EPiC Series in Computing

Volume 103, 2024, Pages 122–144

Proceedings of the 11th Int. Workshop on Applied Verification for Continuous and Hybrid Systems

ARCH-COMP 2024 Category Report: Falsification

Tanmay Khandait[1], Federico Formica[2*], Paolo Arcaini[3], Surdeep Chotaliya[1], Georgios Fainekos[4], Abdelrahman Hekal[5], Atanu Kundu[6], Ethan Lew[7], Michele Loreti[8], Claudio Menghi[9,2], Laura Nenzi[10], Giulia Pedrielli[1], Jarkko Peltomäki[11], Ivan Porres[11], Rajarshi Ray[6], Valentin Soloviev[11], Ennio Visconti[12], Masaki Waga[13], and Zhenya Zhang[14]

[1] Arizona State University (ASU), Tempe, USA {tkhandai,schotali,gpedriel}@asu.edu
[2] McMaster University, Hamilton, Canada {formicaf,menghic}@mcmaster.ca
[3] National Institute of Informatics (NII), Tokyo, Japan arcaini@nii.ac.jp
[4] Toyota Motor North America, Research & Development georgios.fainekos@toyota.com
[5] Newcastle University, Newcastle upon Tyne, UK a.waleed-elsayed-aly-hekal@newcastle.ac.uk
[6] Indian Association for the Cultivation of Science, Kolkata, India {mcsak2346,rajarshi.ray}@iacs.res.in
[7] Galois Inc, Portland, USA elew@galois.com
[8] University of Camerino, Camerino, Italy michele.loreti@unicam.it
[9] University of Bergamo, Bergamo, Italy claudio.menghi@unibg.it
[10] University of Trieste, Trieste, Italy lnenzi@units.it
[11] Åbo Akademi University, Turku, Finland {jarkko.peltomaki,ivan.porres,valentin.soloviev}@abo.fi
[12] TU Wien, Vienna, Austria ennio.visconti@tuwien.ac.at
[13] Kyoto University, Japan mwaga@fos.kuis.kyoto-u.ac.jp
[14] Kyushu University, Japan zhang@ait.kyushu-u.ac.jp

Abstract

This report presents the results from the falsification category of the 2024 competition in the Applied Verification for Continuous and Hybrid Systems (ARCH) workshop. The report summarizes the competition rules and settings, the benchmark models for the tool comparison, and provides background on the participating teams and tools. Finally, it presents and discusses the results of the competition.

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

1 Introduction

The Applied Verification for Continuous and Hybrid Systems (ARCH) competition is a yearly competition comparing state-of-the-art tools for testing and verifying hybrid systems. The com-

*The first two authors lead the validation effort for the falsification category. The remaining authors represent all participants who have contributed results and/or text to this report and they are listed alphabetically.

petition (a.k.a. ARCH-COMP) is organized in different categories. This document presents the results from the 2024 *falsification category*. Past reports (2017–2023) for this category are available online [11, 12, 18, 17, 16, 19, 33]. The falsification category targets the analysis of executable models. The participants need to falsify requirements expressed in temporal logic with time bounds, encoded in *Metric Temporal Logic (MTL)* [29] or *Signal Temporal Logic (STL)* [32] by searching for initial system configurations and time-varying inputs subject to given constraints. Typical approaches employed by participants are simulation-based and employ quantitative metrics [21, 23] measuring how close a given input is to violating a requirement ("robustness semantics"). Recent survey articles [7, 10] summarize the research results for this area. The benchmark set developed by this competition series can be seen as a baseline for research in the area [14]: We encourage authors to compare their solutions with the one presented in this report.

The competition of 2024 followed the structure of previous years: Participants agreed on the benchmarks to be considered, ran the experiments on their machines, and submitted the results including concrete input traces that witnessed falsification. The first three authors of the paper were involved in the validation step, which analyzed the input traced submitted by the participants to confirm that they falsified the requirements. Compared with the previous edition [33] the changes are as follows:

- We had three new tools participating in the competition: EXAMNET, Moonlight [36], FalStar [20][1], FReaK [5], OD. One tool (OGAN) decided not to participate this year. The remaining tools ARIsTEO [34], ATheNA [22], `FalCAuN` [46], ForeSee [48], NN-Fal [30], and Ψ-TaLiRo [45] confirmed their participation.
- We added the simulation time and the sample step for all the models within the report (Table 1).
- For the benchmark model NNx, the requirement for this year in instance 1 requires discontinuities to be at least one time unit apart (instead of requiring discontinuities to be at least three time units apart)
- For the benchmark model CC4, we fixed a discrepancy between the competition report and the requirement presented online and considered by the participants.
- For AT51, we increased the simulation time to ensure that the requirements are "fully evaluated" (Table 1).
- We asked all the participants to document the configuration of additional parameters used in their tools within the report or online.
- We removed column the ratio between the simulation time and the total falsification time (column "R") from Tables 5 and 6 since we discovered different teams were using different formulae for its computation. We plan to precisely formalize this metric in the next edition of the competition to ensure a fair comparison between tools.

