EPiC Series in Computing
Volume 61, 2019, Pages 129–140
ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems

ARCH-COMP 2019 Category Report: Falsification*

Gidon Ernst[1], Paolo Arcaini[2], Alexandre Donze[3]
Georgios Fainekos[4], Logan Mathesen[4], Giulia Pedrielli[4],
Shakiba Yaghoubi[4], Yoriyuki Yamagata[5], and Zhenya Zhang[2]

[1] Ludwig-Maximilians-University (LMU), Munich, Germany
gidon.ernst@sosy.ifi.lmu.de
[2] National Institute of Informatics (NII), Tokyo, Japan
{arcaini,zhangzy}@nii.ac.jp
[3] Decyphir SAS, Moirans, France
alex@decyphir.com
[4] Arizona State University (ASU), Tempe, USA
{fainekos,lmathese,gpedriel,syaghoub}@asu.edu
[5] National Institute of Advanced Industrial Science and Technology (AIST), Osaka, Japan
yoriyuki.yamagata@aist.go.jp

Abstract

This report presents the results from the 2019 friendly competition in the ARCH workshop for the falsification of temporal logic specifications over Cyber-Physical Systems. We describe the organization of the competition and how it differs from previous years. We give background on the participating teams and tools and discuss the selected benchmarks and results. The benchmarks are available on the ARCH website[1], as well as in the competition's `gitlab` repository[2]. The main outcome of the 2019 competition is a common benchmark repository, and an initial base-line for falsification, with results from multiple tools, which will facilitate comparisons and tracking of the state-of-the-art in falsification in the future.

1 Introduction

The friendly competition of the ARCH workshop is running yearly since 2014. The goal is to compare the state-of-the-art of tools for testing and verification of hybrid systems. The competition is organized in several categories, with different specifications (computing reachable regions, checking temporal properties) and varying dynamics in the system models (such as linear/non-linear and hybrid).

In the falsification category, benchmarks typically consist of executable Matlab code or Simulink/Stateflow models, each associated with a set of requirements in temporal logic with

*The falsification category was coordinated by the first author. The remaining authors represent all participants and they are listed alphabetically.

[1] https://cps-vo.org/group/ARCH/FriendlyCompetition

[2] https://gitlab.com/goranf/ARCH-COMP

time bounds, encoded in MTL [13] or STL [14]. The task is to find initial conditions and time-varying inputs subject to given constraints that steer the system into a violation of the respective requirement. This search is typically guided using well-established robustness metrics [8] that give a quantitative account of how close a given input is to violating a requirement. Using such metrics as score functions permits one to employ standard optimization techniques to find falsifying inputs. Recent results in falsification have produced a variety of techniques, mature tools, and practical applications, see [3] for an overview. Due to the complexity and unclear semantics of Matlab and Simulink models, many previous techniques are entirely black-box and just observe the input/output behavior of the system via simulations, but grey-box approaches have been developed recently [18, 1] to take some knowledge on the internals of the system into deliberation. This year's falsification competition featured more tools and participating teams in comparison to previous years, prompting a few changes to the organization with respect to benchmark solicitation and specification, as well as the validation of the results (Sec 2).

The participating tools 2019 were S-TaLiRo [2], Breach [6], FalStar [19, 7], and falsify [1] in different configurations (Sec 3).

There were 6 benchmark models overall with individual requirements, taken from previous competitions and from the literature (Sec 4): Automatic Transmission (AT), Fuel Control of an Automotive Powertrain (AFC), Neural-network Controller (NN), Wind Turbine (WT), Chasing cars (CC), Aircraft Ground Collision Avoidance system (F16), and Steam Condenser with Recurrent Neural Network Controller (SC).

As expected, the results (Sec 5) show that tools perform better on some benchmarks and worse on others, and that different tools have different strengths. Notable success is achieved by the tool falsify [1] , which often performs best by far, sometimes requiring just a single simulation to find a falsifying input thanks to its incremental strategy.

2 Organization

Several major changes to the format of the competitions were made in 2019.

Benchmark Solicitation. For a more comprehensive assessment of the capabilities of tools, the benchmark pool was extended to cover several models with multiple requirements. Source of these benchmarks were known models from the literature that had served in evaluations of new methods and at previous years of the competition.

Two changes and the presence of more teams than in the years before (see section 3) prompted a more systematic organization. To this end, benchmarks were collected in a fork of the `gitlab` repository in which the reproducibility packages are maintained. Participating teams had access to that repository to contribute and download benchmarks models.

