testing of compute-intensive cyber-physical models: An approach based on system identification. In *International Conference on Software Engineering (ICSE)*. IEEE / ACM, 2020.

[29] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pages 1928–1937, 2016.

[30] Dejan Ničković and Tomoya Yamaguchi. Rtamt: Online robustness monitors from stl. In *Automated Technology for Verification and Analysis: 18th International Symposium, ATVA 2020,*

Hanoi, Vietnam, October 19–23, 2020, Proceedings, page 564–571, Berlin, Heidelberg, 2020. Springer-Verlag.

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

[32] Doron A. Peled, Moshe Y. Vardi, and Mihalis Yannakakis. Black box checking. In Jianping Wu, Samuel T. Chanson, and Qiang Gao, editors, *Formal Methods for Protocol Engineering and Distributed Systems, FORTE XII / PSTV XIX'99, IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XII) and Protocol Specification, Testing and Verification (PSTV XIX), October 5-8, 1999, Beijing, China*, volume 156 of *IFIP Conference Proceedings*, pages 225–240. Kluwer, 1999.

[33] Quinn Thibeault, Jacob Anderson, Aniruddh Chandratre, Giulia Pedrielli, and Georgios Fainekos. Psy-taliro: A python toolbox for search-based test generation for cyber-physical systems. In Alberto Lluch Lafuente and Anastasia Mavridou, editors, *Formal Methods for Industrial Critical Systems*, pages 223–231, Cham, 2021. Springer International Publishing.

[34] Masaki Waga. Falsification of cyber-physical systems with robustness-guided black-box checking. In Aaron D. Ames, Sanjit A. Seshia, and Jyotirmoy Deshmukh, editors, *HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, April 21-24, 2020*, pages 11:1–11:13. ACM, 2020.

[35] Shakiba Yaghoubi and Georgios Fainekos. Gray-box adversarial testing for control systems with machine learning components. In *International Conference on Hybrid Systems: Computation and Control (HSCC)*, 2019.

[36] Yoriyuki Yamagata, Shuang Liu, Takumi Akazaki, Yihai Duan, and Jianye Hao. Falsification of cyber-physical systems using deep reinforcement learning. *IEEE Transactions on Software Engineering*, 47(12):2823–2840, 2021.

[37] Zhenya Zhang, Deyun Lyu, Paolo Arcaini, Lei Ma, Ichiro Hasuo, and Jianjun Zhao. Effective hybrid system falsification using monte carlo tree search guided by QB-robustness. In Alexandra Silva and K. Rustan M. Leino, editors, *Computer Aided Verification*, pages 595–618, Cham, 2021. Springer International Publishing.