ARCH-COMP 2023 Category Report: Falsification

Menghi, Claudio; Arcaini, Paolo; Baptista, Walstan; Ernst, Gidon; Fainekos, Georgios; Formica, Federico; Gon, Sauvik; Khandait, Tanmay; Kundu, Atanu; Pedrielli, Giulia; Peltomäki, Jarkko; Porres, Ivan; Ray, Rajarshi; Waga, Masaki; Zhang, Zhenya

Published in:
Proceedings of 10th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH23)

DOI:
10.29007/6nqs

Published: 01/01/2023

Document Version
Final published version

Document License
Unknown

EPiC Series in Computing

Volume 96, 2023, Pages 151–169

Proceedings of 10th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH23)

ARCH-COMP 2023 Category Report: Falsification*

Claudio Menghi[1,2], Paolo Arcaini[3], Walstan Baptista[7], Gidon Ernst[4], Georgios Fainekos[5], Federico Formica[2], Sauvik Gon[6], Tanmay Khandait[7], Atanu Kundu[6], Giulia Pedrielli[7], Jarkko Peltomäki[8], Ivan Porres[8], Rajarshi Ray[6], Masaki Waga[9], and Zhenya Zhang[10]

[1] University of Bergamo, Bergamo, Italy claudio.menghi@unibg.it
[2] McMaster University, Hamilton, Canada {menghic,formicaf}@mcmaster.ca
[3] National Institute of Informatics (NII), Tokyo, Japan arcaini@nii.ac.jp
[4] Ludwig-Maximilians-University (LMU), Munich, Germany gidon.ernst@lmu.de
[5] Toyota Motor North America, Research & Development georgios.fainekos@toyota.com
[6] Indian Association for the Cultivation of Science, Kolkata, India {ugsg2584,mcsak2346,rajarshi.ray}@iacs.res.in
[7] Arizona State University (ASU), Tempe, USA {wbaptist,tkhandai,gpedriel}@asu.edu
[8] Åbo Akademi University, Turku, Finland {jarkko.peltomaki,ivan.porres}@abo.fi
[9] Kyoto University, Japan mwaga@fos.kuis.kyoto-u.ac.jp
[10] Kyushu University, Japan zhang@ait.kyushu-u.ac.jp

Abstract

This report presents the results from the 2023 friendly competition in the ARCH workshop for the falsification of temporal logic specifications over Cyber-Physical Systems. We describe the benchmark models selected to compare the tools and the competition settings and provide background on the participating teams and tools. Finally, we present and discuss our results.

Data: https://gitlab.com/goranf/ARCH-COMP, https://dx.doi.org/10.5281/zenodo.8024426

1 Introduction

We report on one category of the friendly competition associated with the ARCH 2023 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. This report concerns the *falsification category*, which targets the analysis of executable models with respect to requirements expressed in temporal logic with time bounds, encoded in *Metric Temporal Logic (MTL)* [24] or *Signal Temporal Logic (STL)* [25]. The task is to search for

*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.

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 [17, 19] 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 [5, 8]. For past instalments of this competition 2017–2022 see [9, 10, 14, 13, 12, 15]. The benchmark set developed by this competition series can be seen as a baseline for research in the area (cf. [11]), and we encourage authors to compare to the results presented here.

The competition of 2023 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 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:

- We added a Pacemaker benchmark model to the competition [3]. This benchmark model provides a practical example from the medical domain, which the other models considered in the competition do not cover.
- We had three new tools participating in the competition: ATheNA [18], NNFal [28], and STGEM [34] (see Section 3.6). The tools falsify [38] and FalStar [16] decided to not join the competition this year.
- We reintroduced a maximal number of simulations that can be run in one falsification trial. Unlike previous versions of the competition that set a maximal number of simulations (in 2021 the maximum number was set to 300), we set the maximum number of simulations to the value 1500 to obtain a more accurate comparison of the tools for difficult benchmarks.
- We report the ratio of the the time taken by simulation in a falsification trial and the total time. The aim is to understand the cost of different falsification algorithms and implementations.

This report is structured as follows. Section 2 introduces our benchmark models and requirements. Section 3 describes the tools participating in the competition. Section 4 presents the results obtained by the different tools. Section 5 presents the results obtained by tools that can produce probabilistic guarantees for falsification. Finally, Section 6 concludes the report with our reflections.

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 `2023/FALS`. An archive containing the traces submitted for validation is available at `https://dx.doi.org/10.5281/zenodo.8024426`. This archive contains the result of validation and instructions to re-validate the results.

2 Benchmark

This section presents our benchmarks. First, we present two parametrizations of the input domain considered in this edition of the competition to generate the input signals for the models (Section 2.1). Then, we present the benchmark models and requirements (Section 2.2).

2.1 Input Parameterization

The participants have to consider two parametrizations to generate inputs signals for their models: (a) arbitrary piecewise continuous input signals and (b) constrained input signals.

