Effective Hybrid System Falsification Using Monte Carlo Tree Search Guided by QB-Robustness

Zhenya Zhang[1(✉)], Deyun Lyu[1], Paolo Arcaini[2], Lei Ma[1,3,4], Ichiro Hasuo[2], and Jianjun Zhao[1]

[1] Kyushu University, Fukuoka, Japan
`zhang.zhenya.623@m.kyushu-u.ac.jp`
[2] National Institute of Informatics, Tokyo, Japan
[3] University of Alberta, Edmonton, Canada
[4] Alberta Machine Intelligence Institute, Edmonton, Canada

Abstract. Hybrid system falsification is an important quality assurance method for cyber-physical systems with the advantage of scalability and feasibility in practice than exhaustive verification. Falsification, given a desired temporal specification, tries to find an input of violation instead of a proof guarantee. The state-of-the-art falsification approaches often employ stochastic hill-climbing optimization that minimizes the degree of satisfaction of the temporal specification, given by its quantitative *robust semantics*. However, it has been shown that the performance of falsification could be severely affected by the so-called *scale problem*, related to the different scales of the signals used in the specification (e.g., rpm and speed): in the robustness computation, the contribution of a signal could be *masked* by another one. In this paper, we propose a novel approach to tackle this problem. We first introduce a new robustness definition, called *QB-Robustness*, which combines classical Boolean satisfaction and quantitative robustness. We prove that QB-Robustness can be used to judge the satisfaction of the specification and avoid the scale problem in its computation. QB-Robustness is exploited by a falsification approach based on Monte Carlo Tree Search over the structure of the formal specification. First, tree traversal identifies the sub-formulas for which it is needed to compute the quantitative robustness. Then, on the leaves, numerical hill-climbing optimization is performed, aiming to falsify such sub-formulas. Our in-depth evaluation on multiple benchmarks demonstrates that our approach achieves better falsification results than the state-of-the-art falsification approaches guided by the classical quantitative robustness, and it is largely not affected by the scale problem.

Keywords: Falsification · Signal temporal logic · Scale problem · Monte carlo tree search · Robust semantics · QB-Robustness

This work is supported in part by JSPS KAKENHI Grant No. 20H04168, 19K24348, 19H04086, JST-Mirai Program Grant No. JPMJMI20B8, Japan. Lei Ma is also supported by Canada CIFAR AI Program and Natural Sciences and Engineering Research Council of Canada. Paolo Arcaini and Ichiro Hasuo are supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST.

A. Silva and K. R. M. Leino (Eds.) CAV 2021, LNCS 12759, pp. 595–618, 2021.
https://doi.org/10.1007/978-3-030-81685-8_29

1 Introduction

Cyber-Physical Systems (CPS) are *hybrid systems* that combine physical systems (with continuous dynamics) and digital controllers (that are inherently discrete). Being often safety-critical, their quality assurance is of great importance and widely investigated by both academia and industry. The continuous dynamics of hybrid systems leads to infinite search spaces, making their verification often extremely difficult.

Falsification has been proposed as a more practically feasible approach that tackles the dual problem of verification: instead of exhaustively proving a property, falsification intends to uncover the existence of its violation with counterexamples. Formally, the problem is defined as follows. Given a *model* $\mathcal{M}$ taking an input signal $\mathbf{u}$ and outputting a signal $\mathcal{M}(\mathbf{u})$, and a *specification* φ (a temporal formula), the falsification problem consists in finding a *falsifying input*, i.e., an input signal $\mathbf{u}$ such that the corresponding output $\mathcal{M}(\mathbf{u})$ violates φ.

The most pursued and successful approach to the falsification problem consists in turning it into an optimization problem; we call it *optimization-based falsification*. This is possible thanks to the quantitative *robust semantics* of temporal formulas [14,19]. Robust semantics extends the classical Boolean satisfaction relation $\mathbf{v} \models \varphi$ in the following way: it assigns a value $[\![\mathbf{w}, \varphi]\!] \in \mathbb{R} \cup \{\infty, -\infty\}$ (i.e., *robustness*) that tells not only whether φ is satisfied or violated (by the sign), but also *how robustly* the formula is satisfied or violated.

