EPiC Series in Computing

Volume 74, 2020, Pages 140–152

ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)

ARCH-COMP 2020 Category Report: Falsification*

Gidon Ernst[1], Paolo Arcaini[2], Ismail Bennani[7], Alexandre Donze[3], Georgios Fainekos[4], Goran Frehse[8], Logan Mathesen[4], Claudio Menghi[6], Giulia Pedrielli[4], Marc Pouzet[7], Shakiba Yaghoubi[4], Yoriyuki Yamagata[5], and Zhenya Zhang[2]

[1] Ludwig-Maximilians-University (LMU), Munich, Germany
gidon.ernst@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
[6] University of Luxembourg, Luxembourg, Luxembourg
claudio.menghi@uni.lu
[7] École Normale Supérieure (ENS), Paris, France
{ismail.lahkim.bennani,marc.pouzet}@ens.fr
[8] ENSTA Paris, Palaiseau, France
goran.frehse@ensta-paris.fr

Abstract

This report presents the results from the 2020 friendly competition in the ARCH workshop for the falsification of temporal logic specifications over Cyber-Physical Systems. We briefly describe the competition settings, which have been inherited from the previous year, give background on the participating teams and tools and discuss the selected benchmarks. The benchmarks are available on the ARCH website[1], as well as in the competition's `gitlab` repository[2]. In comparison to 2019, we have two new participating tools with novel approaches, and the results show a clear improvement over previous performances on some benchmarks.

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

*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

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 time bounds, encoded in MTL [20] or STL [21]. 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 [13] 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, 6] 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 [27, 1, 26] to take some knowledge on the internals of the system into deliberation. This year's falsification competition featured two more tools and participating teams in comparison to the previous year [11]

The participating tools 2019 were S-TaLiRo [2], Breach [9], FalStar [28, 12], falsify [1], ARIsTEO [23], and `zlscheck` (based on Zélus [4]), in different configurations (Sec 2).

The format of the competitions was essentially that of 2019. It is based on a selection of benchmark models and requirements from the literature, for which falsifying input traces have to be found within a given maximum of simulations. The format of the competitions was essentially that of 2019. It is based on a selection of benchmark models and requirements from the literature, for which falsifying input traces have to be found within a given maximum of simulations.

The 6 benchmark were those from last year, each with individual requirements, taken from previous competitions and from the literature (Sec 3): 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). There were two different general settings for the parameterization of the search space, as described below. As part of the preparation for `zlscheck`, there are now variants of the model written in Zélus (see Sec 2).

The results (Sec 4) have been extended to include those of ARIsTEO and `zlscheck`, whereas the results for the remaining tools are those of 2019. As expected, the results show that tools perform better on some benchmarks and worse on others, and that different tools have different strengths. A characteristic shared with falsify is that the new contenders can find falsifying inputs "online" on a single trace.

2 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) [22] 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 [27]. S-TaLiRo is publicly available on-line under General Public License (GPL) [3].

Breach. Breach [9] 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 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 [28], and a probabilistic algorithm [12] 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) [24].

ARIsTEO. ARIsTEO [23] is a Matlab toolbox for test case generation against system specifications presented in STL and it is developed on the top of S-TaLiRo. ARIsTEO is designed to targeting a large and practically-important category of CPS models, known as *compute-intensive* CPS (CI-CPS) models, where a single simulation of the model may take hours to complete. ARIsTEO embeds black-box testing into an iterative approximation-refinement loop. At the start, some sampled inputs and outputs of the model under test are used to generate a surrogate model that is faster to execute and can be subjected to black-box testing. Any failure-revealing test identified for the surrogate model is checked on the original model. If spurious, the test results are used to refine the surrogate model to be tested again. Otherwise, the test reveals a valid failure. ARIsTEO is publicly available under General Public License (GPL).[6]

`zlscheck`. `zlscheck` is a tool for test case generation of programs written in Zélus[7] [4], a language reminiscent of the synchronous languages Lustre [14] and Scade [5] extended in order to express ODEs. For now `zlscheck` applies to the discrete-time subset of Zélus.
Properties are expressed as synchronous observers [15] with a quantitative semantics to solve the falsification problem as an optimization problem. `zlscheck` uses automatic differentiation to compute gradients of the robustness of a model w.r.t. some input parameters and uses