Arbitrary piecewise 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 piecewise continuous input signals (i.e., discontinuities are permitted), where the values for each individual dimensions are from a given range. 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 additional knowledge about the 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 a possible downside it 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

We provide a short textual description of our benchmark models. Table 1 reports the corresponding requirements formalized as STL/MTL formulas.

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 [21].

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 [21] where the parameters have been chosen to be somewhat difficult.

Fuel Control of an Automotive Powertrain (AFC). The model is described in [23]. The values used in the requirements are chosen to make falsification possible but reasonably hard.

Input specification: 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 feedback and feedforward control), and the engine speed ω to be constant with $900 \leq \omega < 1100$ to capture the input profile outlined in [23]. As in previous

edition of the competition, 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 requirement ensures that after changes to the reference, the actual position eventually stabilizes around that value with small error.

Input specification: 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.

Chasing cars (CC). The model is derived from Hu et al. [22] 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.

Input specification: 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 [20]. The F16 aircraft and its inner-loop controller for Ground Collision avoidance have been modeled using 16 continuous variables with piecewise 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 $[0.2\pi, 0.2833\pi] \times [-0.4\pi, -0.35\pi] \times [-0.375\pi, -0.125\pi]$.

Input specification: 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 [37]. 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 $[3.99, 4.01]$.

Input specification: For instance 2, the input signal should be piecewise constant with 20 evenly spaced segments.

Pacemaker (PM). This model represents a simple controller of a pacemaker device [3]. The controller artificially stimulates the heart muscle to contract when no natural activity is present for a given time. The input of the system is the desired lower rate limit that can vary in the range $[50, 90]$.

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

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	$\square_{[0,30]}((\neg g1 \wedge \circ\, g1) \rightarrow \circ\, \square_{[0,2.5]} g1)$	where $\circ\, \phi \equiv \diamondsuit_{[0.001,0.1]}\ \phi$
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	conjunctive 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 $\beta = 0.008$, $\gamma = 0.007$ $\mathit{rise} = (\theta < 8.8) \wedge (\diamondsuit_{[0,0.05]}(\theta > 40.0))$ $\mathit{fall} = (\theta > 40.0) \wedge (\diamondsuit_{[0,0.05]}(\theta < 8.8))$	
NN	$\square_{[1,37]}(\lvert \mathit{Pos} - \mathit{Ref}\rvert > \alpha + \beta \lvert \mathit{Ref}\rvert \rightarrow \diamondsuit_{[0,2]} \square_{[0,1]} \neg(\alpha + \beta \lvert \mathit{Ref}\rvert \leq \lvert \mathit{Pos} - \mathit{Ref}\rvert))$ where $\alpha = 0.005$ and $\beta = 0.03$	
NNx	$\diamondsuit_{[0,1]}(\mathit{Pos} > 3.2) \wedge \diamondsuit_{[1,1.5]}(\square_{[0,0.5]}(1.75 < \mathit{Pos} < 2.25)) \wedge \square_{[2,3]}(1.825 < \mathit{Pos} < 2.175)$	conjunctive requiremet $1.95 \leq \mathit{Ref} \leq 2.05$
CC1	$\square_{[0,100]} y_5 - y_4 \leq 40$	
CC2	$\square_{[0,70]} \diamondsuit_{[0,30]} y_5 - y_4 \geq 15$	
CC3	$\square_{[0,80]}((\square_{[0,20]} y_2 - y_1 \leq 20) \vee (\diamondsuit_{[0,20]} y_5 - y_4 \geq 40))$	
CC4	$\square_{[0,65]} \diamondsuit_{[0,30]} \square_{[0,20]} y_5 - y_4 \geq 8$	
CC5	$\square_{[0,72]} \diamondsuit_{[0,8]}((\square_{[0,5]} y_2 - y_1 \geq 9) \rightarrow (\square_{[5,20]} y_5 - y_4 \geq 9))$	
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	$\square_{[30,35]}(87 \leq \mathit{pressure} \wedge \mathit{pressure} \leq 87.5)$	
PM	$\square_{[0,10]}(\mathit{paceCount} \leq 15) \wedge \diamondsuit_{[0,10]}(\mathit{paceCount} \geq 8)$	

Input specification: For instance 2, the input signal should be piecewise constant with 5 evenly spaced segments.

3 Participants

We present 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.

3.1 ARIsTEO

Description. ARIsTEO [26] is a Matlab toolbox for test case generation against system specifications 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). [2]

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 $nk = 2$[3] 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 1500.

3.2 ATheNA

Description. ATheNA [18] is a Matlab toolbox for automatic test case generation guided by a combination of automatic and manual fitness functions. ATheNA allows the user to specify a manually-defined fitness function and choose a strategy to combine the automatic and manual fitness values into the ATheNA fitness value. The manually-defined fitness functions can designed by the engineer and can consider both the inputs and the outputs of the model. ATheNA employs S-TaLiRo to compute the automatic fitness function, starting from the MTL/STL specification. The model inputs are generated by an optimization algorithm that tries to lower the value of the ATheNA fitness. ATheNA enables the engineer to focus the exploration of the input space on areas that are considered particularly critical and to switch between different types of fitness functions depending on the situation.