Similarly, descriptions on the use of the models and a precise, informal characterization of the respective settings was communicated in this way, too.

Input Specifications. There was some discussion on the format and strictness of the input characterizations (for time varying inputs). Two options were proposed and it was agreed to run separate evaluations for each of the two options (called instance 1 and instance 2 in the following).

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

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

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

Validation and Repeatability. To prevent implementation bugs and specification errors and to check for differences in implementations of robustness evaluations, it was planned early on to validate all results. The group's organizer provided a Matlab-based interface for validation, implemented by FalStar. Moreover, the organizers of the ARCH workshop decided that all groups in the competition should submit Docker-based packages for repeatability evaluation, but in the end it was infeasible to package an entire Matlab installation including the necessary toolboxes.

3 Participants

S-TaLiRo. S-TaLiRo [2] is a Matlab toolbox for monitoring and test case generation against system specifications presented in STL. The test cases are automatically generated using optimization techniques guided by formal requirements in STL in order to find falsifying systems behaviors. The tool has different optimization algorithms. Specifically, in this competition, the stochastic optimization with adaptive restarts (SOAR) [15] framework is used for all the benchmarks except for choosing instance 1 type inputs in Steam Condenser model. In that benchmark Simulated annealing global search was combined by a local optimal control based search [18]. S-TaLiRo is publicly available on-line under General Public License (GPL) [3].

Breach. Breach [6] is a Matlab toolbox for test case generation, formal specification monitoring and optimization-based falsification and mining of requirements for hybrid dynamical systems. A particular emphasis is put on modularity and flexibility of inputs generation, requirement evaluation and optimization strategy. For this work, the approach has been to ensure that each benchmark was properly implemented and a default, relatively basic falsification strategy has been applied. The idea was to perform a first systematic investigation of the proposed problems, and then to provide a base to work on for future editions of the competition to test a

[3] https://sites.google.com/a/asu.edu/s-taliro

larger variety on approaches on the most challenging instances. Breach is available under BSD license[4].

FalStar. FalStar is an experimental prototype of a falsification tool that explores the idea to construct falsifying inputs incrementally in time, thereby exploiting potential time-causal dependencies in the problem. It implements several algorithms, of which two were used in the competition: A two layered framework combining Monte-Carlo tree search (MCTS) with stochastic optimization [19], and a probabilistic algorithm [7] that adapts to the difficulty of the problem dubbed adaptive Las-Vegas tree search (aLVTS). The code is publicly available under the BSD license.[5]

falsify. falsify is an experimental program which solves falsification problems of safety properties by reinforcement learning [1]. falsify uses a *grey-box* method, that is, it learns system behavior by observing system outputs during simulation. falsify is currently implemented by a deep reinforcement learning algorithm *Asynchronous Advantage Actor-Critic* (A3C) [16].

4 Benchmarks

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 [10].

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 (where $\circ\, \phi \equiv \Diamond_{[0.001,0.1]}\, \phi$) are specific versions of those in [10] where the parameters have been chose to be somewhat difficult.

Benchmark	STL formula
AT1	$\Box_{[0,20]} v < 120$
AT2	$\Box_{[0,10]} \omega < 4750$
AT51	$\Box_{[0,30]}((\neg g1 \wedge \circ\, g1) \rightarrow \circ\, \Box_{[0,2.5]} g1)$
AT52	$\Box_{[0,30]}((\neg g2 \wedge \circ\, g2) \rightarrow \circ\, \Box_{[0,2.5]} g2)$
AT53	$\Box_{[0,30]}((\neg g3 \wedge \circ\, g3) \rightarrow \circ\, \Box_{[0,2.5]} g3)$
AT54	$\Box_{[0,30]}((\neg g4 \wedge \circ\, g4) \rightarrow \circ\, \Box_{[0,2.5]} g4)$
AT6a	$(\Box_{[0,30]} \omega < 3000) \rightarrow (\Box_{[0,4]} v < 35)$
AT6b	$(\Box_{[0,30]} \omega < 3000) \rightarrow (\Box_{[0,8]} v < 50)$
AT6c	$(\Box_{[0,30]} \omega < 3000) \rightarrow (\Box_{[0,20]} v < 65)$

Fuel Control of an Automotive Powertrain (AFC). The model is described in [12] and has been used in two previous instalments of this competition [4, 5]. The specific limits used in the requirements are chosen such that falsification is possible but reasonably hard.