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

[1] Unfortunately, the authors of FalStar could not submit the results in time due to other commitments.

in the subfolders `models/FALS` and `2024/FALS`. This archive contains the results of validation and instructions to re-validate the results.

2 Benchmark

This section presents the two parametrizations of the input domain to generate the input signals for the models (Section 2.1), and the benchmark models and requirements (Section 2.2) for this edition of the competition.

2.1 Input Parameterization

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

Arbitrary piecewise continuous input signals (Instance 1). The input specification is up to the participants. The search space is the set of piecewise continuous input signals (i.e., discontinuities are permitted), where the values for each individual dimensions are from a given range. 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 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). It fixes the format of the input signal, potentially allowing discontinuities. An example input signal would be a piecewise constant with k equally spaced control points, with ranges for each input dimension, 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 are just compared concerning their optimization algorithm. Nevertheless, such a comparison is still meaningful since fundamentally different approaches to falsification have entered the competition.

2.2 Models and Requirements

We provide a short textual description of our benchmark models. Table 1 reports the simulation time and the sample step for all of our models. Table 2 reports the corresponding requirements formalized as STL/MTL formulas.

Automatic Transmission (AT) - [26].[2] A controller selecting the gear (from 1 to 4) depending on two inputs (throttle, brake) and the current engine load, rotations per minute ω, and car speed v.

Input specification: $0 \leq \mathit{throttle} \leq 100$ and $0 \leq \mathit{brake} \leq 325$ (both can be active simultaneously). Constrained input signals (instance 2) permit discontinuities at most every five

[2] Derived from a model proposed by Mathworks.

Table 1: Minimum Simulation time and the sample step for all the benchmark models.

Model	Simulation time	Sample step
AT	33s	0.01s
AFC	51s	0.01s
NN	40s	0.01s
CC	100s	0.01s
F16	15s	0.01s
SC	35s	0.01s
PM	10s	0.01s

time units. Requirements are specific versions of those in [26] where the parameters have been chosen to be somewhat difficult.

Fuel Control of an Automotive Powertrain (AFC) - [28]. A controller for the air-fuel ratio in an automotive powertrain engine. The values used in the requirements are chosen to make falsification feasible 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 [28]. We do not consider the unconstrained (instance 1) input specification. Faults are disabled (e.g. by setting `fault_time` > 50).

Neural-network Controller (NN) - [13].[3] 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 a 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) - [27]. 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) - derived from [24]. 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

[3]Derived from a model proposed by Mathworks https://au.mathworks.com/help/deeplearning/ug/design-narma-l2-neural-controller-in-simulink.html

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) [47]. A dynamic model of a 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) [4]. A controller for a pacemaker device: It artificially contracts the heart muscle when no natural activity is present for a given time. The input is the desired lower rate limit that can change within the range $[50, 90]$.

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 [34][4] is a Matlab toolbox for test case generation against system specifications developed on the top of S-TaLiRo [2]. 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).[5]

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 $nk = 2$[6] 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

[4] Participants: Menghi and Formica.
[5] https://github.com/SNTSVV/ARIsTEO
[6] https://nl.mathworks.com/help/ident/ref/arx.html

Table 2: 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	$\square_{[0,65]} \diamondsuit_{[0,30]} \square_{[0,5]} y_5 - y_4 \geq 8$	
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	$\square_{[0,10]}(\mathit{paceCount} \leq 15) \wedge \diamondsuit_{[0,10]}(\mathit{paceCount} \geq 8)$	

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 [22][7] 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

[7]Participants: Formica and Menghi.

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 of leading to the identification of a fault. A brief description of the manual fitness functions and the weight used for the automatic fitness for each requirement is reported in Table 3. 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.

3.3 EXAM-Net

Description. EXAMNET (Exploratory Adversarial Mutator Network)[8] is a black box falsification algorithm written using the STGEM[9] open-source toolbox. The algorithm takes inspiration from OGAN [42] as well as diffusion-based algorithms and consists of three neural networks: discriminator, mutator and validator. Using executed tests the discriminator is trained to learn the mapping from proposed inputs to robustness values. The mutator is trained to take any random input and mutate it so that discriminator's score for the mutated tests is lower than discriminator score for initial random input, but also so that the L2-distance between the two is minimized. The first condition ensures that the network learns to mutate tests so that they have lower estimated robustness, while the latter condition lets the network avoid mode collapse. Validator is trained to identify invalid tests and is used to provide a proper training signal for the mutator to avoid generating invalid tests. We emphasize that no prior training or data collection is necessary - all the training is performed during runtime as tests are being generated and executed.

Setup. As EXAMNET is an algorithm that requires training the neural networks, the first 40 tests are selected by sampling input space uniformly randomly. We have noted this to be a sufficient amount for the model to start generating reasonable suggestions for tests. Additionally, every tenth test is also sampled in this manner to promote exploration and help the model avoid local minima.