Setup. ATheNA has the same interface of S-TaLiRo, but requires additional information on the manual fitness function and the ATheNA fitness function. We defined a manual fitness function for each model and requirement by reverse-engineering the model. The ATheNA fitness has been calculated as the weighted average of the automatic and manual values. The weight of the two values depends on our confidence in the capabilities of the functions to lead to the identification of a fault. The manual and ATheNA fitness functions used in Instance 1 and Instance 2 are the same. The search algorithm used is Simulated Annealing and the maximum number of iterations for the search process has been set to 1500.

[2] https://github.com/SNTSVV/ARIsTEO
[3] https://nl.mathworks.com/help/ident/ref/arx.html

3.3 FalCAuN

Description. FalCAuN [36] is an experimental tool for testing a Simulink model using black-box checking [30], 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[4].

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.

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 $y_2 - y_1$.

3.4 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 [39] (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

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

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 [14, 13]. It is publicly available under GNU General Public License (GPL) v3.[5]

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.

3.5 NNFal

Description. NNFal is a surrogate model-based falsification framework for CPS. The framework treats CPS as a black box and only assumes that the system to be falsified can be simulated/executed. The foremost step in the framework is building a surrogate model from the simulated trajectories of the CPS. In NNFal, we use a feed-forward neural network as a surrogate model to leverage the adversarial attack algorithms targeted toward the robustness evaluation of neural networks. The safety property is examined in the surrogate model to find a counterexample using a deep neural network falsifier. The counterexample generated by the framework is the initial system configuration along with the piecewise constant input signal that drives the CPS to a safety-violating state. Since the surrogate model is an approximation of the CPS, the generated counterexample on the surrogate may be spurious. The last step of our framework is therefore validating the counterexample in the actual CPS. If the counterexample is found to be spurious, necessary constraints are added in the property specification to eliminate the spurious counterexample from the state space and search for a new counterexample for further investigation. NNFal is publicly available.[6]

Setup. The current implementation of NNFal uses pre-trained fully connected Feed-forward Neural Network (FNN), which we build from the simulation traces of the CPS. The FNN is built using the Keras API, a high-level API of TensorFlow. NNFal supports a portfolio of robustness and reachability property falsifiers for finding counterexamples from the neural network. In the experiments, we have seen that the robustness property falsifier outperforms the reachability property falsifiers, which is why we consider the robustness falsifier as the default in our tool. We specifically employed DNNF, a falsification method for the robustness properties of deep neural networks. DNNF offers options for executing various adversarial attack algorithms, in

[5] https://github.com/choshina/ForeSee
[6] https://gitlab.com/Atanukundu/NNFal

which we use the Projected Gradient Descent (PGD) attack algorithm among them. It has the advantage of random initialization to find an adversarial example. As a result, it can generate varied counterexamples across multiple executions. The current version of NNFal does not support all the property specifications presented in STL. In the current version, we are able to falsify the specifications AT1, AT6c, and CC1, all for constrained input signals (instance 2). Note that the dataset and model learning is a one-time effort. The number of simulations taken in generating the dataset is not included in the results table. Also, the time taken in learning the FNN model from the dataset is also not included in the results.

We target only the instance 2, so the values for NNFal in Table 2 are simply copies of those in Table 3.

3.6 STGEM

Description. STGEM[7] is an open-source toolbox for the falsification of Cyber-Physical Systems. It is written in Python and is currently in active development. STGEM supports the falsification of an arbitrary requirement for any system under test as long as the system (with vector or signal inputs and outputs) and the requirement robustness monitor are implemented as Python classes. Out of the box, STGEM supports Matlab Simulink models as systems under test and provides a monitor for requirements specified in STL. The user can implement a falsification algorithm as a part of STGEM or use the existing falsification algorithms such as OGAN [33] or WOGAN [32].

For the results presented in this report, we have chosen to use the OGAN algorithm for falsification. It is designed for finding a single falsification whereas WOGAN is designed to find multiple different faults. Briefly explained, OGAN trains two neural networks: a generator and a discriminator. The discriminator is trained to learn the mapping from inputs to robustness values. In the training of the generator a specific loss function is used to encourage the generator to turn noise into inputs that the discriminator estimates to have low robustness. The trained generator is sampled for an input, and the selected input is evaluated on the actual system under test. This evaluation results in more training data for the training of the neural networks; an initial training data is obtained via a random search. OGAN in STGEM utilizes in its search the usual STL robustness objective [25]. We emphasize that all training is done from scratch: OGAN does not use any precollected data or pretrained models.

Setup. The total simulation budget of 1500 simulations is large. OGAN needs an initial random search in order to perform well, but fixing it to be a certain proportion of 1500 is problematic. On one hand, when a large budget is available, a more exhaustive random search can help OGAN to learn in hard problems. On the other hand, a long random search can be excessive with easier problems, and it is possible that more simulations are done than is necessary. To alleviate this problem, we perform the random search as follows. Initially 75 inputs are generated based on a Latin hypercube design. The system is simulated with these inputs, and an initial training data is collected. After this, we either use OGAN to generate the next simulation input or obtain the input by sampling the input space uniformly randomly. The decision is done randomly in such a way that during 1500 simulations, we simulate on average 375 random inputs (25% of 1500).