[3] https://sites.google.com/a/asu.edu/s-taliro
[4] https://github.com/decyphir/breach
[5] https://github.com/ERATOMMSD/falstar
[6] https://github.com/SNTSVV/ARIsTEO
[7] http://zelus.di.ens.fr/

gradient-based techniques to find a falsifying input. Additionally, it uses FADBADml[8], a tool for automatic differentiation which we ported from C++ to OCaml.
All the models of this competition have been rewritten manually in Zélus and are available along with the tool on github[9]. The Simulink's Integrator block is programmed in Zélus as a fixed-step forward euler scheme.
The counter-examples found by `zlscheck` were systematically validated on their corresponding Simulink models, the ones that did not pass validation were discarded from the final results.

3 Benchmark Definitions

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

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

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

A brief description of the benchmark models follows, the requirements are shown in Table 1.

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 [17].

Input specification: $0 \leq \mathit{throttle} \leq 100$ and $0 \leq \mathit{brake} \leq 325$ (both can be active at the same time). Constrained input signals (instance 2) permit discontinuities at most every 5 time units. Requirements are specific versions of those in [17] where the parameters have been chose to be somewhat difficult.

[8] https://fadbadml-dev.github.io/FADBADml/
[9] https://github.com/ismailbennani/zlscheck

Table 1: Requirement formulas for the benchmarks

Benchmark	STL formula	Input Constraint
AT1	$\square_{[0,20]} v < 120$	
AT2	$\square_{[0,10]} \omega < 4750$	
AT51	$\square_{[0,30]}((\neg g1 \wedge \circ\, g1) \rightarrow \circ\, \square_{[0,2.5]} g1)$	
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	$(\square_{[0,30]} \omega < 3000) \rightarrow (\square_{[0,20]} v < 65)$	
	where $\circ\, \phi \equiv \Diamond_{[0.001,0.1]}\ \phi$	
AFC27	$\square_{[11,50]}((rise \vee fall) \rightarrow (\square_{[1,5]} \lvert\mu\rvert < \beta))$	$0 \leq \theta < 61.2$
AFC29	$\square_{[11,50]} \lvert\mu\rvert < \gamma$	$0 \leq \theta < 61.2$
AFC33	$\square_{[11,50]} \lvert\mu\rvert < \gamma$	$61.2 \leq \theta \geq 81.2$
	where $\beta = 0.008$, $\gamma = 0.007$ $rise = (\theta < 8.8) \wedge (\Diamond_{[0,0.05]}(\theta > 40.0))$ $fall = (\theta > 40.0) \wedge (\Diamond_{[0,0.05]}(\theta < 8.8))$	
NN	$\square_{[1,37]}\big(\lvert Pos - Ref\rvert > \alpha + \beta \lvert Ref\rvert \rightarrow \Diamond_{[0,2]} \square_{[0,1]} \neg(\alpha + \beta \lvert Ref\rvert \leq \lvert Pos - Ref\rvert)\big)$ where $\alpha = 0.005$ and $\beta = 0.03$	
WT1	$\square_{[30,630]} \theta \leq 14.2$	
WT2	$\square_{[30,630]} 21000 \leq M_{g,d} \leq 47500$	
WT3	$\square_{[30,630]} \Omega \leq 14.3$	
WT4	$\square_{[30,630]} \Diamond_{[0,5]} \lvert\theta - \theta_d\rvert \leq 1.6$	
CC1	$\square_{[0,100]} y_5 - y_4 \leq 40$	
CC2	$\square_{[0,70]} \Diamond_{[0,30]} y_5 - y_4 \geq 15$	
CC3	$\square_{[0,80]}((\square_{[0,20]} y_2 - y_1 \leq 20) \vee (\Diamond_{[0,20]} y_5 - y_4 \geq 40))$	
CC4	$\square_{[0,65]} \Diamond_{[0,30]} \square_{[0,20]} y_5 - y_4 \geq 8$	
CC5	$\square_{[0,72]} \Diamond_{[0,8]}((\square_{[0,5]} y_2 - y_1 \geq 9) \rightarrow (\square_{[5,20]} y_5 - y_4 \geq 9))$	
F16	$\square_{[0,15]}\, altitude > 0$	
SC	$\square_{[30,35]}(87 \leq pressure \wedge pressure \leq 87.5)$	

Fuel Control of an Automotive Powertrain (AFC). The model is described in [19] and has been used in two previous instalments of this competition [7, 8]. 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 $900 \leq \omega < 1100$ to capture the input profile outlined in [19] and to match the previous competitions. For this reason, we do not consider the unconstrained (instance 1) input specification. Faults are disabled (e.g. by setting `fault_time` > 50).