To keep results comparable with other models implemented in STGEM, the hyperparameters and other settings for the models are kept as close as possible to the other models in the toolbox. EXAMNET also utilizes in its search the STL robustness objective [32]. The model targets only instance 2, so the values for EXAMNET in Table 5 are the same as in Table 6. One of the

[8] Participants: Soloviev

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

Table 3: Manual fitness functions description and weight used by ATheNA for the benchmark requirements. The weight refers to the automatic fitness function; the manual fitness weight is equal to 1 minus the automatic fitness weight.

Benchmark	Weight	Manual Fitness Description
AT1	0.4	Maximizes the lowest throttle value within $[0, 17]$s and minimizes the highest brake value within $[0, 25]$s.
AT2	0.5	Maximizes the average throttle value within $[0, 8]$s, then minimizes the average brake value within $[0, 25]$s.
AT51	0.0	Makes the first three throttle control points get as close as possible to $\{35\%, 0\%, 50\%\}$ respectively and maximize brake within $[0, 25]$s.
AT52	0.5	Maximizes the minimum throttle value between $[0, 8]$s.
AT53	0.4	Makes the first three throttle control points get as close as possible to $\{100\%, 20\%, 0\%\}$ respectively and minimize brake within $[0, 25]$s.
AT54	0.0	Makes the first three throttle control points form an upward arc and the brake ones a downward arc.
AT6a	0.6	Makes the average throttle value within $[0, 33]$s as close as possible to 45% and minimizes the average brake value within $[0, 25]$s.
AT6b	0.4	Makes the average throttle value within $[0, 33]$s as close as possible to 45% and minimizes the average brake value within $[0, 25]$s.
AT6c	0.8	Makes the average throttle value within $[0, 33]$s as close as possible to 45% and minimizes the average brake value within $[0, 25]$s.
AT6abc	0.5	Makes the average throttle value within $[0, 33]$s as close as possible to 45% and minimizes the average brake value within $[0, 25]$s.
AFC27	0.2	Increases the two control points adjacent to the lowest one above 40 deg, then minimizes the lowest value within $[10, 50]$s.
AFC29	0.5	Minimizes the lowest throttle value within $[10, 50]$s.
AFC33	0.5	Minimizes the engine speed value.
NN	0.2	Minimizes the reference position control point at 20s.
NNx	0.5	Maximizes the lowest reference position within $[0, 20]$s.
CC1	0.5	Maximizes the lowest throttle value within $[0, 100]$s and minimizes the highest brake value within $[0, 100]$s.
CC2	0.4	Minimizes the highest throttle value within $[0, 100]$s and maximizes the lowest brake value within $[0, 100]$s.
CC3	0.8	Maximizes the lowest throttle value within $[0, 100]$s and minimizes the highest brake value within $[0, 100]$s.
CC4	0.5	Minimizes the minimum distance between cars 4 and 5 within $[0, 100]$s.
CC5	0.5	Makes the average throttle value within $[0, 33]$s as close as possible to 0.3 and maximizes the average brake value within $[0, 50]$s.
CCx	0.4	Maximizes the throttle control point at $0s$ and minimizes the throttle control point at $17s$.
F16	0.5	Maximizes the initial roll angle and minimizes the initial pitch angle.
SC	0.6	Maximizes the peak-to-peak distance of the steam flow rate within $[29.5, 35]$s.
PM	0.5	Minimizes the highest lower rate limit within $[0, 10]s$.

benchmarks had small validation mismatch due to version difference and some validation results are omitted since the validation tool does not support all benchmarks.

3.4 FalCAuN

Description. FalCAuN [46][10] is an experimental toolkit for testing a Simulink model using black-box checking [39], an automated testing method based on active automata learning and

[10]Participant: Waga.

Table 4: Configuration parameters for FalCAuN.

Model	Duration between control points	Possible input values
AT	2.0	throttle: 0.0, 50.0, 100.0; brake: 0.0, 325.0
CC	5.0	throttle: 0.0, 1.0; brake: 0.0, 1.0
SC	1.0	3.99, 4.00, 4.01
PM	0.5	50.0, 60.0, 70.0, 80.0, 90.0

model checking. In FalCAuN, the input and output signals given to the Simulink model under testing are discretized in time and values, and the model is deemed a black-box transition system with potentially infinite states. FalCAuN learns an approximation of the transition system as a Mealy machine and conducts model checking to find a counterexample. By reusing the learned Mealy machine, FalCAuN is designed to falsify a Simulink model against multiple specifications efficiently. FalCAuN is publicly available under General Public License (GPL) v3[11].

We employ 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, to capture the timing constraint $\Diamond_{[0,0.05]}$, the duration between the control points must be at most 0.05. Due to this restriction, FalCAuN cannot handle the STL formulas with such small timing constraints. Also, since some of the behaviors between control points are ignored in discrete-time semantics, the falsification results may deviate from the standard semantics. We remark that the current version of FalCAuN can handle the maximum and minimum values between control points, which prevents the above deviation if the temporal operators are not nested. The current version of FalCAuN only supports piecewise linear signals, and we have no results for instance 2.

Setup. For the signal discretization, we have the following parameters: the (constant) duration step of the intervals between control points, the possible values I of input signals at control points, and the thresholds of output signal values for discretization. We use 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 0.5 to 5.0 time units. We used the constants in the given STL formulas as the thresholds of the output signal values. Table 4 summarizes the parameters we used.