The constrained input signal (instance 2) fixes the throttle θ to be piecewise constant with 10 uniform segments over a time horizon of 0 with two modes (normal and power corresponding to feedback and feedforward control), and the engine speed ω to be constant with $900 \leq \omega < 1100$

[4]`https://github.com/decyphir/breach`
[5]`https://github.com/ERATOMMSD/falstar`

to capture the input profile outlined in [12] and to match the previous competitions. For this reason, we do not consider the unconstrained (instance 1) input specification. Faults are disabled (e.g. by setting `fault_time` > 50).

Benchmark	STL formula	Input Constraint
AFC27	$\square_{[11,50]}((\mathit{rise} \vee \mathit{fall}) \rightarrow (\square_{[1,5]}\|\mu\| < \beta))$	$0 \leq \theta < 61.2$ (normal)
AFC29	$\square_{[11,50]}\|\mu\| < \gamma$	$0 \leq \theta < 61.2$ (normal)
AFC33	$\square_{[11,50]}\|\mu\| < \gamma$	$61.2 \leq \theta \geq 81.2$ (power)

where

$$\beta = 0.008 \qquad \gamma = 0.007$$
$$\mathit{rise} = (\theta < 8.8) \wedge (\diamond_{[0,0.05]}(\theta > 40.0))$$
$$\mathit{fall} = (\theta > 40.0) \wedge (\diamond_{[0,0.05]}(\theta < 8.8))$$

Note that the time interval starts at 11, which is one time unit after the feedback mode is enabled at time 10.

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.[6] 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 $1 \leq \mathit{Ref}$ and $\mathit{Ref} \leq 3$. It outputs the current position of the levitating magnet Pos. The input specification for instance 1 requires discontinuities to be at least 3 time units apart, whereas instance 2 specifies an input signal with exactly three constant segments. The time horizon for the problem is 40. The requirement ensures that after changes to the reference, the actual position eventually stabilizes around that value with small error.

Benchmark	STL formula
NN	$\square_{[1,37]}\big(\|Pos - \mathit{Ref}\| > \alpha + \beta\|\mathit{Ref}\| \rightarrow \diamond_{[0,2]}\square_{[0,1]}\neg(\alpha + \beta\|\mathit{Ref}\| \leq \|Pos - \mathit{Ref}\|)\big)$

where $\alpha = 0.005$ and $\beta = 0.03$.

Wind Turbine (WT). The model is a simplified wind turbine model proposed in [17]. The input of the system is wind speed v and the outputs are blade pitch angle θ, generator torque $M_{g,d}$, rotor speed Ω and demanded blade pitch angle θ_d. The wind speed is constrained by $8.0 \leq v \leq 16.0$. Instance 1 allows any piece-wise continuous inputs, while instance 2 constrains inputs to piece-wise constant signals whose control points which are evenly spaced each 5 seconds. The model is relatively large. Further, the time horizon is long (630) compared to other benchmarks.

Benchmark	STL formula
WT1	$\square_{[30,630]}\theta \leq 14.2$
WT2	$\square_{[30,630]}21000 \leq M_{g,d} \leq 47500$
WT3	$\square_{[30,630]}\Omega \leq 14.3$
WT4	$\square_{[30,630]}\diamond_{[0,5]}\|\theta - \theta_d\| \leq 1.6$

Chasing cars (CC). The model is derived from Hu et al. [11] which presents a simple model of an automatic chasing car. Chasing cars (CC) model consists of five cars, in which the first

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

car is driven by inputs (*throttle* and *brake*), and other four are driven by Hu et al.'s algorithm. The output of the system is the location of five cars y_1, y_2, y_3, y_4, y_5. The properties to be falsified are constructed artificially, to investigate the impact of complexity of the formulas to falsification. The input specifications for instance 1 allows any piecewise continuous signals while the input specification for instance 2 constraints inputs to piecewise constant signals with control points for each 5 seconds, i.e., 20 segments