Optimization-based falsification approaches adopt *hill-climbing* stochastic optimization strategies to generate inputs to decrease robustness, which terminate when they find an input with negative robustness, i.e., a *falsifying input* that triggers the violation of the specification φ. Different optimization-based falsification algorithms have been proposed (see [26] for a survey), and mature tools (e.g., Breach [13] and S-TaLiRo [4]) have also been developed.

The *scale problem* is a recognized issue in optimization-based falsification [21, 40], which could arise when multiple signals with different scales are present in the specification. Namely, it is due to the computation of robust semantics of Boolean connectives, i.e., the way in which the robustness values of different sub-formulas are compared and aggregated: such computation is problematic in the presence of signals that take values having different order of magnitudes.

Example 1. As very simple example, let us consider the formula $\varphi \equiv \Box_{[0,30]}(\varphi_1 \wedge \varphi_2)$, with $\varphi_1 \equiv \mathit{gear} < 6$ and $\varphi_2 \equiv \mathit{speed} < 130$. It is apparent that φ_1 is always satisfied (in any car model with 5 gears), and it has been added in the specification as redundant check.[1] According to robust semantics, the Boolean connective $\wedge$ is interpreted by minimum $\sqcap$, and the "always" operator $\Box_{[0,30]}$ is interpreted by infimum $\bigsqcap$; the robustness of an atomic formula $f(\boldsymbol{x}) < c$ is given by the margin $c - f(\boldsymbol{x})$. Therefore, the robustness of φ under the signal $(\mathit{gear}, \mathit{speed})$, where $\mathit{gear}, \mathit{speed} \colon [0, 30] \to \mathbb{R}$, is

[1] Note that we built such a trivial example just to make the scale problem very easy to understand. However, in general, the scale problem frequently occurs on much less trivial specifications, as we will see in the experiments.

$[\![(gear, speed), \varphi]\!] = \bigsqcap_{t\in[0,30]} \left(\left(6 - gear(t)\right) \sqcup \left(130 - speed(t)\right) \right)$. 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. It is not difficult to see that, if both φ_1 and φ_2 are satisfied, the robustness of φ will only depend on φ_1 (because of the minimum in the robust semantics of the logical connective). In this case, we say that φ_1 *masks* φ_2. In such a case, a falsification approach relying on robustness will be misled during the search. Note that, in this particular case, the only way to falsify φ is to falsify φ_2, because φ_1 is always satisfied; therefore, falsifying this relatively simple formula could be extremely difficult for state-of-the-art optimization-based falsification approaches (as we will show and have confirmed in the experiments).

In this paper, we propose a novel approach to tackle the scale problem in optimization-based falsification. Our intuition and insights are that we should try to avoid the comparison of robustness values of different sub-formulas, so that one sub-formula does not mask the contribution of another one.

To achieve this, we first propose a new way of computing the satisfaction of a formula that combines *quantitative* robust semantics and *Boolean* semantics. We name the new semantics as QB-Robustness. QB-Robustness, for each type of formula φ, requires selecting a sub-formula φ_k among its sub-formulas $\{\varphi_1, \ldots, \varphi_K\}$. For φ_k, the quantitative robust semantics is computed, while for the other sub-formulas the Boolean semantics is computed. Therefore, the computation of QB-Robustness requires identifying a path Σ along the parse tree of the formula φ, where visited sub-formulas are those for which the quantitative robustness is computed. We prove that QB-Robustness, independently of the selected Σ, is equivalent (in terms of sign and satisfaction) to the quantitative robust semantics (and also to the Boolean one).