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 [48][12] (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.

[11] https://github.com/MasWag/FalCAuN
[12] Participants: Zhang, Arcaini

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 [18, 17]. It is publicly available under GNU General Public License (GPL) v3.[13]

Setup. Since ForeSee is implemented on the basis of Breach [13], 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 [3] 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, which takes a balance of *exploration* and *exploitation*; and we set 10 generations as the budget for the playout phase of MCTS.

3.6 FReaK

Description. FReaK [5][14] is a falsification framework for black-box models that uses an iterative refinement technique based on surrogate modeling. In particular, simulations are used to construct a surrogate model for the system dynamics using data-driven Koopman operator linearization. The reachable set of states are then computed and combined with an encoding of the signal temporal logic specification in a mixed-integer linear program (MILP). To determine the next sample, the MILP solver computes the least robust trajectory inside the reachable set of the surrogate model. The trajectory's initial state and input signal are then executed on the original black-box system, where the specification is either falsified or additional simulation data is generated that we use to retrain the surrogate Koopman model and repeat the process. FReaK is publicly available[15].

Setup. The configuration parameters for FReaK depend on its main constituents. For learning the Koopman surrogate model, we use AutoKoopman [31], which automatically tunes for the associated hyperparameters. We use random Fourier features as observables with an upper bound of 20 observables, and apply grid-search for hyperparameter optimization. We use CORA [1] for reachability analysis and Gurobi[16] for solving the MILP optimization problems. Koopman linearization, reachability analysis and MILP optimization each have an associated time step size parameter determined by the chosen discretization. For consistency, we use the same time step across all three processes. Moreover, we use the number of control points of an input signal to determine the time step size for the processes. For instance, given a system with a time horizon of $100s$ and 10 control points, the associated time step size is $\Delta t = 10$. The only exception is the AFC benchmark, where a time step size of $\Delta t = 5$ was found to be too coarse for effective analysis, so we used $\Delta t = 1$ instead. We use a consistent number of control

[13] https://github.com/choshina/ForeSee
[14] Participants: Hekal and Lew.
[15] https://github.com/Abdu-Hekal/FReaK
[16] https://www.gurobi.com/

points for each model across all requirement formulas, where all signal are evenly spaced and interpolated with the `pchip` function or piecewise constant interpolation.

3.7 Moonlight

Description. Moonlight [36][17] is an open source[18] lightweight tool for monitoring (online or offline) temporal and spatio-temporal properties. The specification language adopted by Moonlight is STL or its spatial extension, STREL [35]. It can efficiently process piece-wise constant multi-valued numerical, Boolean, and categorical types of signals. The result of the monitoring process can either be in terms of Boolean satisfaction or of real-valued robustness, depending on the purpose of the monitoring process. Moonlight offers Java, Python, and Matlab APIs. Moreover, it supports a minimal scripting language for defining signal interfaces and formulae monitors. The falsification process of this competition is managed by adopting the TuRBO [15] optimizer. TuRBO performs Bayesian Optimization using trust regions, a technique where simultaneous local runs of independent models are used. To falsify, a multi-armed bandit strategy is used at each iteration to allocate samples for each dimension and thus decide which local optimization to run. The function to optimize is the robustness value of the target formula, which is retrieved by running Moonlight on the signals generated by a simulation using the parameters provided by the optimizer.

Setup. The current implementation of the benchmarks is based on the Moonlight Python APIs, and the combination with TuRBO makes its usage straightforward, where only the maximum number of iterations (300) and the interval of acceptable parameters for each dimension is necessary. To maximize reproducibility, the benchmark framework has been built so that both the monitor and the optimizer can be swapped with alternative ones as long as their interfaces are honored. The interested reader can run the benchmarks locally, as well as tweak any of the optimization parameters[19].

3.8 NNFal

Description. NNFal [30][20] 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.[21]

[17] Participants: Loreti, Nenzi and Visconti
[18] https://github.com/MoonLightSuite/moonlight
[19] https://github.com/MoonLightSuite/arch-comp-benchmarks
[20] Participants: Kundu and Ray
[21] https://gitlab.com/Atanukundu/NNFal

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 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 5 are simply copies of those in Table 6.

3.9 OD

Description. OD[22] (online diffusion) is a black-box requirement falsification algorithm based on diffusion models. It implements the DDPM diffusion model of [25], and the chosen denoising model is a modified UNet [43]. The model denoises Gaussian noise to test vectors, which are representations of piecewise constant signals (i.e., OD targets Instance 2 input parameterization). The target distribution of the diffusion model is the set of falsifying tests. The training of the model is done in an online fashion in such a way that the training data distribution shifts progressively towards test that have low STL robustness. This is similar to the online training of Wasserstein Generative Adversarial networks in the WOGAN algorithm [41]. During a round of the algorithm, the diffusion model denoises tests and selects a candidate test among the denoised tests that is then executed on the system under test. The candidate test is selected uniformly randomly 50% of the time (to explore), and for the remaining time the test estimated to have the lowest robustness by a random forest regression model is selected (to exploit). The diffusion model training data is then augmented with the executed test together with its true STL robustness. An initial training data is obtained by uniform random sampling of the input space. The diffusion model is trained and sampled until a falsifying input is found or the execution budget is exhausted. The OD algorithm does not use any precollected data or models; the models are trained from scratch.