Benchmark	STL formula
CC1	$\square_{[0,100]} y_5 - y_4 \leq 40$
CC2	$\square_{[0,70]} \diamond_{[0,30]} y_5 - y_4 \geq 15$
CC3	$\square_{[0,80]}((\square_{[0,20]} y_2 - y_1 \leq 20) \vee (\diamond_{[0,20]} y_5 - y_4 \geq 40))$
CC4	$\square_{[0,65]} \diamond_{[0,30]} \square_{[0,20]} y_5 - y_4 \geq 8$
CC5	$\square_{[0,72]} \diamond_{[0,8]}((\square_{[0,5]} y_2 - y_1 \geq 9) \rightarrow (\square_{[5,20]} y_5 - y_4 \geq 9))$

Aircraft Ground Collision Avoidance system (F16). The model has been derived from [9]. The F16 aircraft and its inner-loop controller for Ground Collision avoidance have been modeled using 16 continuous variables with piece-wise nonlinear differential equations. Autonomous maneuvers are performed in an outer-loop controller that uses a finite-state machine with guards involving the continuous variables. The system is required to always avoid hitting the ground during its maneuver starting from all the initial conditions for roll, pitch, and yaw in the range $[0.2\pi, 0.2833\pi] \times [-0.5\pi, -0.54\pi] \times [0.25\pi, 0.375\pi]$. 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. This requirement can be captured using the following formula

Benchmark	STL formula
F16	$\square_{[0,15]} \mathit{altitude} > 0$

Steam condenser with Recurrent Neural Network Controller (SC). The model is presented in [18]. It is a dynamic model of an steam condenser based on energy balance and cooling water mass balance controlled with a Recurrent Neural network in feedback. The time horizon for the problem is 35 seconds. The input to the system can vary in the range $[3.99, 4.01]$. For instance 2, the input signal should be piecewise constant with 20 evenly spaced segments. The pressure output should satisfy the following requirement:

Benchmark	STL formula
SC	$\square_{[30,35]}(87 \leq \mathit{pressure} \wedge \mathit{pressure} \leq 87.5)$

5 Evaluation

Falsification tools were instructed to run each individual requirement 50 times, to account for the stochastic nature of most algorithms. We report the falsification rate, i.e., the number of trials where a falsifying input was found, as well as the median and mean of the number of simulations required to find such input (not including the unsuccessful runs in the aggregate). The cut-off for the number of simulations per trial was 300.

The results for unconstrained piecewise-continuous input signals (instance 1) are shown in Table 1. They depend on the choices for the search space, which we briefly discuss for each participating tool:

Breach. For most benchmarks (exceptions detailed below), a piecewise constant signal generation was used with fixed step size. For all instances, the optimization strategy used is the default global Nelder Mead (GNM) approach with a custom configuration for the competition, resulting in the following three phases behavior:

- Phase 1.: at most $n_{\text{corners}} = 64$ corner samples are tested, i.e., inputs for which control points take only extreme values;
- Phase 2.: $n_{\text{quasi-rand}} = 100 - n_{\text{corners}}$ quasi-random samples from the Halton sequence with varying start points determined by a random seed are tested;
- Phase 3.: the robustness results from phase 1 and 2 are sorted and Nelder Mead optimization is run from the most promising samples.

Note that as a result of this approach, whenever a falsifying input is consistently found with less than 100 simulations, it indicates that the problem is likely trivially falsifiable with extreme inputs or a quick stochastic exploration of the search space. The following settings were chosen for input generation for each benchmark:

- AT: throttle input and brake inputs were configured with respectively 3 and 2 control points at variable times;
- NN: input was piecewise constant with 3 control points regularly spaced;
- WT: spline interpolation with control points regurlary spaced by 5s and saturation to domain [8;16] (same for instance 2);
- CC: same as instance 2, i.e., piecewise constant input with control points regurlary spaced by 5s;
- SC: same as instance 2, i.e., piecewise constant input with control points regurlary spaced by 1.75s;

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

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

FalStar (MCTS). The search space included piecewise constant inputs. For all the benchmarks and for all the specifications, the number of control points was computed according to the simulation time that was divided by a fixed interval of 5 time units (e.g., a simulation time of 50 has 10 control points).

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

Table 1: Results for piecewise continuous input signals (instance 1). *FR*: falsification rate wrt. number of trials, $\overline{S}$ and $\widetilde{S}$: mean resp. median number of simulations over successful trials ("–" if *FR* is zero).