Neural-network Controller (NN). This benchmark is based on MathWork's neural network controller for a system that levitates a magnet above an electromagnet at a reference position.[10] It has been used previously as a falsification demonstration in the distribution of Breach. The model has one input, a reference value *Ref* for the position, where $1 \leq Ref$ and $Ref \leq 3$. It outputs the current position of the levitating magnet *Pos*. The input specification for instance 1 requires discontinuities to be at least 3 time units apart, whereas instance 2 specifies an input signal with exactly three constant segments. The time horizon for the problem is 40. The requirement ensures that after changes to the reference, the actual position eventually stabilizes around that value with small error.

Wind Turbine (WT). The model is a simplified wind turbine model proposed in [25]. 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.

Chasing cars (CC). The model is derived from Hu et al. [18] which presents a simple model of an automatic chasing car. Chasing cars (CC) model consists of five cars, in which the first car is driven by inputs (*throttle* and *brake*), and other four are driven by Hu et al.'s algorithm. The output of the system is the location of five cars y_1, y_2, y_3, y_4, y_5. The properties to be falsified are constructed artificially, to investigate the impact of complexity of the formulas to falsification. The input specifications for instance 1 allows any piecewise continuous signals while the input specification for instance 2 constraints inputs to piecewise constant signals with control points for each 5 seconds, i.e., 20 segments.

Aircraft Ground Collision Avoidance System (F16). The model has been derived from the one presented in [16]. The F16 aircraft and its inner-loop controller for Ground Collision avoidance have been modeled using 16 continuous variables with piece-wise nonlinear differential equations. Autonomous maneuvers are performed in an outer-loop controller that uses a finite-state machine with guards involving the continuous variables. The system is required to always avoid hitting the ground during its maneuver starting from all the initial conditions for roll, pitch, and yaw in the range $[0.2\pi, 0.2833\pi] \times [-0.4\pi, -0.35\pi] \times [-0.375\pi, -0.125\pi]$.[11] Since the benchmark has no time-varying input, there is no distinction between instance 1 and instance 2. The requirement is checked for a time horizon equal to 15.

Steam condenser with Recurrent Neural Network Controller (SC). The model is presented in [27]. 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.

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

[11] Last year's report erroneously specifies: $[0.2\pi, 0.2833\pi] \times [-0.5\pi, -0.54\pi] \times [0.25\pi, 0.375\pi]$, however, the results were in fact obtained with the correct range.

4 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 2. 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 3.

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 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 $\Delta T = 1$.

ARIsTEO. 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 $nk = 2$[12] as structure for the surrogate model used in the approximation-refinement loop of ARIsTEO. For models with multiple inputs and outputs the dimension of the matrix na, nb and nk is changed depending on the number of inputs and outputs. We used the default configuration of S-TaLiRo for searching failure-revealing revealing tests on the surrogate model. We considered the same parametrization of S-TaLiRo for the input signals. The original Simulink model was executed once to learn the initial surrogate model. The cut-off values for the number of simulations of the original model and for the number of simulations of the surrogate model (per trial) were set to 300. The results of ARIsTEO can further improve by (i) using configurations for the surrogate model that provide more accurate approximations of the original models and more effectively guide the search toward faulty inputs; and (ii) using the SOAR option of S-TaLiRo that significantly improved the results of S-TaLiRo compared with the last edition of this competition.

Properties AT51, AT52, AT53 and AT54 are currently not supported. To verify these properties with S-TaLiRo it is necessary to embed the model into a Matlab function. Then, S-TaLiRo performs simulation by iteratively executing the Matlab function rather than directly simulating the model. This feature is currently not supported by ARIsTEO. We are working on removing this limitation.