The OD algorithm is implemented in the STGEM tool [44], which is a falsification framework supporting several requirement falsification algorithms. The code and instructions to use OD are available online[23].

Setup. The input space for each benchmark is determined by a vector whose components correspond to the pieces of the piecewise constant input signals. The components vary in the ranges specified in Section 2. Of the total simulation budget of 1500 the first 50 are reserved for uniform random sampling of the input space to obtain the initial training data. The DDPM

[22] Participants: Peltomäki and Porres

[23] https://gitlab.abo.fi/stc/experiments/arch-comp-2024

model uses a sinusoidal time embedding with dimension 20, and the forward process applies Gaussian noise for 75 iterations and the forward process variances increase from 0.0001 to 0.02 as suggested in [25]. During each training phase, the DDPM model is trained for 20 epochs using the Adam optimizer with learning rate 0.001. We use the same hyperparameter setup for all benchmarks. We have manually tuned the hyperparameters to give a good overall performance.

For the conjunctive requirements, we use the multi-armed bandit approach described in [40] meaning that for N requirements we use N distinct diffusion models, one per requirement, but only sample one of them for a test.

We target only the instance 2, so the values for OD in Table 5 are the same as in Table 6. Some validation results are omitted since the validation tool does not support all benchmarks.

3.10 Ψ-TaLiRo

Description. Ψ-TaLiRo [45][24] , the python version of S-TaLiRo [2], 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).[25] For this competition, we provide results with two different optimization algorithms:

1. **Conjunctive Bayesian Optimization - Large Scale (ConBO-LS)** accounts for the dependencies between two requirements and leverages their mutual information to achieve relatively early falsification. Additionally, the approach employs a sampler to select a subset of requirements that are more promising for falsification. Unlike the conventional approach, the algorithm takes the conjunction of all the requirements from a particular benchmark model and attempts to falsify this conjunction. It is important to note that this algorithm turns into a simple Bayesian Optimization if we do not have conjunctive requirements. However, once a requirement is falsified, the algorithm proceeds with the conjunction of the remaining unfalsified 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 [38].

Setup. In Ψ-TaLiRo, input signals to black-box models are parameterized with control points and their corresponding timestamps (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 [37] for robustness calculation.

The repeatability package for ConBO-LS and Part-X is available online[26]. In the instances

[24] Participants: Khandait, Chotaliya, Fainekos, and Pedrielli
[25] https://github.com/cpslab-asu/psy-taliro
[26] https://github.com/cpslab-asu/ARCH-Comp-2024-Repeatability

presented in this paper, ConBO-LS utilizes only a 1500-evaluation budget to falsify all the requirements from a particular benchmark model, rather than allocating 1500 evaluations for each individual requirement. In other words, the evaluation budget is allocated to a benchmark model. For example, in the Automatic Transmission benchmark in instance 1, the ConBO-LS tries to falsify the conjunction of all the 10 requirements. If any particular requirement is falsified, the algorithm continues with conjunction of the remaining requirements. In these experiments, the ConBO-LS optimizer samples 100 points from the search space and then sequentially samples points until all the requirements are falsified or the maximum budget of 1500 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 7.

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).

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 5 and Table 6 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 ($\overline{S}$), and median (rounded down) number of simulations ($\widetilde{S}$). $\overline{S}$ and $\widetilde{S}$ are computed using the successful executions, i.e., the executions where the falsification was successful. 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 in instance 1.

For some of the tools, e.g., `FalCAuN` (CC4 — instance 1), 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) and the Aircraft Ground Collision Avoidance System (F16) benchmarks. We plan to address these limitations in the next edition of the competition.

Table 5: 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.