In general QB-Robustness is a useful tool for avoiding the scale problem of falsification. By definition, the quantitative robustness of different sub-formulas is never compared, so removing the main cause of the scale problem. It would then make sense to use it for guiding the optimization-based falsification process. However, QB-Robustness requires to choose a particular sequence Σ of sub-formulas for which to compute the quantitative robustness. It is relatively easy to show that some of them provide a better guidance than others to the falsification search. Considering the previous example, if Σ contains φ_1, we can encounter the problem that the quantitative robustness of φ_1 would not provide any guidance (i.e., no big variations in the robustness values would be observed). On the other hand, if Σ contains φ_2, the quantitative robustness would have larger variations, providing more effective guidance to the search.

Then, the key problem is how to select the *best* Σ, that enables the hill-climbing optimization used in falsification to be more effective. In general, although it is often difficult to know the best Σ in advance, it is still possible to learn it by observing sampling results using different Σ. Based on this intuition, we propose a novel falsification approach that identifies the sequences Σ that is more likely to be efficient, and uses them in the new falsification trials. Our approach could be seen as an instantiation of the classical Monte

Carlo Tree Search (MCTS) method [8,28], which is able to efficiently tackle the *exploration-exploitation* tradeoff. In our context, exploration consists in incrementally constructing the tree that represents all the possible sequences, and exploitation consists in selecting the best Σ and running optimization-based falsification in which QB-Robustness with Σ is used.

Overall, the major *Contributions* of this paper are summarized as follows:

- We propose a novel semantics (QB-Robustness) for STL formulas that combines quantitative robustness and Boolean satisfaction. We prove that QB-Robustness can be used to show the satisfiability of STL formulas;
- We define a falsification approach based on MCTS that exploits QB-Robustness to address the scale problem;
- We implement the approach in the tool `ForeSee`, based on which, we performed in-depth evaluation, demonstrating the effectiveness and advantage of our approach compared with the state of the art.

Paper Structure. In Sect. 2, we introduce the preliminaries of the optimization-based falsification. In Sect. 3, we introduce the novel STL semantics QB-Robustness, and, in Sect. 4, we describe the MCTS-based falsification approach that uses QB-Robustness. In Sect. 5, we describe the experiments and evaluation results. Finally, we discuss most relevant work to ours in Sect. 6, and conclude the paper in Sect. 7.

2 Preliminaries

In this section, we briefly review the falsification framework based on *robust semantics* of temporal logic [14].

Let $T \in \mathbb{R}_+$ be a positive real. An *M-dimensional signal* 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 **Var** of variables. In Signal Temporal Logic (STL), *atomic propositions* and *formulas* are defined as follows, respectively: $\alpha ::\equiv f(x_1,\dots,x_N) > 0$, and $\varphi ::\equiv \alpha \mid \bot \mid \neg\varphi \mid \bigwedge\varphi \mid \bigvee\varphi \mid \Box_I\varphi \mid \Diamond_I\varphi \mid \varphi\,\mathcal{U}_I\,\varphi$ Here f is an N-ary function $f : \mathbb{R}^N \to \mathbb{R}$, $x_1,\dots,x_N \in \mathbf{Var}$, 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 $a < b$. $\Box, \Diamond$ and $\mathcal{U}$ are temporal operators, which are usually known as *always*, *eventually* and *until* respectively. The always operator $\Box$ and eventually operator $\Diamond$ can also be considered as special cases of the until operator $\mathcal{U}$, where $\Diamond_I\varphi \equiv \top\,\mathcal{U}_I\,\varphi$ and $\Box_I\varphi \equiv \neg\Diamond_I\neg\varphi$. Other common connectives such as $\to, \top$ are introduced as syntactic sugar: $\top \equiv \neg\bot$, $\varphi_1 \to \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$.

Definition 2 (Quantitative 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 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. $\sqcap$, the binary version of $\bigsqcap$, denotes minimum.

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

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 [19, Prop. 16].