Tool:	**S-TaLiRo**			**Breach**			**FalStar**						**falsify**		
Configuration:	SOAR			GNM			MCTS			aLVTS			A3C		
Benchmark	*FR*	$\overline{S}$	$\widetilde{S}$	*FR*	$\overline{S}$	$\widetilde{S}$	*FR*	$\overline{S}$	$\widetilde{S}$	*FR*	$\overline{S}$	$\widetilde{S}$	*FR*	$\overline{S}$	$\widetilde{S}$
AT1	50/50	118.84	116	50/50	11	11	0/50	–	–	50/50	33.04	30	17/50	224.53	223.0
AT2	50/50	23.88	19.5	50/50	2	2	50/50	21.68	5	50/50	4.32	3	50/50	22.10	10.0
AT51	50/50	26.72	22	41/50	74.56	67	50/50	39.10	43	50/50	69.48	56	50/50	1.00	1.0
AT52	50/50	4.06	3	49/50	72	67	50/50	2.68	2	26/50	125.35	137	50/50	1.00	1.0
AT53	50/50	3.44	3	49/50	74.51	73	50/50	2.52	3	50/50	70.82	68	50/50	1.00	1.0
AT54	50/50	10.46	2	21/50	84.86	85	50/50	26.56	10	50/50	71.10	52	50/50	1.00	1.0
AT6a	49/50	78.41	40	50/50	97.92	97.5	49/50	93.65	80	50/50	76.06	70	(not a safety property)		
AT6b	33/50	132.64	128	49/50	112.9	118	29/50	166.90	173	50/50	82.38	75	(not a safety property)		
AT6c	47/50	61.34	38	50/50	94.1	89	6/50	105.92	125	0/50	–	–	(not a safety property)		
NN	50/50	26.72	22	48/50	96.29	101.5	50/50	47.98	40	36/50	122.80	106	50/50	1.00	1.0
NN($\beta = 0.04$)	4/50	193	222.5												
WT1	50/50	91	91	50/50	3	3	50/50	4.00	4	(unsupported)			37/50	47.68	7.0
WT2	50/50	32.62	30	50/50	3	3	50/50	1.00	1				46/50	8.04	2.0
WT3	50/50	44.12	60	50/50	3	3	50/50	2.00	2				50/50	2.48	1.0
WT4	50/50	3.32	2	50/50	30	30	50/50	2.00	2				50/50	4.90	4.0
CC1	50/50	9.38	7	50/50	3	3	50/50	15.00	15	50/50	4.06	2	47/50	51.26	17.0
CC2	50/50	6	4	50/50	1	1	50/50	26.00	26	50/50	4.02	2	37/50	24.24	4.0
CC3	50/50	19.94	5	50/50	3	3	50/50	14.40	17	50/50	6.86	5	46/50	35.41	8.5
CC4	20/50	188	179.5	0/50	–	–	0/50	–	–	2/50	52.00	60	1/50	26.00	26.0
CC5	50/50	42.94	36.5	49/50	26.08	19	50/50	132.00	140	46/50	91.19	79	31/50	29.65	26.0
F16	7/50	127.57	94	1/50	297	297	(unsupported)			0/50†	–	–	(unsupported)		
SC	50/50 ⋆	62.26	55.5	0/50	–	–	0/50	–	–	0/50	–	–	0/50	–	–

F16 †: FalStar/aLVTS currently samples initial conditions uniformly at random
SC ⋆: The S-TaLiRo results for this benchmark are yielded by Simulated annealing assisted with gradient based search (See [18]).

are allowed (resp. upto 3 for NN). In this configuration FalStar benefits from a low number of control points and is more likely to try inputs with fewer control points first. For the AT benchmarks it was clear beforehand that this choice suffices to falsify all benchmarks, and the setting was then kept for the remaining experiments.

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

Common Settings. For a better comparison of the performance of the tools, a common ground is piecewise constant input signals (instance 2) with a concrete specification of the number of discontinuities allowed. The corresponding results are shown in Table 2.

The implied intention was that the search space is actually large enough so that in principle input signals are generated that exhibit that number of variability. For this reason, for example, FalStar (aLVTS) was instructed not to generate input signals with fewer control points (even though it would likely have improved the results). FalStar (MCTS) similarly takes the number of control points as input.

Table 2: Results for constrained input signals/instance 2. FR: falsification rate, $\overline{S}$: mean number of simulations, $\widetilde{S}$: median number of simulations.