`zlscheck`. The inputs of the systems are bounded piecewise constant streams. A bounded piecewise constant stream x of size N and period k is such that $\forall n \in [0, N], x(n) = x(\lfloor \frac{n}{k} \rfloor \cdot k)$.

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

It is totally defined by $(\lfloor \frac{N}{k} \rfloor + 1)$ values (parameters).
Two different strategies were used in these benchmarks:

- classic: the optimization is done offline: the parameters are generated at the beginning of the simulation, then the corresponding robustness is computed. The optimization algorithm then uses the robustness and its gradient w.r.t. the parameters to compute the next parameters.
  This strategy has been used for properties AT1, AT2, AT6, AFC29, AFC33, NN, WT, F16 and SC.

- mode switch: the optimization is performed online: for each input, the parameter number $\lfloor \frac{n}{k} \rfloor$ is generated at step $\lfloor \frac{n}{k} \rfloor \cdot k$ of the simulation.
  In order to achieve coverage of the different modes of the system (a mode is a state in a hierarchical automaton), `zlscheck` chooses randomly an available transition and drives the system towards triggering it: it computes a quantitative interpretation of the complement of its guard (transition robustness) and the gradients of that robustness w.r.t. the current values of the inputs of the system (at step n this would be the parameters number $\lfloor \frac{n}{k} \rfloor$). The value of the next parameter is then computed by the optimization algorithm using its precedent value and the gradient.
  Once the transition has been triggered, the tool chooses a new one randomly and so on, until the simulation ends. In this mode, the input are generated independently of the property being falsified.
  This strategy has been used for properties AT5, AFC27 and CC.

The gradient-based algorithm used by `zlscheck` is a simple gradient descent with decreasing step-size: at step l of the optimization, the step-size is $a(l) = a(0)/\sqrt{l}$ where $a(0)$ is a meta-parameter. The first input is sampled uniformly in the input space, and if the same input is generated twice in a row, the algorithm restarts. Other gradient descent algorithms (ADAM, ADAGRAD, AMSGRAD) are implemented in `zlscheck` and have been tested but did not give significantly better results.
Additionally, given that the integration scheme is fixed-step, we can express the period k of the inputs as times instead of number of simulation steps. For instance 1, those periods are (in seconds): $\Delta T_{AT} = 0.5$ except $\Delta T_{AT2} = 2.5$, $\Delta T_{AT6a} = 5$, $\Delta T_{AT6b} = 10$, $\Delta T_{AT6c} = 15$ and $\Delta T_{AFC} = 5$ and $\Delta T_{NN} = 3$ and $\Delta T_{WT} = 5$ and $\Delta T_{CC} = 5$ and $\Delta T_{F16} = +\infty$, $\Delta T_{SC} = 0.2$.

Discussion. Several tools manage to consistently falsify using just a few simulations only (falsify, ARIsTEO, `zlscheck`), in particular on benchmarks which are difficult for other approaches (NN, CC4). This is remarkable as all sampling and tuning of the input is done on the fly with respect to an established prefix. The results from the first few columns are discussed in more depth in [11].

5 Conclusion and Outlook

The benchmarks established in this competition provide a reasonable basis for comparison of methods, which are starting to be used not only here but also in related work [10].[13] This is an encouraging trend. However, clearly more difficult benchmarks are needed, possibly with classification according to their respective difficulties.

[13] https://github.com/decyphir/ARCH20_ATwSS

Table 2: Results for piecewise continuous input signals (instance 1). *FR*: falsification rate wrt. number of 50 trials, $\overline{S}$ and $\widetilde{S}$: mean resp. median (rounded down) number of simulations over successful trials ("–" if *FR* is zero).

