On the Effectiveness of Signal Rescaling in Hybrid System Falsification*

Zhenya Zhang[1 [0000−0002−3854−9846]], Deyun Lyu[1 [0000−0003−3017−7977]], Paolo Arcaini[2 [0000−0002−6253−4062]], Lei Ma[1 [0000−0002−8621−2420]], Ichiro Hasuo[2 [0000−0002−8300−4650]], and Jianjun Zhao[1]

[1] Kyushu University, Fukuoka, Japan
[2] National Institute of Informatics, Tokyo, Japan

Abstract. Hybrid system falsification employs stochastic optimization to search for counterexamples to a system specification in Signal Temporal Logic (STL), guided by quantitative STL robustness. The *scale problem* could arise when the STL formula is composed of sub-formulas concerning signals having different scales (e.g., speed [km/h] and rpm): the performance of falsification could be negatively affected because different scales can mask each other's contribution to robustness. A natural solution consists in rescaling the signals to the same order of magnitude. In this paper, we investigate whether this "basic" approach is always effective, or better rescaling strategies could be devised. Experimental results show that basic rescaling is not always the best strategy, and sometimes "unbalanced" rescalings work better. We investigate the reasons of this, and we identify future research directions based on this observation.

Keywords: falsification · Signal Temporal Logic · scale problem · rescaling

1 Introduction

Automated formal verification of *hybrid systems* is almost infeasible due to the infinite search spaces given by the physical components. *Falsification* has been proposed as a more practical approach that, rather than attempting to prove the system specification, tries to violate it: given a *model* $\mathcal{M}$ that takes an input signal $\mathbf{u}$ and outputs a signal $\mathcal{M}(\mathbf{u})$, and a temporal logic *specification* φ (usually in Signal Temporal Logic (STL) [5]), the falsification problem consists in finding an input signal $\mathbf{u}$ such that the corresponding output $\mathcal{M}(\mathbf{u})$ violates φ.

Falsification is usually turned into an optimization problem by exploiting the *robust semantics* of temporal logic formulas [3, 5]: instead of the classical Boolean satisfaction relation $\mathbf{v} \models \varphi$, robust semantics assigns a value $[\![\mathbf{v}, \varphi]\!] \in \mathbb{R} \cup \{\infty, -\infty\}$ (i.e., *robustness*) that tells not only whether φ is satisfied or violated (by the sign), but also assesses *how robustly* it is satisfied or violated. Since negative robustness indicates that

* This work is supported in part by JSPS KAKENHI Grant No.20H04168, 19K24348, 19H04086, and JST-Mirai Program Grant No.JPMJMI18BB, Japan. Paolo Arcaini and Ichiro Hasuo are supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST.

the specification is violated, the goal of falsification is to minimize the robustness to obtain a negative value. Different optimization-based falsification algorithms have been proposed [7], that employ stochastic optimization approaches, such as *hill-climbing*: they generate inputs with the aim of decreasing robustness, and terminate when they find an input with negative robustness (i.e., a *falsifying input*). Also falsification tools, as Breach [2] and S-TaLiRo [1], are available.

The *scale problem* is a recognized issue in falsification [6,8]. It is due to the computation of robust semantics, namely the way in which the robustness values of different sub-formulas are compared: such computation is problematic in the presence of signals that take values having different order of magnitudes. As a simple example, let's consider the formula $\varphi \equiv \varphi_1 \vee \varphi_2$, with $\varphi_1 \equiv \mathit{gear} < 3$ and $\varphi_2 \equiv \mathit{speed} > 35$. According to the robust semantics, the robustness of φ_1, at a given moment t, is $\big(3 - \mathit{gear}(t)\big)$, and of φ_2 is $\big(\mathit{speed}(t) - 35\big)$; the Boolean connective $\vee$, instead, is interpreted by supremum $\sqcup$. Note that the robustness of φ_1 is always in the order of units, while the robustness of φ_2 is, in general, in the order of tens. Because of this, whenever φ_2 is satisfied, it will almost always *mask* the contribution of φ_1 to the final robustness of φ. The situation is even more frequent if complete formulas with temporal operators (the ones we consider) and their semantics are taken into account. Such a masking effect could be problematic for falsification. Indeed, if the contribution of a signal s_1 to the global robustness is masked by another signal s_2, the falsification algorithm has no guidance, because it does not know how s_1 should be modified to falsify the whole formula.