Tool:		**S-TaLiRo**			**Breach**				**FalStar**					**falsify**	
Configuration:		SOAR			GNM			MCTS			aLVTS			A3C	
Benchmark	*FR*	$\overline{S}$	$\widetilde{S}$	*FR*	$\overline{S}$	$\widetilde{S}$	*FR*	$\overline{S}$	$\widetilde{S}$	*FR*	$\overline{S}$	$\widetilde{S}$	*FR*	$\overline{S}$	$\widetilde{S}$
AT1	50/50	170.32	171	0/50	–	–	0/50	–	–	50/50	33.04	30	39/50	125.79	110
AT2	50/50	16.82	9	50/50	2	2	50/50	21.68	5	50/50	4.32	3	48/50	18.96	7.5
AT51	50/50	12.64	11	50/50	7	7	50/50	39.10	43	50/50	69.48	56	50/50	1.00	1.0
AT52	49/50	17.57	4	50/50	3	3	50/50	2.68	2	26/50	125.35	137	50/50	1.00	1.0
AT53	50/50	3.38	3	50/50	3	3	50/50	2.52	3	50/50	70.82	68	50/50	1.00	1.0
AT54	50/50	24.15	16.5	50/50	3	3	50/50	26.56	10	50/50	71.10	52	50/50	1.00	1.0
AT6a	44/50	130.353	149.5	0/50	–	–	49/50	93.65	80	50/50	76.06	70		(not safety)	
AT6b	39/50	207.206	236	0/50	–	–	29/50	166.90	173	50/50	82.38	75		(not safety)	
AT6c	42/50	197.5	208.5	0/50	–	–	36/50	105.92	125	0/50	–	–		(not safety)	
AFC27	50/50	70.34	78.5	50/50	3	3	–	–	–	50/50	3.88	3	50/50	1.64	1.0
AFC29	50/50	13.54	10	50/50	3	3	–	–	–	50/50	1.18	1	50/50	1.00	1.0
AFC33	0/50	–	–	0/50	–	–	–	–	–	0/50	–	–	50/50	1.00	1.0
NN	49/50	67.96	48	50/50	6	6	50/50	177.44	183	26/50	176.96	197	50/50	1.00	1.0
NN(β=0.04)	3/50	127	74												
WT1	50/50	7.1	5	50/50	3	3	50/50	4	4		(unsupported)		49/50	8.57	2.0
WT2	50/50	1.02	1	50/50	3	3	50/50	1	1				50/50	2.76	2.0
WT3	50/50	1.04	1	50/50	3	3	50/50	2	2				50/50	2.04	1.0
WT4	50/50	12.02	9	50/50	30	30	50/50	2	2				50/50	3.68	3.0
CC1	50/50	67.8	91	50/50	5	5	50/50	15.00	15	50/50	7.30	6	50/50	23.48	7.0
CC2	48/50	114.48	105	50/50	1	1	50/50	26.00	26	50/50	15.88	9	46/50	14.41	4.0
CC3	50/50	22.36	13	50/50	5	5	50/50	14.40	17	4/50	207.50	229	44/50	13.52	2.5
CC4	0/50	–	–	0/50	–	–	0/50	–	–	0/50	–	–	9/50	120.44	168.0
CC5	50/50	74.88	48	16/50	84.69	79	50/50	132.00	140	39/50	117.94	103	32/50	37.22	8.5
SC	0/50	–	–	0/50	–	–	0/50	–	–	0/50	–	–	0/50	–	–

Discussion. The most successful outcome can be attributed to falsify, which can find a falsifying input signal with a single simulation only. This is remarkable as all sampling and tuning of the input is done on the fly with respect to an established prefix. Notably, most results for AT, AFC, NN, and WT models show this behavior. While the strategy works well enough and applies generally, there are a few instances where other strategies work better, e.g., the tendency of GNM and aLVTS to prefer extreme values for AT1, as well as the S-TaLiRo's algorithm used to solve the SC benchmark.

Another observation is that many benchmarks can be solved quite easily, leading to perfect falsification rates (FR) of 50/50. This suggests that the overall mass of falsifying inputs is large enough so that those can be found reliably with significantly less than 300 simulations, independently of whether sophisticated algorithms are used (S-TaLiRo, MCTS, falsify), or whether sampling is random (but not necessarily uniform, GNM, aLVTS). This can be seen as a confirmation of folklore knowledge that many falsification problems are actually not that hard to solve, and that conceptually simple methods can indeed work well. A clear take-away is that the set of benchmarks needs to be more challenging in the future.