For the conjunctive requirements, we use the multi-armed bandit approach described in [31] meaning that for N requirements we use N OGAN algorithms, one per requirement, but only train and utilize one of them at each step.

[7] https://gitlab.abo.fi/stc/stgem.

Since OGAN utilizes neural networks, there are many hyperparameters to choose. Not only is it necessary to decide how many layers and nodes do the neural networks contain, but learning rates and optimizers need to be set as well. The full description of the parameters is available at https://gitlab.abo.fi/stc/experiments/arch-comp-23. We remark that we have done a partial hyperparameter optimization to obtain a good overall performance on all benchmarks, but we have not focused on any particular benchmark, and we use the same settings in all benchmarks.

We target only the instance 2, so the values for STGEM in Table 2 are simply copies of those in Table 3. Some validation results are omitted since we were unable to obtain them due to technical problems with the validation tool.

3.7 Ψ-TaLiRo

Description. Ψ-TaLiRo [35], the python version of S-TaLiRo [1], 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 new optimizers. The tool box is publicly available on-line under General Public License (GPL).[8] For this competition, we provide results with two different optimization algorithms:

1. **Conjunctive Bayesian Optimization (ConBO)** models 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, and can produce probabilistic guarantees on the presence of falsifying behaviors. 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 of the specification robustness, 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 [29].

Setup. In Ψ-TaLiRo, input signals to black-box models are parameterized with control points and their corresponding time stamps (for interpolation), which then leads to the formation of an optimization problem with dimensionality depending upon the number of control points. The input signals along with their corresponding time stamps are interpolated depending on the benchmark problem instance. For this competition, all signals have evenly spaced control points and are interpolated using the `pchip` function for instance 1, and a piecewise constant interpolation function for instance 2. We utilize RTAMT [27] for robustness calculation.

The ConBO optimizer samples 20 points from the search space and then sequentially samples 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

[8] https://gitlab.com/sbtg/psy-taliro

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 4.

4 Evaluation & Validation

We present the experimental setup (Section 4.1) and the results of our experiment (Section 4.2)

4.1 Setup

The tools participating in the competition were instructed to run the falsification of each individual requirement 10 times, to account for the stochastic nature of most algorithms. The cut-off for the number of simulations imposed on the experiments was 1500. This value enables a more accurate comparison of the tools for difficult benchmarks. The results were provided by the participants and have been obtained on multiple platforms with varying resources and different MATLAB/Simulink versions.

The participants have to report information related to each falsification trial per requirement, according to the reporting format available at `https://gitlab.com/gernst/ARCH-COMP/-/blob/FALS/2021/FALS/Validation.md`. The information includes:

- the 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 this information as the "reported" result.

For each tool, we compute 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). Finally, to estimate the time spent on the search algorithm versus the time required to simulate the model, we computed the ratio between the simulation time and the total falsification time.

We continue the effort to validate 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. We validated the following

- the reported input signal adhere to the valid ranges of input for that particular model;
- the correctness of the reported verdict;
- the consistency of the reported robustness value and the verdict.

The results reported by the participants are presented in the following.

4.2 Results

Table 2 and Table 3 respectively report the results for instances 1 and 2 for each of participant. The tables also report the results obtained using a Uniform Random (UR) testing strategy; that is no optimization strategy is used in the search. For each instance, the tables report the falsification rate (FR), validated falsification rate (✓), mean number of simulations ($\overline{S}$), median (rounded down) number of simulations ($\widetilde{S}$). Empty cells indicate a lack of data for a particular benchmark due to lack of support or simply that the respective participants did not take the time to set up and/or run these experiments. For example, NNb, NNx, F16, and SC were not assessed for UR.

Significant effort was spent in validating the results provided by the different participants. Two of the authors (i.e., Gidon Ernst and Tanmay Khandait) prepared a validation platform that the participants used to assess the correctness of the results provided by the participants.

For some of the tools, e.g., `FalCAuN`, the results were not confirmed by the validation platform due to differences between the configuration of the validation platform and the platform used to run the tool. For example, some results of `FalCAuN` could not be confirmed due to the use of the discrete-time semantics of STL, which was, to some degree, expected. Before evaluating STL formulas, `FalCAuN` discretizes the observed signals to interpret them as strings to apply standard automata-based techniques. However, this approach overlooks the behavior between observed points. As a result, the validation fails for STL formulas, for example, of the form $\Diamond\varphi$. For these cases, the value reported by the validated falsification rate (✓) column is lower than that reported by the falsification rate (FR) column.

For some tools, the participants found technical problems running the validation for some benchmarks, or the validation platform does not support the benchmark. These problems are reasonable since the validation tool is in early development, and some participants used it for the first time. For these cases, the participants reported the symbol "–" in the validated falsification rate (✓) column. For example, the validation platform currently does not support the validation of the pacemaker (PM) benchmark. We plan to address these limitations in the next edition of the competition.