A naïve solution to the scale problem consists in *rescaling* the signals used in the specification at the same scale; we name such approach as *basic rescaling*. In this paper, we are interested in assessing to what extent such approach is effective, i.e., if applying the basic rescaling leads to optimal falsification results. We perform an empirical evaluation using 2 benchmarks and 12 specifications, showing that the basic rescaling is not always the best strategy; indeed, in some cases, scaling the signals in other ways (e.g., making their orders of magnitude even more different) leads to better falsification results. We do a further analysis of these cases, explaining why this is the case; we then describe how such findings can be used to pave new research directions in falsification.

Paper structure. §2 provides some necessary background. §3 introduces the scale problem, and the basic rescaling approach to tackle it. §4 presents the experiments we conducted to assess the effectiveness of the basic rescaling, and to discover whether other rescaling strategies are more effective. Finally, §5 concludes the paper.

2 Preliminary

In this section, we review the falsification framework based on *robust semantics* of temporal logic [3]. Let $T \in \mathbb{R}_+$ be a positive real. An *M-dimensional signal* ($M \in \mathbb{N}$) with a time horizon T is a function $\mathbf{w} \colon [0, T] \to \mathbb{R}^M$. We treat the system model as a black box, i.e., its behaviors are only observed from inputs and their corresponding outputs. Formally, a *system model*, with M-dimensional input and N-dimensional output, is a function $\mathcal{M}$ that takes an input signal $\mathbf{u} \colon [0, T] \to \mathbb{R}^M$ and returns a signal $\mathcal{M}(\mathbf{u}) \colon [0, T] \to \mathbb{R}^N$. Here the common time horizon $T \in \mathbb{R}_+$ is arbitrary.

Definition 1 (STL syntax). We fix a set $\mathbf{Var}$ of variables. In Signal Temporal Logic (STL), *atomic propositions* and *formulas* are defined as follows, respectively: $\alpha ::\equiv f(x_1, \ldots, x_N) > 0$, and $\varphi ::\equiv \alpha \mid \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \,\mathcal{U}_I\, \varphi$. Here f is an N-ary function $f : \mathbb{R}^N \to \mathbb{R}$, $x_1, \ldots, x_N \in \mathbf{Var}$, and I is a closed non-singular interval in $\mathbb{R}_{\geq 0}$, i.e. $I = [a, b]$ or $[a, \infty)$ where $a, b \in \mathbb{R}$ and $a < b$. Other common connectives such as $\to$, $\top$, $\Box_I$ (always) and $\Diamond_I$ (eventually), are introduced as abbreviations: $\Diamond_I \varphi \equiv \top \,\mathcal{U}_I\, \varphi$ and $\Box_I \varphi \equiv \neg\Diamond_I \neg\varphi$.

Definition 2 (Robust semantics). Let $\mathbf{w} \colon [0, T] \to \mathbb{R}^N$ be an N-dimensional signal, and $t \in [0, T)$. The *t-shift* $\mathbf{w}^t$ of $\mathbf{w}$ is the signal $\mathbf{w}^t \colon [0, T - t] \to \mathbb{R}^N$ defined by $\mathbf{w}^t(t') := \mathbf{w}(t + t')$. Let $\mathbf{w} \colon [0, T] \to \mathbb{R}^{|\mathbf{Var}|}$ be a signal, and φ be an STL formula. We define the *robustness* $[\![\mathbf{w}, \varphi]\!] \in \mathbb{R} \cup \{\infty, -\infty\}$ as follows, by induction on the construction of formulas. $\bigsqcap$ and $\bigsqcup$ denote infimums and supremums of real numbers, respectively. Their binary version $\sqcap$ and $\sqcup$ denote minimum and maximum.