There is not so much of a difference between the results for instance 1 and instance 2. The latter typically require more time to solve, as the parameter space is larger (e.g., the CC benchmark has many more control points in the second instance).

Due to the inhomogeneity of results we refrain from a further in-depth analysis of when and why exactly certain algorithms fare better than others. This is left for future work (cf. also the comment on benchmark classification below). Similarly, we do not attempt to derive an aggregate score to determine a "winner".

6 Conclusion and Outlook

Setting up the competition with the format described in Sec. 2 proved to be difficult and time-consuming.

The first issue is a general lack of a well-prepared benchmark suite for falsification of black-box systems. In comparison to the other tracks (and similar competitions such as SMT-COMP and SV-COMP), the ARCH group on falsification could previously not refer to an established repository of benchmarks. By collecting existing benchmarks and soliciting new ones in a single place now we have that for the first time.

Nevertheless, even though all participating tools interface with Matlab/Simulink, there are differences in how this is done, e.g., how input parameters are passed to the simulation engine. As an example, Breach and S-TaLiRo internally interpolate input signals to a high sample rate, whereas FalStar relies on Simulink for this and provides compressed input signals with a low sample rate. As a consequence, the result for FalStar differs depending on the interpolation setting of the input ports in the model, whereas that is not the case for the other tools. Another issue that had to be clarified was the input signal specification, as discussed in Sec. 2. Repeatability and cross-validation of results currently involves a lot of manual set-up due to lack of common standards for reporting and processing these. For future competitions, it would be much better, if such standards are established beforehand, so that the initial phase of the competition can focus on benchmark selection. In the end, the envisioned cross-checking of results happened not systematically, so the results cannot be trusted entirely, but at least a partial success in this regard could be achieved.

Running the benchmarks is computationally expensive, which is aggravated by the need to run many trials (50 here) to account for the stochastic nature of the algorithms. As a consequence, running all benchmarks for a given tools takes several machine-days. In combination with the discussions, this factor lead to a delayed availability of stable results, and a short time frame for validating the results. It would be great to use existing computing infrastructure such as StarExec[7] to run the benchmarks, but it is currently unclear whether this is feasible with respect to availability of sufficient Matlab licenses.

We hope that future competition installments will benefit from the now established set up, and that furthermore this benchmark repository will eventually serve as a base-line for experimental research in the falsification community. Moreover, with respect to such a base-line it will be easier to track the state-of-the-art and progress of the effectiveness and efficiency of falsification tools. Potential future work would be to include a classification of the difficulties of the individual benchmarks (and the falsification rate of uniform random sampling), so that it can be investigated which approach works well for which kind of problem.

Acknowledgments P. Arcaini and Z. Zhang are supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST. Funding Reference number: 10.13039/501100009024 ERATO. The report on falsify is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO). The ASU team (S-TaLiRo) was partially supported by NSF CNS 1350420, NSF CMMI 1829238, NSF IIP-1361926 and the NSF I/UCRC Center for Embedded Systems.

[7] https://www.starexec.org/starexec/public/about.jsp

References

[1] Takumi Akazaki, Shuang Liu, Yoriyuki Yamagata, Yihai Duan, and Jianye Hao. Falsification of cyber-physical systems using deep reinforcement learning. In *Formal Methods - 22nd International Symposium, FM 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 15-17, 2018, Proceedings*, pages 456–465, 2018.

[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] 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.

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

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

[6] Alexandre Donzé. Breach, A toolbox for verification and parameter synthesis of hybrid systems. In *International Conference on Computer Aided Verification (CAV 2010)*, LNCS, pages 167–170. Springer, 2010.

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

[8] 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.

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

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

[11] 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.

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

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

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

[15] Logan Mathesen and Giulia Pedrielli. Stochastic optimization with adaptive restart: A framework

for integrated local and global learning. *submitted for publication*, 2019.

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

[17] Simone Schuler, Fabiano Daher Adegas, and Adolfo Anta. Hybrid modelling of a wind turbine. In Goran Frehse and Matthias Althoff, editors, *ARCH16. 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems*, volume 43 of *EPiC Series in Computing*, pages 18–26. EasyChair, 2017.

[18] 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.

[19] Zhenya Zhang, Gidon Ernst, Sean Sedwards, Paolo Arcaini, and Ichiro Hasuo. Two-Layered Falsification of Hybrid Systems Guided by Monte Carlo Tree Search. *Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 37(11):2894–2905, 2018.