Table 2: 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, *R*: $(\frac{SimulationTime}{TotalTime}) * 100$ (%).

Tool:	**UR**					**ARIsTEO**					**ATheNA**					**FalCAuN**					**ForeSee**					**NNFal**					**Ψ-TaLiRo**					**STGEM**				
Approach:						ARX-2																									ConBO					OGAN				
Benchmark	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*
AT1	0	0	–	–	100.0	0	0	–	–	53.0	4	4	532	576.8	61.9	4	4	1029.5	1082	95.1	10	–	382.4	354	98.4	2	2	1.5	1.5	99.5	10	10	106.3	105.5	73.3	10	10	185.8	153	29.2
AT2	10	10	7.6	5.0	99.9	10	10	2	5.4	22.3	10	10	36	51.4	78.6	6	6	501.5	374	93.5	10	–	15.5	7	96.5						10	10	15.5	12.5	54.7	10	10	15.3	11	100.0
AT51	1	1	923.0	923.0	100.0	0	0	–	–		10	10	192	274.3	50.3																1	1	22.0	22.0	91.6	10	10	15	13	100.0
AT52	10	10	4.1	2.0	99.9	10	10	1	2.4		10	10	31	36.8	58.5						10	–	6.1	2	92.1						10	10	3.2	2.5	63.4	10	10	6.3	3.5	100.0
AT53	10	10	18.6	15.0	100.0	10	10	8	12.8		10	10	70	74.9	58.2						10	–	5.1	3.0	91.2						10	10	28.0	21.0	60.9	10	10	2	2	100.0
AT54	3	3	932.0	868.0	100.0	5	5	101	254.0		10	10	242	288.3	53.4																7	7	715.3	544.0	89.9	10	10	18.4	12	100.0
AT6a	10	10	74.4	41.5	100.0	10	10	70	93.4	54.0	10	10	90	96.4	56.6	10	4	457.9	316	90.7	10	–	80.3	60.5	97.5						10	10	76.6	89.0	60.6	10	10	74.2	69.5	60.4
AT6b	10	10	251.3	189.0	100.0	9	9	141	268.0	44.7	10	10	86	120.5	53.1	10	0	600.0	566	91.6	10	–	152.8	154	97.8						6	6	185.5	152.0	58.8	10	10	128.3	129	38.3
AT6c	10	10	185.2	86.0	100.0	10	10	122	154.3	45.7	10	10	79	90.6	46.4	10	0	901.2	862	89.9	10	–	179.9	169	97.7	1	1	3	3	99.62	8	8	166.6	101.0	60.2	10	10	89.6	88.5	66.4
AT6abc	10	10	58.8	33.5	100.0	10	10	122	154.3	25.7	10	10	99	142.7	26.7						10	–	121.6	92	96.0						10	10	34.2	29.5	37.0	10	10	63.8	67.5	55.4
NN	10	–	38.6	27.5	100.0	8	–	81	341.9	1.5	10	–	167	215.3	35.9						10	–	308.7	341	98.0						10	–	36.4	35.5	84.1	10	–	120.4	135.5	50.8
$NN_{\beta=0.04}$						0	0	–	–	2.4	0	0	–	–	36.4																						7	–	607.7	482	28.6
NNx						0	0	–	–	2.5	0	0	–	–	72.7						10	–	1.4	1.0	85.7											0	–	–	–	23.9
CC1	10	10	10.4	9.5	100	10	10	10.4	9.5	100.0	10	10	42	54.7	86.4	10	10	229.2	214	98.6	10	–	34.9	37	99.1	9	9	1.22	1	99.68	10	10	13.1	8.5	69.9	10	10	33.3	21	88.0
CC2	10	10	15.4	15.0	100.0	10	10	9	13.8	66.9	10	10	44	72.1	81.6	10	10	119.0	119	97.1	3	–	1013.7	1351	99.3						10	10	16.4	11.0	69.3	10	10	65.9	66	92.1
CC3	10	10	77.9	54.5	100.0	10	10	37	60.0	58.6	10	10	84	101.7	76.2	10	7	195.2	178	98.3	10	–	64.9	63	98.8						10	10	21.5	15.0	72.3	10	10	20.3	24	100.0
CC4	0	0	–	–	100.0	0	0	–	–	59.9	2	2	764	764.0	79.9						10	–	581.7	603	99.2						1	1	1253.0	1253.0	93.6	0	–	–	–	18.5
CC5	10	10	28.5	14.5	100.0	10	10	18	19.7	75.5	10	10	68	75.6	80.8						10	–	101.0	76.0	98.9						10	10	47.3	39.0	80.8	10	–	43.7	46.5	95.0
CCx	7	7	338.1	300.0	100.0	8	8	820	846.4	40.4	10	10	173	174.5	65.6						10	–	150.2	120.5	98.5						10	10	210.6	70.0	20.8	6	6	635.3	534	19.9
F16											10	–	145	151.1	99.4																					7	–	612.1	522	2.6
SC						0	0	–	–	53.9	0	–	–	–	95.8																						0	–	–	–	7.1
PM	8	–	571.8	443.5	100.0	6	–	643	614.5	17.1	3	–	151	239.0	10.8	10	–	34.0	34	84.6											10	–	101.0	102.5	91.5	10	–	245.1	166.5	19.3

Table 3: 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.