$$
\begin{array}{ll}
[\![\mathbf{w}, f(x_1, \cdots, x_N) > 0]\!] := f\big(\mathbf{w}(0)(x_1), \cdots, \mathbf{w}(0)(x_N)\big) & \\
[\![\mathbf{w}, \bot]\!] := -\infty & [\![\mathbf{w}, \neg\varphi]\!] := -[\![\mathbf{w}, \varphi]\!] \\
[\![\mathbf{w}, \varphi_1 \wedge \varphi_2]\!] := [\![\mathbf{w}, \varphi_1]\!] \sqcap [\![\mathbf{w}, \varphi_2]\!] & [\![\mathbf{w}, \varphi_1 \vee \varphi_2]\!] := [\![\mathbf{w}, \varphi_1]\!] \sqcup [\![\mathbf{w}, \varphi_2]\!] \\
[\![\mathbf{w}, \varphi_1 \,\mathcal{U}_I\, \varphi_2]\!] := \bigsqcup_{t \in I \cap [0,T]} \big([\![\mathbf{w}^t, \varphi_2]\!] \sqcap \bigsqcap_{t' \in [0,t)} [\![\mathbf{w}^{t'}, \varphi_1]\!] \big) &
\end{array}
$$

The original STL semantics is Boolean, given by a binary relation $\models$ between signals and formulas. The robust semantics refines the Boolean one in the following sense: $[\![\mathbf{w}, \varphi]\!] > 0$ implies $\mathbf{w} \models \varphi$, and $[\![\mathbf{w}, \varphi]\!] < 0$ implies $\mathbf{w} \not\models \varphi$, see [5, Prop. 16].

Optimization-Guided Falsification Falsification can be transformed into an optimization problem by taking the robustness as objective function. The goal of optimization is to minimize the robustness value by varying input signals—once a negative robustness is found, it indicates the existence of a counterexample violating the system specification. To solve the optimization problem, different metaheuristic-based optimization techniques can be used (e.g., CMA-ES, Simulated Annealing), and these have been implemented in state-of-the-art falsification tools such as Breach [2].

3 A Rescaling Approach for Tackling the Scale Problem

The scale problem [6, 8] is known to affect the falsification performance. We here shed a light on this problem and present a straightforward solution based on signal rescaling.

Scale Problem An STL formula is commonly composed of multiple sub-formulas concerning different signals. These signals are likely to have different magnitudes. First of all, different signals can range differently (e.g., *speed* ranges over $[0, 150]$ while *gear* is an integer less than 5). Moreover, the magnitude of a signal may also depend to the use of different measurement units (e.g., *speed* may be measured in *km/h*, *m/s*, *mph*, etc.). The scale problem arises when the robustness of such a formula is computed: the process requires the comparison between robustness values coming from different sub-formulas (see the definition of robust semantics in Def. 2), and so the global robustness may be dominated by only one of the involved signals. As the optimization process in falsification is guided by robustness, the scale problem can pose an influence on the performance of falsification. We show the harmfulness of this issue via an example.