2.1 Hill Climbing-Guided Falsification

So far, the falsification problem has received extensive industrial and academic attention. One possible approach direction by hill-climbing optimization is an established field, too: see [2–4,10,13–15,17,26,29,36–39,42] and the tools Breach [13] and S-TaLiRo [4]. We formulate the problem and the methodology, for later use in describing our falsification approach.

Definition 3 (Falsifying Input). Let $\mathcal{M}$ be a system model, and φ be an STL formula. A signal $\mathbf{u} \colon [0, T] \to \mathbb{R}^{|\mathbf{Var}|}$ is a *falsifying input* if $[\![\mathcal{M}(\mathbf{u}), \varphi]\!] < 0$; the latter implies $\mathcal{M}(\mathbf{u}) \not\models \varphi$.

The use of quantitative robust semantics $[\![\mathcal{M}(\mathbf{u}), \varphi]\!] \in \mathbb{R} \cup \{\infty, -\infty\}$ in the above problem enables the use of hill-climbing optimization.

Definition 4 (Hill Climbing-Guided Falsification). Assume the setting in Definition 3, for finding a falsifying input, the methodology of *hill climbing-guided falsification* is presented in Algorithm 1. Here, the function Hill-Climb makes a guess of an input signal $\mathbf{u}'$, aiming at minimizing the robustness $[\![\mathcal{M}(\mathbf{u}'), \varphi]\!]$. It does so, learning from the sampling history H that contains the previous observations of input signals and their corresponding robustness values.

The Hill-Climb function can be designed based on various stochastic optimization algorithms. Typically, at the early phase of the optimization, the proposal of new input is usually based on random sampling; as the set of sampling history grows larger, the algorithm takes various metaheuristic-based strategies to achieve the optimization goal efficiently. Examples of such algorithms include Covariance Matrix Adaption Evolution Strategy (CMA-ES) [7] (used in our experiments), Simulated Annealing, Global Nelder Mead [32], etc.

Algorithm 1. Hill climbing-guided falsification

```
Require: a system model $\mathcal{M}$, an STL formula $\varphi$, and a time budget
 1: function HILL-CLIMB-FALSIFY($\mathcal{M}, \varphi$)
 2:     initialize a placeholder $\mathbf{u}$ and $\mathsf{rb} \leftarrow \infty$        ▷ the best input signal and robustness
 3:     $H \leftarrow \varnothing$        ▷ sampling history of input signals and robustness
 4:     while $\mathsf{rb} \geq 0$ and within the time budget do
 5:         $\mathbf{u}' \leftarrow$ HILL-CLIMB($H$)        ▷ run hill climbing based on sampling history
 6:         $\mathsf{rb}' \leftarrow [\![\mathcal{M}(\mathbf{u}'), \varphi]\!]$        ▷ compute robustness
 7:         $H \leftarrow H \cup \{(\mathbf{u}', \mathsf{rb}')\}$        ▷ update sampling history
 8:         if $\mathsf{rb}' < \mathsf{rb}$ then
 9:             $\mathsf{rb} \leftarrow \mathsf{rb}'$, $\mathbf{u} \leftarrow \mathbf{u}'$        ▷ update the best input and robustness
10:     return $\begin{cases} \mathbf{u} & \text{if } \mathsf{rb} < 0 \\ \text{Failure} & \text{otherwise, that is, no falsifying input found within the budget} \end{cases}$
```

3 QB-Robustness

The scale problem is a known important issue that negatively affects the performance of falsification, which arises when connective operators (i.e., conjunction and disjunction) with operands that predicate on different signals appear in the STL formula under falsification. According to the classic quantitative robust semantics (see Definition 2), the robustness of those formulas is calculated based on the comparison (minimum for conjunction, and maximum for disjunction) between robustness values coming from the different operand sub-formulas. However, since different signals may differ in magnitude, the comparison may be biased, such that one signal $\mathbf{w}$ may always (or often) *mask* the contribution of the others, and, therefore, the final robustness may be dominated by this signal $\mathbf{w}$. Note that, although the scale problem affects connective operators, it is not only local to the place of their application, but it is always propagated to the robustness of the whole formula. The scale problem has been shown as a root cause of the failure of many falsification problems [21,40].