Tool:	**UR**					**ARIsTEO**					**ATheNA**					**FalCAuN**				
Approach:						ARX-2														
Benchmark	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*
AT1	0	0	–	–	100.0	0	0	–	–	54.4	10	10	488	511.0	60.8	10	10	130.1	122	88.8
AT2	10	10	18.8	13.5	99.9	10	10	8	12.9	79.9	10	10	96	131.7	76.3	8	8	727.9	596	93.8
AT51	10	10	20.5	16.5	100.0	10	10	10	19.0		10	10	108	114.2	59.7					
AT52	10	10	74.1	65.0	100.0	10	10	47	107.5		4	4	19	20.5	52.1					
AT53	10	10	1.5	1.0	99.9	10	10	1	1.4		10	10	1	1.3	93.2					
AT54	10	10	47.9	42.0	100.0	10	10	42	55.4		8	8	67	215.5	49.1					
AT6a	10	10	156.6	138.0	100.0	10	10	102	241.7	49.6	10	10	109	108.1	52.6					
AT6b	10	10	472.2	588.0	100.0	10	10	193	412.4	46.4	10	10	131	132.7	52.3					
AT6c	10	10	326.8	176.0	100.0	10	10	238	412.4	40.7	10	10	80	99.2	44.4	10	0	186.6	172	89.5
AT6abc	10	10	149.0	125.5	100.0	10	10	238	412.4	22.0	10	10	129	132.6	23.7					
AFC27	0	–	–	–	100.0	10	–	8	8.5	1.4	8	–	142	204.2	29.1					
AFC29	10	–	25.1	19.0	100.0	10	–	2	5.3	8.5	10	–	25	31.4	36.6					
AFC33	0	–	–	–	100.0	0	0	–	–	53.7	0	–	–	–	35.8					
NN	10		277.2	158.5	100.0	10	–	61	83.7	6.0	10	–	25	35.8	46.9					
NN$_{\beta=0.04}$						5	–	8	91.4	2.9	1	–	1475	1475.0	33.9					
NNx	8		457.1	380.5	100.0	0	–	–	–	4.1	0	–	–	–	67.2					
CC1	10	10	16.4	9.5	100.0	10	10	8	9.1	82.5	10	10	18	34.7	88.1					
CC2	10	10	12.4	13.0	100.0	10	10	9	10.8	74.7	10	10	82	66.7	82.4					
CC3	10	10	19.6	21.0	100.0	10	10	13	12.8	56.5	10	10	50	66.0	78.4					
CC4	0	0	–	–	100.0	0	0	–	–	57.6	1	1	1479	1479.0	81.5					
CC5	10	10	37.4	22.0	100.0	10	10	11	21.1	62.3	10	10	108	127.3	81.7					
CCx	6	6	396.7	284.5	100.0	9	9	394	495.4	38.4	8	8	112	112.1	64.7					
SC						0	0	–	–	58.8	0	–	–	–	95.7					
PM	6	–	575.8	617.0	100.0	6	–	826	734.7	17.2	8	–	362	466.4	10.9					