Example 3. Consider an automatic transmission system that outputs $gear \in \{1, 2, 3, 4\}$ and $speed \in [0, 150]$. A safety property concerning the system is as follows: $\varphi \equiv \Box_I(gear = 4 \rightarrow speed > 35)$. φ is equivalent to $\Box_I(\varphi_1 \vee \varphi_2)$ where $\varphi_1 \equiv \neg(gear = 4)$, $\varphi_2 \equiv speed > 35$. Given a signal $\mathbf{w}$ consisting of $speed$ and $gear$, the calculation of its robustness consists in computing the infimum of $\{[\![\mathbf{w}^t, \varphi_1 \vee \varphi_2]\!] \mid t \in I\}$. This process is unfolded as follows: (i) for each $t \in I$, compute $[\![\mathbf{w}^t, \varphi_1]\!]$ and $[\![\mathbf{w}^t, \varphi_2]\!]$, and take their maximum as $[\![\mathbf{w}^t, \varphi_1 \vee \varphi_2]\!]$; (ii) obtain $\{[\![\mathbf{w}^t, \varphi_1 \vee \varphi_2]\!] \mid t \in I\}$, and take its infimum as $[\![\mathbf{w}, \varphi]\!]$.

In case $T = \{[\![\mathbf{w}^t, \varphi_1 \vee \varphi_2]\!] \mid t \in I\}$ contains values both from $[\![\mathbf{w}^t, \varphi_1]\!]$ and $[\![\mathbf{w}^t, \varphi_2]\!]$, the infimum of T will almost always be dominated by $[\![\mathbf{w}^t, \varphi_1]\!]$ due to the scale issue of the two signals. This scenario is actually common, especially when φ is not falsified. Hence, the contribution of φ_2 to the final robustness of φ is almost always masked by φ_1. Even worse, $[\![\mathbf{w}^t, \varphi_1]\!]$ changes discretely due to the nature of $gear$, and so does the final robustness $[\![\mathbf{w}, \varphi]\!]$; this means that a small variation of the system input may not result in a change of the final robustness. Such *flat robustness* is problematic for optimization (that has no guidance), and so falsification will likely fail.

Rescaling Approach Since the scale problem arises because of signals having different magnitudes, a straightforward solution could be to rescale the signals to the same magnitude. We call this approach as *basic rescaling*. For this approach, domain knowledge regarding the ranges $[l, u]$ of each signal is needed for computing the rescaling factor δ. Namely, given an STL formula concerning two signals ranging over $[l_1, u_1]$ and $[l_2, u_2]$, respectively, the *basic* approach for deciding the rescaling factor δ w.r.t. the former signal is $\delta = \frac{u_2 - l_2}{u_1 - l_1}$.

Note that, in our experiments, we will also investigate the use of "unbalanced" scaling factors that rescale the signal in a different way from the basic rescaling (i.e., not at the same order of magnitude), and we will compare their performance.

In our experiments, we use Simulink for the system models. In order to implement the rescaling approach in Simulink, we perform these two steps: (i) we amplify/diminish a selected signal $\mathbf{w}$ by δ times, by adding a *gain* block to $\mathbf{w}$ with a parameter δ; (ii) we modify the constants of the STL formula in accordance with the rescaled signals.

4 Experimental Evaluation

We selected two Simulink models used in falsification competitions [4], namely, Automatic Transmission (AT) and Abstract Fuel Control (AFC). Our domain knowledge on the signal ranges of the models is as follows: AT takes throttle $th \in [0, 100]$ and brake $br \in [0, 325]$ as input signals, and gives $gear \in \{1, 2, 3, 4\}$, $speed \in [0, 150]$, and $rpm \in [0, 4500]$ as output signals; AFC takes pedal angle $pa \in [8.8, 70]$ and engine speed $es \in [900, 1100]$ as input signals, and gives controller mode $cm \in \{0, 1, 2\}$ and a scalar $\mu \in [0, 0.25]$ (the performance of the system) as output signals. Note that the ranges for input signals are set by users and thus precise, but the ranges for output signals are reported empirically by sampling. In total, we evaluate 9 specifications for AT and 3 for AFC (see Table 1).

Experiments were conducted using Breach 1.2.13 (with CMA-ES as solver) on an Amazon EC2 c4.large, 2.9 GHz Intel Xeon E5-2666, 2 virtual CPU cores, 4 GB RAM.