In this work, we propose a novel approach for solving the *scale problem* in falsification. Our approach consists in introducing a new semantics for STL that does not suffer from the scale problem. Such new semantics will be used in a falsification approach based on Monte Carlo Tree Search. We describe details of the new semantics in this section, and the new falsification approach in Sect. 4.

The new proposed semantics, called QB-Robustness, combines quantitative robustness and Boolean satisfaction. By construction, it never compares quantitative robustness values that come from different sub-formulas, thus avoiding the scale problem. QB-Robustness is defined for the whole STL formulas, except for the "until" operator $\varphi_1 \mathcal{U}_I \varphi_2$, when φ_1 is an arbitrary formula. We still support it as "eventually" and "always" operators[2], i.e., when $\varphi_1 = \top$. Note that this is not a major limitation, as QB-Robustness still supports the majority of specifications that are used in industry: indeed, in the experiments, we were able to

[2] Recall from Definition 1 that the "eventually" and "always" operators are defined in terms of the "until" operator.

handle all the specifications used in falsification competitions [18], which collect benchmarks from industrial case studies.

To better explain the computation of QB-Robustness, we introduce some definitions. Let us first define the notion of immediate sub-formula for STL.

Definition 5 (Immediate Sub-Formulas). Let φ be an STL formula (see Definition 1). We define the set $\mathsf{ISForm}(\varphi)$ of *immediate sub-formulas* of φ as follows:

$$\mathsf{ISForm}(\alpha) := \varnothing \qquad \mathsf{ISForm}(\bot) := \varnothing \qquad \mathsf{ISForm}(\neg\varphi) := \mathsf{ISForm}(\varphi)$$
$$\mathsf{ISForm}(\bigwedge_{i\in\{1,\dots,K\}} \varphi_i) := \{\varphi_1,\dots,\varphi_K\} \qquad \mathsf{ISForm}(\bigvee_{i\in\{1,\dots,K\}} \varphi_i) := \{\varphi_1,\dots,\varphi_K\}$$
$$\mathsf{ISForm}(\Box_I\varphi) := \mathsf{ISForm}(\varphi) \qquad \mathsf{ISForm}(\Diamond_I\varphi) := \mathsf{ISForm}(\varphi)$$

Intuitively, the immediate sub-formula set of a connective (conjunction or disjunction) contains all its operands. For the other unary operators (temporal operators, negation, etc.), its immediate sub-formula set is given by the immediate sub-formula set of its argument.

The computation of QB-Robustness requires to select some nested immediate sub-formulas. To this aim, we introduce the notion of sub-formula sequence.

Definition 6 (Sub-Formula Sequence). Let φ be an STL formula. A sub-formula sequence $\Sigma = \sigma_1 \cdot \ldots \cdot \sigma_L$ w.r.t. φ is defined as follows:

$$\sigma_1 \in \mathsf{ISForm}(\varphi) \qquad \sigma_{l+1} \in \mathsf{ISForm}(\sigma_l) \quad \text{with } l = 1,\dots,L-1$$

where the $\cdot$ is the concatenation operator in the sequence. We use Σ_k to denote the kth element of Σ. Moreover, we denote the first element by Σ_{head}, and the last element by Σ_{rear}. We use $\Sigma_{\overline{\mathsf{head}}}$ to denote Σ without Σ_{head}. We identify with ε the empty sequence; when $\mathsf{ISForm}(\varphi) = \varnothing$, we use ε as its sub-formula sequence. We identify with $\boldsymbol{\Sigma}_\varphi$ the set of all the sub-formula sequences rooted in φ.