Tool:	**UR**			**ARIsTEO** ARX-2			**ATheNA**			**EXAM-Net**			**FalCAuN**			**ForeSee**			**FReaK**			**Moonlight**			**NNFal**			**OD**			**Ψ-TaLiRo** ConBO-LS		
Approach:																																
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*	✓	$\overline{S}$	$\widetilde{S}$	*FR*	✓	$\overline{S}$	$\widetilde{S}$
AT1	0	0	–	–	0	0	–	–	10	10	150.4	98	10	10	896.0	896	10	10	387.4	429.0	10	10	4.8	4					2	2	1.5	1.5	0	0	–	–	0	–	–	–
AT2	10	10	7.6	5.0	10	10	25.9	16.0	9	9	105.0	47.0	10	10	22.7	31	10	10	256.0	256	9	9	196.6	38.5	10	10	2.1	2					10	10	18.5	13.0	10	10	9.7	10.0
AT51	1	1	923.0	923.0	1	1	1166.0	1166.0	3	3	370.3	321.0	10	10	8.5	4					10	10	15.0	14.5	10	10	8.7	6.5					10	10	11.6	10.0	0	–	–	–
AT52	10	10	4.1	2.0	10	10	5.0	3.0	10	10	17.3	9.0	10	10	10.3	7.5					10	10	65.7	35.0	10	10	1.3	1.0					10	10	8.6	6.0	10	10	6.3	5.0
AT53	10	10	18.6	15.0	10	10	5.8	6.0	10	10	51.4	20.5	10	10	1.5	1					10	10	4.3	3.5	10	10	1.1	1.0					10	10	2.2	2.0	10	10	19.7	20.5
AT54	3	3	932.0	868.0	7	7	554.7	709.0	1	1	6.0	6.0	10	10	89.6	12.5					10	10	60.3	29.5	10	10	2.4	1.0					10	10	18.4	10.0	6	6	1008.3	1158.0
AT6a	10	10	74.4	41.5	9	9	231.0	272.0	10	10	200.0	144.0	10	10	61.1	59.5	10	10	1002.0	1002	10	10	115.2	96.5	10	10	7.4	5.0					10	10	64.3	61.0	10	10	187.4	82.0
AT6b	10	10	251.3	189.0	5	5	591.4	621.0	8	8	345.1	310.0	10	10	119.4	108	0	0	–	–	10	10	253.9	274.5	10	10	6.2	5.0					10	10	122.0	94.0	9	9	544.6	324.5
AT6c	10	10	185.2	86.0	10	10	329.2	227.5	10	10	224.2	97.0	10	10	176.5	131.5	10	10	898.0	898	10	10	133.1	150.5	10	10	5.9	5.0	1	1	3	3	10	10	118.5	88.0	10	10	249.0	209.5
AT6abc	10	10	58.8	33.5	10	10	349.6	324.5	9	9	297.8	103.0	10	10	57.4	57	10	10	1232.0	1232	10	10	123.6	119.0	10	10	6.4	5.0					10	10	61.3	68.0	10	10	65.6	47.0
NN	10	–	38.6	27.5	5	5	133.4	108.0	9	9	386.0	276.0	10	10	47.2	47.0					10	10	122.8	97.5	10	10	2.0	2.0	10	0	26.8	22.0					10	10	53.0	39.0	10	10	366.9	236.0
NN$_{\delta=0.04}$					0	0	–	–	0	0	–	–									6	–	963.7	1019.0									4	4	591.0	690.0	0	–	–	–
NNx					0	0	–	–	0	0	–	–	0	-	-	-									10	10	192.3	165.5	9	0	1442.7	128.0	0	0	–	–	0	–	–	–
CC1	10	10	10.4	9.5	10	10	26.6	17.5	10	10	104.7	86.5	10	10	36.1	36.0	10	10	396.0	396	10	5	21.8	21.5	10	10	3.6	3.0					9	9	1.22	1	10	10	61.2	53.0	10	10	22.5	10.5
CC2	10	10	15.4	15.0	10	10	10.4	8.0	10	10	78.3	60.5	9	9	398.6	364	10	10	76.0	76	8	8	224.3	19.5	10	10	3.0	3.0									4	4	109.3	44.0	10	10	20.2	19.5
CC3	10	10	77.9	54.5	10	10	41.6	32.5	10	10	185.5	179.0	10	10	14.9	10	10	10	122.0	122	10	4	36.1	11.5	10	10	5.7	5.5									10	10	35.2	31.0	10	10	62.1	42.0
CC4	0	0	–	–	0	0	–	–	3	3	747.0	603.0	0	-	-	-	4	0	1256.5	1198	7	1	680.6	732.0	10	10	176.7	148.0									0	0	–	–	0	–	–	–
CC5	10	10	28.5	14.5	10	10	16.1	10.5	10	10	103.1	110.5	10	10	78.2	106					10	10	88.7	25.5	10	10	48.2	10.5									10	9	55.7	53.0	10	10	42.3	41.0
CCx	7	7	338.1	300.0	9	9	337.2	183.0	9	9	149.1	96.0	10	10	448.1	390.5					10	4	228.0	189.5	10	10	110.5	62.0									2	–	261.5	261.0	10	10	232.2	230.0
F16									10	–	155.6	164.0	10	-	70.5	67									10	-	1.0	1.0	10	0	21	19					10	–	111.5	97.0	10	–	123.0	125.0
SC					0	0	–	–	0	0	–	–	0	-	-	-	0	0	–	–					10	10	45.1	29.5									0	0	–	–	0	–	–	–
PM	8	–	571.8	443.5	9	–	565.9	510.0	10	–	139.6	120.5	10	-	37.3	34	0	0	–	–									0	0	–	–					4	–	83.3	52.0	10	–	201	201