Tool:	**S-TaLiRo**			**Breach**			**FalStar**			**FalStar**			**falsify**			**ARIsTEO**			**zlscheck**		
Configuration:	SOAR			GNM			MCTS			aLVTS			A3C			ARX-2			GD		
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*	$\overline{S}$	$\widetilde{S}$
AT1	50	118.8	116	50	11.0	11	0	–	–	50	33.0	30	17	224.5	223	0	-	-	50	3.4	2
AT2	50	23.9	19	50	2.0	2	50	21.7	5	50	4.3	3	50	22.1	10	50	4.4	4	50	15.5	2
AT51	50	26.7	22	41	74.6	67	50	39.1	43	50	69.5	56	50	1.0	1	(no support)			50	3.5	5
AT52	50	4.1	3	49	72.0	67	50	2.7	2	26	125.4	137	50	1.0	1	(no support)			50	1.6	1
AT53	50	3.4	3	49	74.5	73	50	2.5	3	50	70.8	68	50	1.0	1	(no support)			50	1.3	1
AT54	50	10.5	2	21	84.9	85	50	26.6	10	50	71.1	52	50	1.0	1	(no support)			50	3.5	2
AT6a	49	78.4	40	50	97.9	97	49	93.7	80	50	76.1	70	(not safety)			45	90.7	69	50	48.3	29
AT6b	33	132.6	128	49	112.9	118	29	166.9	173	50	82.4	75	(not safety)			50	18.1	15	35	77.7	102
AT6c	47	61.3	38	50	94.1	89	6	105.9	125	0	–	–	(not safety)			44	95.6	66	32	18.9	23
NN	50	26.7	22	48	96.3	101	50	48.0	40	36	122.8	106	50	1.0	1	50	62.8	46	50	1.0	1
NN($\beta = 0.04$)	4	193.0	222																50	1.1	1
WT1	50	91.0	91	50	3.0	3	50	4.0	4	(no support)			37	47.7	7	50	15.6	10	50	1.4	1
WT2	50	32.6	30	50	3.0	3	50	1.0	1				46	8.0	2	50	1.5	1	48	1.0	1
WT3	50	44.1	60	50	3.0	3	50	2.0	2				50	2.5	1	50	3.4	3	50	1.1	1
WT4	50	3.3	2	50	30.0	30	50	2.0	2				50	4.9	4	50	1.0	1	0	-	-
CC1	50	9.5	7	50	3.0	3	50	15.0	15	50	4.1	2	47	51.3	17	50	16.1	11	50	8.8	12
CC2	50	6.0	4	50	1.0	1	50	26.0	26	50	4.0	2	37	24.2	4	50	1.0	1	50	4.7	3
CC3	50	19.9	5	50	3.0	3	50	14.4	17	50	6.9	5	46	35.4	8	50	45.8	27	50	23.4	16
CC4	20	188.0	179	0	–	–	0	–	–	2	52.0	60	1	26.0	26	50	1.0	1	32	124.6	164
CC5	50	42.9	36	49	26.1	19	50	132.0	140	46	91.2	79	31	29.7	26	49	52.5	40	50	2.0	3
F16	7	127.6	94	1	297.0	297	(no support)			0†	–	–	(no support)			(no support)			0	-	-
SC	50 ⋆	62.2	55	0	–	–	0	–	–	0	–	–	0	–	–	50	1.0	1	50	2.1	2

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 [27]).

Finally, the community should aim at reproducing the experiments and at validating whether the generated input signals are indeed falsifying traces, which needs further work. Here, the translation of the benchmarks to Zélus is highly useful, to have an independent and open source alternative to using Matlab/Simulink as an execution engine.

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. C. Menghi is supported by the European Research Council under the European Union's Horizon 2020 research and innovation programme (grant No 694277). `zlscheck` was funded by the ModeliScale Inria Project Lab.

Table 3: Results for constrained input signals/instance 2. FR: falsification rate (of 50), $\overline{S}$: mean number of simulations, $\widetilde{S}$: median (rounded down) number of simulations.