Table 1: STL specifications ($\Delta_t(\mathbf{w}) = \mathbf{w}^t - \mathbf{w}$)

Spec. ID	Temporal specification in STL
AT1	$\square_{[0,30]}\ (\Delta_1(speed) < 30 \wedge \Delta_1(rpm) < 3500)$
AT2	$\square_{[0,30]}\ (gear = 4 \rightarrow \Diamond_{[0,5]}\ (rpm < 4000))$
AT3	$\square_{[0,30]}\ (\Diamond_{[0,10]}\ (rpm < 600) \rightarrow gear = 1)$
AT4	$\Diamond_{[10,30]}\ (speed > 60 \vee rpm < 1000)$
AT5	$\square_{[0,30]}\ (\Diamond_{[0,8]}\ (speed < 130 \wedge rpm < 4750))$
AT6	$\square_{[0,10]}\ (speed < 50) \vee \Diamond_{[0,30]}\ (rpm > 2520)$
AT7	$\square_{[0,30]}\ (gear = 4 \rightarrow speed \geq 35)$
AT8	$\square_{[0,30]}\ (speed < 135 \wedge rpm < 4780)$
AT9	$\square_{[0,30]}\ (th = 0 \vee br = 0) \rightarrow \square_{[0,30]}\ (speed < 110)$
AFC1	$\square_{[11,50]}\ (cm = 1 \rightarrow \mu < 0.228)$
AFC2	$\Diamond_{[0,50]}\ (pa > 40) \rightarrow \square_{[11,50]}\ (\mu < 0.225)$
AFC3	$\Diamond_{[0,50]}\ (pa > 40) \rightarrow \square_{[11,50]}\ (\Diamond_{[0,8]}\ (\mu < 0.06))$

Table 2: Experimental results with different rescaling strategies

δ	AT1			AT2			AT3			AT4			AT5			AT6		
	SR	time	#sim	SR	time	#sim	SR	time	#sim	SR	time	#sim	SR	time	#sim	SR	time	#sim
no-rescaling	2	57.4	44	23	190.9	144.3	9	7.9	6.1	7	49.7	38.3	30	166.4	130.1	7	377.5	280.3
basic	8	152.2	109.1	30	126.5	86.1	30	3.1	2.3	10	24.2	17.8	18	196.2	140.5	13	375.6	269.2
$rpm \times 10^{-3}$	26	103.8	81.7	30	121.5	96.7	18	6.8	5.4	5	14.8	12	10	69.5	55.2	2	452.0	349
$rpm \times 10^{-2}$	12	138.9	105.7	17	203.0	155.2	14	8.2	6.4	3	19.2	15	11	154.3	114.7	13	389.2	273.8
$rpm \times 10^{-1}$	3	114.7	82.3	19	182.3	123.4	13	8.1	6	8	31.7	23.4	30	1.3	1	13	372.4	266
$rpm \times 10^{1}$	4	54.7	41	18	175.8	130.2	17	8.0	6.1	8	26.6	20.1	30	149.2	109.5	5	348.4	244.6

δ	AT7			AT8			AT9		
	SR	time	#sim	SR	time	#sim	SR	time	#sim
no-rescaling	10	137.0	102.7	11	320.5	236.8	22	268.2	209
basic	28	112.7	84.9	0	-	-	23	192.2	137.5
$speed \times 10^{-2}$	29	152.9	122.7	29	326.1	256.7	1	592.4	461
$speed \times 10^{-1}$	23	136.6	108.0	29	296.4	229.1	0	-	-
$speed \times 10^{1}$	10	82.9	62.3	0	-	-	13	388.8	256.2
$speed \times 10^{2}$	8	139.9	94.1	0	-	-	2	139.1	101.5