Table 6: Results for piecewise continuous 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**			**EXAM-Net**			**FalCAuN**			**ForeSee**			**FReaK**			**Moonlight**			**NNFal**			**OD**			**Ψ-TaLiRo**		
Approach:				ARX-2																										ConBO-LS		
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*	✓	$\overline{S}$	$\widetilde{S}$	*FR*	✓	$\overline{S}$	$\widetilde{S}$
AT1	0	0	–	–	0	0	–	–	0	0	–	–	10	10	150.4	98	10	10	387.4	429.0	10	10	4.5	3.5					2	2	1.5	1.5	0	0	–	–	5	5	1131.7	1226.0
AT2	10	10	18.8	13.5	10	10	15.1	11.5	10	10	62.3	54.0	10	10	22.7	31	9	9	196.6	38.5	10	10	2.2	2.0					10	10	18.5	13.0	10	10	13.7	10.0
AT51	10	10	20.5	16.5	1	1	1371.0	1371.0	10	10	106.0	87.5	10	10	8.5	4	10	10	15.0	14.5	10	10	17.7	11.5					10	10	11.6	10.0	10	10	10.1	8.5
AT52	10	10	74.1	65.0	10	10	4.4	3.0	10	10	19.0	7.0	10	10	10.3	7.5	10	10	65.7	35.0	10	10	5.2	2.0					10	10	8.6	6.0	10	10	67.8	62.0
AT53	10	10	1.5	1.0	10	10	4.4	3.5	10	10	2.2	1.0	10	10	1.5	1	10	10	4.3	3.5	10	10	3.9	2.0					10	10	2.2	2.0	10	10	4.1	3.0
AT54	10	10	47.9	42.0	6	6	571.5	575.0	10	10	138.9	23.5	10	10	89.6	12.5	10	10	60.3	29.5	10	10	103	83					10	10	18.4	10.0	10	10	37.4	23.0
AT6a	10	10	156.6	138.0	8	8	271.4	138.0	10	10	245.5	209.0	10	10	61.1	59.5	10	10	115.2	96.5	10	10	24.6	15.5					10	10	64.3	61.0	10	10	304.5	231.5
AT6b	10	10	472.2	588.0	6	6	536.8	527.0	9	9	279.3	248.0	10	10	119.4	108	10	10	253.9	274.5	10	10	17.4	11.0					10	10	122.0	94.0	6	6	782.2	538.0
AT6c	10	10	326.8	176.0	8	8	643.8	768.5	10	10	194.5	127.5	10	10	176.5	131.5	10	10	133.1	150.5	10	10	15.5	9.5	1	1	3	3	10	10	118.5	88.0	9	9	472.1	369.0
AT6abc	10	10	149.0	125.5	8	8	505.2	446.5	10	10	234.4	197.5	10	10	57.4	57	10	10	123.6	119.0	10	10	13.1	8.5					10	10	61.3	68.0	10	10	239.2	213.5
AFC27	0	–	–	–	9	9	56.2	6.0	8	7	137.8	109.5	10	8	235.8	34.5					10	10	12.0	10.0					10	10	391.2	233.0	10	10	101.8	101.0
AFC29	10	–	25.1	19.0	10	10	3.9	2.0	10	10	18.6	12.0	10	10	8.5	6.5					10	10	3.1	2.0					10	10	13.0	9.0	10	10	9.1	7.5
AFC33	0	–	–	–	0	0	–	–	0	0	–	–	0	-	-	-													0	0	–	–	0	–	–	–
NN	10	277.2	158.5	10	10	119.2	91.0	10	10	49.7	47.5	10	10	47.2	47.0		10	10	122.8	97.5	10	10	37.2	25.0	10	0	48.0	32.0					10	10	53.0	39.0	10	10	75.9	83.0
$NN_{\delta=0.04}$					1	1	55.0	55.0	2	2	503.5	503.5					6	–	963.7	1019.0	10	10	534.0	472.0									4	4	591.0	690.0	0	–	–	–
NNx	8	457.1	380.5	7	7	11.4	6.0	9	9	182.6	178.0	0	-	-	-						10	10	165.3	65.5	0	0	-	-					0	0	–	–	10	10	143.7	135.0
CC1	10	10	16.4	9.5	10	10	11.3	11.5	10	10	60.5	30.0	10	10	36.1	36.0	10	5	21.8	21.5	10	10	3.0	2.0					9	9	1.22	1	10	10	61.2	53.0	10	10	18.0	10.5
CC2	10	10	12.4	13.0	10	10	10.5	10.0	10	10	94.4	106.5	9	9	398.6	364	8	8	224.3	19.5	10	10	3.3	3.0									4	4	109.3	44.0	10	10	14.8	9.0
CC3	10	10	19.6	21.0	10	10	18.8	12.5	10	10	119.2	95.5	10	10	14.9	10	10	4	36.1	11.5	10	10	2.6	2.0									10	10	35.2	31.0	10	10	11.4	10.5
CC4	0	0	–	–	0	0	–	–	2	2	514.0	514.0	0	-	-	-	7	1	680.6	732.0	10	10	1349.9	1201.0									0	0	–	–	1	1	1387.3	1500.0
CC5	10	10	37.4	22.0	10	10	29.1	22.5	9	9	112.0	91.0	10	10	78.2	106	10	10	88.7	25.5	10	10	47.5	34.0									10	9	55.7	53.0	10	10	32.2	36.5
CCx	6	6	396.7	284.5	9	9	610.4	465.0	5	5	86.4	84.0	10	10	448.1	390.5	10	4	228.0	189.5	7	7	1723.6	938.0									2	–	261.5	261.0	10	10	244.2	225.0
SC					0	0	–	–	0	0	–	–	0	-	-	-																	0	0	–	–	0	–	–	–
PM	6	–	575.8	617.0	6	–	633.5	567.5	10	–	117.1	103.0	10	-	37.3	34									0	0	-	-					4	–	83.3	52.0	10	–	201	201

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 [38] 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 3.10. The results are shown in Table 7 for both instances. We refrain from an in-depth analysis of these results.