Tool:	**S-TaLiRo**			**Breach**			**FalStar**			**FalStar**			**falsify**			**ARIsTEO**			**zlscheck**		
Configuration:	SOAR			GNM			MCTS			aLVTS			A3C			ARX-2			GD		
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*	$\overline{S}$	$\widetilde{S}$
AT1	50	170.3	171	0	–	–	0	–	–	50	33.0	30	39	125.8	110	0	-	-	50	2.3	2
AT2	50	16.8	9	50	2.0	2	50	21.7	5	50	4.3	3	48	19.0	7	50	4.5	3	50	9.2	2
AT51	50	12.6	11	50	7.0	7	50	39.1	43	50	69.5	56	50	1.0	1	(no support)			50	8.7	12
AT52	49	17.6	4	50	3.0	3	50	2.7	2	26	125.4	137	50	1.0	1	(no support)			50	35.1	31
AT53	50	3.4	3	50	3.0	3	50	2.5	3	50	70.8	68	50	1.0	1	(no support)			50	22.7	26
AT54	50	24.2	16	50	3.0	3	50	26.6	10	50	71.1	52	50	1.0	1	(no support)			47	56.3	43
AT6a	44	130.4	149	0	–	–	49	93.7	80	50	76.1	70	(not safety)			41	116.3	72	50	42.7	26
AT6b	39	207.2	236	0	–	–	29	166.9	173	50	82.4	75	(not safety)			50	36.3	27	5	129.5	102
AT6c	42	197.5	208	0	–	–	36	105.9	125	0	–	–	(not safety)			44	89.8	73	2	261.7	288
AFC27	50	70.3	78	50	3.0	3	–	–	–	50	3.9	3	50	1.6	1	50	2.3	1	50	1.0	1
AFC29	50	13.0	10	50	3.0	3	–	–	–	50	1.2	1	50	1.0	1	50	28.5	23	0	-	-
AFC33	0	–	–	0	–	–	–	–	–	0	–	–	50	1.0	1	50	24.7	16	24	2.1	2
NN	49	68.0	48	50	6.0	6	50	177.4	183	26	177.0	197	50	1.0	1	50	62.8	46	50	1.3	1
NN(β=0.04)	3	127.0	74																50	1.4	1
WT1	50	7.1	5	50	3.0	3	50	4.0	4	(no support)			49	8.6	2	50	1.4	1	50	1.4	1
WT2	50	1.0	1	50	3.0	3	50	1.0	1				50	2.8	2	50	1.0	1	48	1.0	1
WT3	50	1.0	1	50	3.0	3	50	2.0	2				50	2.0	1	50	1.1	1	50	1.1	1
WT4	50	12.0	9	50	30.0	30	50	2.0	2				50	3.7	3	50	1.0	1	0	-	-
CC1	50	67.8	91	50	5.0	5	50	15.0	15	50	7.3	6	50	23.5	7	50	28.8	22	50	8.8	12.5
CC2	48	114.5	105	50	1.0	1	50	26.0	26	50	15.9	9	46	14.4	4	50	1.0	1	50	4.7	3
CC3	50	22.4	13	50	5.0	5	50	14.4	17	4	207.5	229	44	13.5	2	50	18.1	16	50	23.4	16
CC4	0	–	–	0	–	–	0	–	–	0	–	–	9	120.4	168	50	1.0	1	32	124.6	164
CC5	50	74.9	48	16	84.7	79	50	132.0	140	39	117.9	103	32	37.2	8	48	66.9	44	50	2.0	3
SC	0	–	–	0	–	–	0	–	–	0	–	–	0	–	–	50	1	1	0	-	-

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] Timothy Bourke and Marc Pouzet. Zélus: A synchronous language with ODEs. In *16th International Conference on Hybrid Systems: Computation and Control (HSCC'13)*, pages 113–118, Philadelphia, USA, March 2013.

[5] Jean-Louis Colaço, Bruno Pagano, and Marc Pouzet. Scade 6: A Formal Language for Embedded Critical Software Development. In *TASE 2017 - 11th International Symposium on Theoretical Aspects of Software Engineering*, pages 1–10, Nice, France, September 2017.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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

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

[13] 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.

[14] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous data flow programming language lustre. *Proceedings of the IEEE*, 79(9):1305–1320, 1991.

[15] Nicolas Halbwachs, Fabienne Lagnier, and Pascal Raymond. Synchronous observers and the verification of reactive systems. In *Proceedings of the Third International Conference on Methodology and Software Technology: Algebraic Methodology and Software Technology*, AMAST '93, pages 83–96, Berlin, Heidelberg, 1994. Springer-Verlag.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

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

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

[23] 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.

[24] 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.

[25] 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.

[26] Masaki Waga. Falsification of cyber-physical systems with robustness-guided black-box checking. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, pages 1–13, 2020.

[27] 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.

[28] 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.