δ	AFC1			AFC2			AFC3		
	SR	time	#sim	SR	time	#sim	SR	time	#sim
no-rescaling	4	320.1	234.8	2	487.5	334.5	4	354.6	245.8
basic	11	381.7	271.6	30	312.8	225.9	0	-	-
$\mu \times 10^{1}$	11	363.4	254	9	408.0	259.4	11	303.1	215.5
$\mu \times 10^{2}$	6	387.5	266.2	30	240.3	176.3	4	307.8	197
$\mu \times 10^{3}$	8	405.7	283.9	29	198.7	147.1	1	117.9	78
$\mu \times 10^{4}$	8	467.7	295.9	28	265.9	170.5	2	320.6	228.5

Evaluation An *experiment* consists in the execution of falsification using a given rescaling strategy (*no-rescaling*, *basic rescaling* as described in §3, or a different rescaling), over a specification for 30 *trials*, using different seeds. For each experiment, we collect the *success rate (SR)* as the number of trials in which a falsifying input was found, the average execution *time* of the successful trials, and the average number of simulations. Table 2 reports all experimental results[3]. We analyze them using 3 research questions.

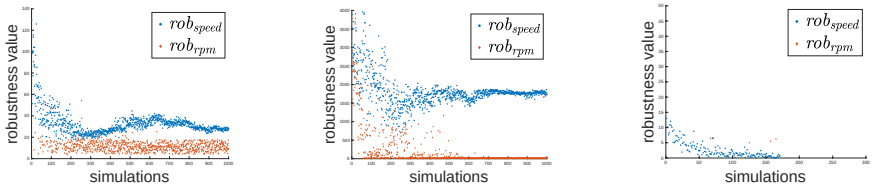
RQ1 *Does the basic rescaling approach always solve the scale problem?*

First, we want to assess the effectiveness of the basic rescaling approach. From Table 2, we observe that in 7 out of 12 cases (AT1, AT2, AT3, AT6, AT7, AFC1, AFC2), the approach does improve the success rate w.r.t. the no-rescaling approach in which the signals are kept in their original order of magnitudes; this is particular evident in AT3. On the other hand, we observe that in some cases there is almost no improvement given by the basic rescaling (as AT4 and AT9) and, in few cases, the basic rescaling approach even diminishes the success rate, as AT5 and AT8: this is an indication that only considering the theoretical ranges of the signals may be not a good strategy, as the concrete robustness values associated to these signals usually have different order of magnitudes.

RQ2 *How does the rescaling factor influence the falsification performance?*

In the previous RQ, we observed that the basic rescaling is not always effective. In this RQ, we investigate whether other types of rescaling (i.e., not at the same order of magnitude) can lead to better falsification results. From Table 2, we observe that this is the case. In 3 out of 12 cases (AT1, AT5, and AT8), the best falsification result is obtained by an "unbalanced" rescaling strategy. This confirms that the best rescaling is

[3] The source code is available at `https://github.com/choshina/FalSTAR-NFM`



(a) No rescaling (b) Basic rescaling (c) Best rescaling

Fig. 1: Comparison of rescaling strategies for AT8

the one that affects the robustness landscape in a way that the falsification algorithm is facilitated. The next RQ provides further analyses that explain such phenomenon.

RQ3 *Why do "unbalanced" rescalings sometimes improve the performance?*
In order to answer this question, we investigate AT8, for which the basic rescaling does not work at all, while two other unbalanced rescalings work very well. AT8 is $\square_{[0,30]}\ (\varphi_1 \wedge \varphi_2)$, where $\varphi_1 \equiv speed < 135$ and $\varphi_2 \equiv rpm < 4780$. We run the falsification algorithm using no-rescaling, basic rescaling, and the best rescaling strategy (i.e., with $speed$ with 10^{-1}), each taking 1000 simulations as timeout. For each simulation, we calculate values rob_{speed} and rob_{rpm} from the output signals $\mathbf{w}$, as follows:

(i) we obtain $\{[\![\mathbf{w}^t, \varphi_1 \wedge \varphi_2]\!] \mid t \in [0, 30]\}$;
(ii) for each $t \in [0, 30]$, we identify whether $[\![\mathbf{w}^t, \varphi_1 \wedge \varphi_2]\!]$ is given by value $[\![\mathbf{w}^t, \varphi_1]\!]$ or $[\![\mathbf{w}^t, \varphi_2]\!]$, and we record $[\![\mathbf{w}^t, \varphi_1]\!]$ in S_{φ_1} or $[\![\mathbf{w}^t, \varphi_2]\!]$ in S_{φ_2}, accordingly. Then, we assign rob_{speed} as $\bigsqcap S_{\varphi_1}$ and rob_{rpm} as $\bigsqcap S_{\varphi_2}$; if S_{φ_1} (or S_{φ_2}) is empty, then rob_{speed} (or rob_{rpm}) is omitted.

In this way, we know, for each sample, which is the signal that contributed to the final robustness. Fig. 1 shows, for the three approaches, rob_{speed} and rob_{rpm} of each sample. Note that the final robustness of each sample is given by the minimum between rob_{speed} and rob_{rpm}. In Fig. 1a, we see that rpm always determines the final value, and no falsifying input is found: this shows that, in this case, falsification driven by rpm is not efficient. This is against the assumption of the basic rescaling that the signals having larger ranges lead to higher robustness values; indeed, we see from Fig. 1b that the basic rescaling actually worsens the situation, having the effect of increasing the robustness related to $speed$. From Fig. 1c, we see that the best rescaling is an unbalanced one, in which the speed is decreased of one order of magnitude: in this case, the falsification algorithm is very effective, and finds a falsifying input after 172 simulations.

5 Conclusion and Future Work

We have shown that having signals of different scales in a specification can affect the falsification effectiveness, because, due to the robust semantics, one signal can mask the other. We have also shown that the naïve approach that rescales the signals to the same order of magnitude does not always solve the problem. Sometimes, "unbalanced" rescalings are better. Future research direction consists in devising falsification approaches that can automatically find such rescalings, either before or during the search.

References

1. Annpureddy, Y., Liu, C., Fainekos, G., Sankaranarayanan, S.: S-TaLiRo: A tool for temporal logic falsification for hybrid systems. In: Tools and Algorithms for the Construction and Analysis of Systems. pp. 254–257. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
2. Donzé, A.: Breach, A toolbox for verification and parameter synthesis of hybrid systems. In: Computer Aided Verification, 22nd Int. Conf., CAV 2010. LNCS, vol. 6174, pp. 167–170. Springer (2010)
3. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: Formal Modeling and Analysis of Timed Systems - 8th Int. Conf. LNCS, vol. 6246, pp. 92–106. Springer (2010)
4. Ernst, G., Arcaini, P., Bennani, I., Donzé, A., Fainekos, G., Frehse, G., Mathesen, L., Menghi, C., Pedrielli, G., Pouzet, M., Yaghoubi, S., Yamagata, Y., Zhang, Z.: ARCH-COMP 2020 category report: Falsification. In: ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20). EPiC Series in Computing, vol. 74, pp. 140–152. EasyChair (2020). https://doi.org/10.29007/trr1
5. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theor. Comput. Sci. **410**(42), 4262–4291 (Sep 2009)
6. Ferrère, T., Nickovic, D., Donzé, A., Ito, H., Kapinski, J.: Interface-aware signal temporal logic. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal, QC, Canada, April 16-18, 2019. pp. 57–66 (2019)
7. Kapinski, J., Deshmukh, J.V., Jin, X., Ito, H., Butts, K.: Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques. IEEE Control Systems **36**(6), 45–64 (Dec 2016)
8. Zhang, Z., Hasuo, I., Arcaini, P.: Multi-armed bandits for boolean connectives in hybrid system falsification. In: Computer Aided Verification. pp. 401–420. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-25540-4_23