Table 7: 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*: 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 successful participation of new tools (EXAMNET, Moonlight [36], and FReaK [5]) 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 5 and 6 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 Federico Formica were pivotal in enabling the validation of the results also this year: They had to solve many technical challenges and provided 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 [8]), Test-Comp (which had independent coverage evaluation from the start in 2019 [9]), and many other competitions (for an overview, see [6]).

We have several items on our agenda for the following year's competition. We would like to precisely formalize the semantics of the ratio between the simulation time and the total falsification time (column "R") from Tables 5 and 6 and reintroduce it next year. We plan to extend the support of our validation platform for the pacemaker (PM) and the Aircraft Ground Collision Avoidance System (F16) benchmarks. We plan 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. We plan to employ the automatic repeatability evaluation platform provided by the organizers of the ARCH competition. Finally, we would like to stimulate and expand the competition rules related to the tools supporting probabilistic guarantees.

Acknowledgments We thank Gidon Ernst for the help with the validation of the results and the organizers of the ARCH Workshop 2024 for hosting this competition and providing a supportive and friendly environment.

The experiments for ARIsTEO and ATheNA were in part enabled by the support provided by Compute Ontario (computeontario.ca) and the Digital Research Alliance of Canada (alliancecan.ca). C. Menghi is supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU, by the European Union - Next Generation EU. "Sustainable Mobility Center (Centro Nazionale per la Mobilità Sostenibile - CNMS)", M4C2 - Investment 1.4, Project Code CN_00000023, and by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU. L. Nenzi is supported by MUR PRIN project 20228FT78M DREAM (modular software design to reduce uncertainty in ethics-based cyber-physical systems), Italy. 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, I. Porres, and V. Soloviev 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. E. Visconti is supported by the Austrian Science Fund (FWF) for the project "High-dimensional statistical learning: New methods to advance economic and sustainability policies" (ZK 35), jointly carried out by the University of Klagenfurt, the University of Salzburg, TU Wien, and the Austrian Institute of Economic Research (WIFO). 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] M. Althoff. An introduction to CORA 2015. In *Proc. of the International Workshop on Applied Verification for Continuous and Hybrid Systems*, pages 120–151, 2015.

[2] 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.

[3] 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.

[4] 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.

[5] Stanley Bak, Sergiy Bogomolov, Abdelrahman Hekal, Niklas Kochdumper, Ethan Lew, Andrew Mata, and Amir Rahmati. Falsification using reachability of surrogate koopman models. In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*, pages 1–13, 2024.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Alexandre Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification: 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings 22*, pages 167–170. Springer, 2010.

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

[15] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[26] 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.

[27] 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.

[28] 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.

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

[30] Atanu Kundu, Sauvik Gon, and Rajarshi Ray. Data-driven falsification of cyber-physical systems. In *Proceedings of the 17th Innovations in Software Engineering Conference*, pages 1–5, 2024.

[31] E. Lew and et al. Autokoopman: A toolbox for automated system identification via koopman operator linearization. In *Proc. of the International Symposium on Automated Technology for Verification and Analysis*, pages 237–250, 2023.

[32] 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.

[33] Claudio Menghi, Paolo Arcaini, Walstan Baptista, Gidon Ernst, Georgios Fainekos, Federico Formica, Sauvik Gon, Tanmay Khandait, Atanu Kundu, Giulia Pedrielli, Jarkko Peltomäki, Ivan Porres, Rajarshi Ray, Masaki Waga, and Zhenya Zhang. Arch-comp 2023 category report: Falsification. In *10th International Workshop on Applied Verification of Continuous and Hybrid Systems. ARCH23*, volume 96, pages 151–169, 2023.

[34] 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.

[35] L. Nenzi, E. Bartocci, L. Bortolussi, and M. Loreti. A Logic for Monitoring Dynamic Networks of Spatially-distributed Cyber-Physical Systems. *Logical Methods in Computer Science*, Volume 18, Issue 1, January 2022.

[36] Laura Nenzi, Ezio Bartocci, Luca Bortolussi, Simone Silvetti, and Michele Loreti. Moonlight: a lightweight tool for monitoring spatio-temporal properties. *International Journal on Software Tools for Technology Transfer*, 25(4):503–517, Aug 2023.

[37] 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.

[38] Giulia Pedrielli, Tanmay Khandait, Yumeng Cao, 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. *IEEE Transactions on Automation Science and Engineering*, pages 1–22, 2023.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer, 2015.

[44] STGEM. https://gitlab.abo.fi/stc/stgem, 06 2024 [Online].

[45] 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.

[46] 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.

[47] 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.

[48] 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.