Tool:	**ForeSee**					**NNFal**					**Ψ-TaLiRo**					**STGEM**				
Approach:											ConBO					OGAN				
Benchmark	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*	*FR*	✓	$\overline{S}$	$\widetilde{S}$	*R*
AT1	8	8	329.1	330	99.5	2	2	1.5	1.5	99.5	10	10	105.4	105.0	86.5	10	10	185.8	153	29.2
AT2	10	10	19.1	17	99.0						10	10	11.6	11.0	50.6	10	10	15.3	11	100.0
AT51	10	10	22.8	19.5	98.7						10	10	13.7	8.5	66.1	10	10	15	13	100.0
AT52	10	10	51	39	98.9						10	10	79.1	95.0	93.2	10	10	6.3	3.5	100.0
AT53	10	10	3	1	96.5						10	10	2.7	2.0	58.0	10	10	2	2	100.0
AT54	10	10	108.8	44	99.0						10	10	37.7	32.0	79.9	10	10	18.4	12	100.0
AT6a	10	10	104.8	105	99.3						9	9	255.0	226.0	52.4	10	10	74.2	69.5	60.4
AT6b	10	10	285.2	255	99.3						9	9	580.0	522.0	58.2	10	10	128.3	129	38.3
AT6c	10	10	117.8	111	99.3	1	1	3	3	99.62	7	7	191.1	148.0	58.9	10	10	89.6	88.5	66.4
AT6abc	10	10	111.7	83	98.8						10	10	240.5	74.0	7.8	10	10	63.8	67.5	55.4
AFC27	10	–	2.6	2	99.0						10	–	113.2	109.5	97.8	10	–	69.4	76	95.0
AFC29	10	–	1	1	98.4						10	–	19.6	19.0	100.0	10	–	11.5	8.5	100.0
AFC33	10	–	1	1	96.4						0	–	–	–	35.0	0	–	–	–	45.8
NN	10	–	33.6	33	98.5						10		155.5	100.0	88.0	10	–	120.4	135.5	50.8
NN$_{\beta=0.04}$																7	–	607.7	482	28.6
NNx											10		121.5	119.5	82.6	0	–	–	–	23.9
CC1	10	8	23.4	17	99.0	9	9	1.22	1	99.68	10	10	10.8	8.0	69.8	10	10	33.3	21	88.0
CC2	5	5	6.7	6	99.3						10	10	9.6	7.0	67.5	10	10	65.9	66	92.1
CC3	10	10	22.4	14	98.6						10	10	11.7	8.0	69.4	10	10	20.3	24	100.0
CC4	9	6	600.6	542	99.3						1	1	1248.0	1248.0	95.9	0	–	–	–	18.5
CC5	10	10	64.5	42	98.9						10	10	28.3	27.0	72.3	10	–	43.7	46.5	95.0
CCx	10	10	211.2	213	98.6						10	10	240.5	74.0	35.4	6	6	635.3	534	19.9
SC																0	–	–	–	7.1
PM	0	–	–	–	99.2						10	–	121.5	119.5	82.6	10	–	245.1	166.5	19.3

5 Probabilistic Guarantees

As done in 2022, we are still assessing tools that can provide probabilistic guarantees for falsifying the system under test to understand if we can provide any conclusion about the system under test and falsifying it. This information becomes even more critical when no falsification is found. Providing probabilistic guarantees can help assess the system's safety, while also providing the quality of test samples generated.

Part-X [29] was the only tool that provided results on probabilistic guarantees: the lower and upper confidence bounds of normalized falsification volume at 95% confidence. The Part-X algorithm is part of the Ψ-TaLiRo tool, and is discussed in section 3.7. The results are shown in Table 4 for both instances. We refrain from an in-depth analysis of these results.

Table 4: 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	10	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	10	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	10	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
PM	10		23.5	22.0	6.75E-3	7.13E-3	80.9	8		253.6	26.5	0.00E+00	3.44E-3	82.7

6 Conclusion and Outlook

The extension of the benchmark set with the Pacemaker benchmark enabled a more extensive comparison of existing tools, which now also includes a model and requirement from the medical domain. The requirement is not trivially falsified. This competition enables maintaining this benchmark, which is getting traction in the falsification literature, maintained and updated.

The participation of new tools (ATheNA [18], NNFal [28], and STGEM [34]) shows that this competition has been getting attention over the years. The tools now use completely different technologies that are challenging to compare. However, the results reported in Tables 2 and 3 can provide a starting point for this comparison. Based on our data, we remark that there is no "best" approach, and the different solutions offer pros and cons that engineers should consider when selecting the appropriate falsification tool. Notice that some tools did not consider all the models and requirements. Considering only some models and requirements was permitted since models use different Simulink features, often requiring some tuning of the falsification tools. This tuning is often not easy, especially for new participants.

Gidon Ernst, Tanmay Khandait, and Walstan Baptista were pivotal for enabling the validation of the results. They set up the validation platform, which was far from trivial, and had to solve many technical challenges. In addition, they provided a timely and thorough support to the other participants during the validation of the results. Our findings stress the importance of their work and 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 [6]), Test-Comp (which had independent coverage evaluation from the start in 2019 [7]), and many other competitions (for an overview, see [4]).

We have several items on our agenda for the following year's competition. First, we would like to facilitate the adoption of Python-based benchmark models from Matlab-based benchmarks. Supporting Python programs will require a careful revision of the rules of the competition. Second, we would like to employ the automatic repeatability evaluation platform provided by the organizers of the ARCH competition. Considering the number of participants, and the tight deadlines, we decided to postpone the usage of this platform this year since it would have required participants to invest additional time in learning the use of the replication platform. Finally, we would like to stimulate and expand the competition rules related to the tools supporting probabilistic guarantees.

Acknowledgments Many thanks to the organizers of the ARCH workshop 2023 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. The participants thank Gidon Ernst, Tanmay Khandait, and Walstan Baptista for developing the validation platform and for their support during the validation phase.

C. Menghi and F. Formica are supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), [funding reference number RGPIN-2022-04622,DGECR-2022-0040]. P. Arcaini is supported by Engineerable AI Techniques for Practical Applications of High-Quality Machine Learning-based Systems Project (Grant Number JPMJMI20B8), JST-Mirai; and ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST, Funding Reference number: 10.13039/501100009024. The ASU team (Ψ-TaLiRo) was partially supported by DARPA FA8750-20-C-0507, NSF CMMI 2046588, NSF CNS 2000792, and NSF CMMI 1829238. Z. Zhang is supported by JSPS KAKENHI Grant No. JP23K16865 and Grant No. JP23H03372. J. Peltomäki and I. Porres are supported by the ECSEL Joint Undertaking (JU) under grant agreement No 101007350. The JU receives support from the

European Union's Horizon 2021 research and innovation programme and Sweden, Austria, Czech Republic, Finland, France, Italy, Spain. M. Waga is partially supported by JST ACT-X Grant Number JPMJAX200U, JSPS KAKENHI Grant Number JP22K17873, 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 *IEEE Congress on Evolutionary Computation, CEC 2005*, pages 1769–1776, 2005.

[3] Mostafa Ayesh, Namya Mehan, Ethan Dhanraj, Abdul El-Rahwan, Simon Emil Opalka, Tony Fan, Akil Hamilton, Akshay Mathews Jacob, Rahul Anthony Sundarrajan, Bryan Widjaja, and Claudio Menghi. Two simulink models with requirements for a simple controller of a pacemaker device. In *International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH22)*, EPiC Series in Computing, pages 18–25. EasyChair, 2022.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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 *ARCH18. International Workshop on Applied Verification of Continuous and Hybrid Systems*, EPiC Series in Computing, pages 104–109. EasyChair, 2018.

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

[12] 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 *International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21)*, EPiC Series in Computing, pages 133–152. EasyChair, 2021.

[13] 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 *ARCH20. International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)*, EPiC Series in Computing, pages 140–152. EasyChair, 2020.

[14] 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 *ARCH19. International Workshop on Applied Verification of Continuous and Hybrid Systems*, EPiC Series in Computing, pages 129–140. EasyChair, 2019.

[15] Gidon Ernst, Paolo Arcaini, Georgios Fainekos, Federico Formica, Jun Inoue, Tanmay Khandait, Mohammad Mahdi Mahboob, Claudio Menghi, Giulia Pedrielli, Masaki Waga, Yoriyuki Yamagata, and Zhenya Zhang. Arch-comp 2022 category report: Falsification with ubounded resources. In *International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH22)*, EPiC Series in Computing, pages 204–221. EasyChair, 2022.

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

[17] 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.

[18] Federico Formica, Mehrnoosh Askarpour, and Claudio Menghi. Search-based software testing driven by automatically generated and manually defined fitness functions. *arXiv preprint arXiv:2207.11016*, 2022.

[19] 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.

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

[21] Bardh Hoxha, Houssam Abbas, and Georgios Fainekos. Benchmarks for temporal logic requirements for automotive systems. In *ARCH14-15. International Workshop on Applied veRification for Continuous and Hybrid Systems*, EPiC Series in Computing, pages 25–30. EasyChair, 2015.

[22] 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.

[23] Xiaoqing Jin, Jyotirmoy V. Deshmukh, James Kapinski, Koichi Ueda, and Ken Butts. Powertrain control verification benchmark. In *International Conference on Hybrid Systems: Computation and Control*, pages 253–262. ACM, 2014.

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

[25] 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. Springer, 2004.

[26] 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.

[27] Dejan Ničković and Tomoya Yamaguchi. RTAMT: Online Robustness Monitors from STL. In *Automated Technology for Verification and Analysis: International Symposium (ATVA)*, page 564–571. Springer-Verlag, 2020.

[28] NNFal. https://gitlab.com/Atanukundu/NNFal, 04 2023 [Online].

[29] 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. https://arxiv.org/abs/2110.10729, 2021.

[30] Doron Peled, Moshe Y Vardi, and Mihalis Yannakakis. Black box checking. In *Formal Methods for Protocol Engineering and Distributed Systems: FORTE XII/PSTV XIX'99 IFIP TC6 WG6. International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XII) and Protocol Specification, Testing and Verification (PSTV XIX)*, pages 225–240. Springer, 1999.

[31] Jarkko Peltomäki and Ivan Porres. Falsification of multiple requirements for cyber-physical systems using online generative adversarial networks and multi-armed bandits. In *IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pages 21–28, 2022.

[32] Jarkko Peltomäki, Frankie Spencer, and Ivan Porres. Wasserstein generative adversarial networks for online test generation for cyber physical systems. In *IEEE/ACM International Workshop on Search-Based Software Testing, SBST 2022*, page 1–5, 2022.

[33] Ivan Porres, Hergys Rexha, and Sebastien Lafond. Online GANs for automatic performance testing. In *IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW 2021)*, pages 95–100, 2021.

[34] STGEM. https://gitlab.abo.fi/stc/stgem, 04 2023 [Online].

[35] 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 *Formal Methods for Industrial Critical Systems*, pages 223–231. Springer, 2021.

[36] Masaki Waga. Falsification of cyber-physical systems with robustness-guided black-box checking. In *International Conference on Hybrid Systems: Computation and Control (HSCC)*, pages 11:1–11:13. ACM, 2020.

[37] 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.

[38] 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.

[39] 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 *Computer Aided Verification*, pages 595–618. Springer, 2021.