To be specific, in a sub-formula sequence Σ, each element is one of the sub-formulas of the previous element. This means that, for Boolean connectives, only one of the operands is selected. Moreover, an atomic sub-formula predicating over a single signal can only appear as the final element of a sequence. We exploit these characteristics of Σ to define QB-Robustness, which combines the quantitative robustness of the sub-formulas related to a given signal with the Boolean satisfaction of the other sub-formulas. QB-Robustness, given a sequence Σ, decides whether to compute the quantitative robust semantics or the Boolean semantics of a sub-formula, by considering whether the sub-formula belongs to Σ or not. This implies that, in the case of conjunction and disjunction, we evaluate the quantitative robustness of the sub-formula in Σ and the Boolean satisfaction of the other sub-formulas. Based on such intuition, we define the semantics of our proposed QB-Robustness in Definition 7, and demonstrate its usefulness in Theorem 1.

Definition 7 (Semantics of QB-Robustness). Let φ be an STL formula as defined in Definition 1, and $\varSigma$ be a sub-formula sequence w.r.t. φ. For $\varphi \equiv \bigwedge \varphi_i \mid \bigvee \varphi_i$, let $\varphi_k \in \mathsf{ISForm}(\varphi)$ be the first element $\varSigma_{\mathsf{head}}$ of $\varSigma$, then we can represent these two cases as $\varphi \equiv \varphi_k \wedge \varphi_{\overline{k}} \mid \varphi_k \vee \varphi_{\overline{k}}$, where $\varphi_{\overline{k}}$ is the conjunction (or disjunction, respectively) of the other formulas in $\mathsf{ISForm}(\varphi) \setminus \{\varphi_k\}$. The QB-Robustness $\mathsf{QBRob}(\mathbf{w}, \varphi, \varSigma)$ of φ w.r.t. $\varSigma$ is defined as follows:

$$\begin{aligned}
&\mathsf{QBRob}(\mathbf{w}, \alpha, \varepsilon) := [\![\mathbf{w}, \alpha]\!] \qquad \mathsf{QBRob}(\mathbf{w}, \bot, \varepsilon) := -\infty \\
&\mathsf{QBRob}(\mathbf{w}, \neg\varphi, \varSigma) := -\mathsf{QBRob}(\mathbf{w}, \varphi, \varSigma) \\
&\mathsf{QBRob}(\mathbf{w}, \varphi_k \wedge \varphi_{\overline{k}}, \varSigma) := \begin{cases} \mathsf{QBRob}(\mathbf{w}, \varphi_k, \varSigma_{\overline{\mathsf{head}}}) & \text{if } \mathbf{w} \models \varphi_{\overline{k}} \\ -\infty & \text{otherwise} \end{cases} \\
&\mathsf{QBRob}(\mathbf{w}, \varphi_k \vee \varphi_{\overline{k}}, \varSigma) := \begin{cases} \mathsf{QBRob}(\mathbf{w}, \varphi_k, \varSigma_{\overline{\mathsf{head}}}) & \text{if } \mathbf{w} \not\models \varphi_{\overline{k}} \\ \infty & \text{otherwise} \end{cases} \\
&\mathsf{QBRob}(\mathbf{w}, \Box_I \varphi, \varSigma) := \bigsqcap_{t \in I} \mathsf{QBRob}(\mathbf{w}^t, \varphi, \varSigma) \\
&\mathsf{QBRob}(\mathbf{w}, \Diamond_I \varphi, \varSigma) := \bigsqcup_{t \in I} \mathsf{QBRob}(\mathbf{w}^t, \varphi, \varSigma)
\end{aligned}$$

We now prove that the semantics of QB-Robustness is equivalent (in the sense of satisfaction) to the Boolean semantics, and so it can be used to show violation of a specification in a falsification algorithm, as we do in this paper.

Theorem 1. *Let φ be an STL formula. Given a signal $\mathbf{w}$, for any $\varSigma \in \boldsymbol{\varSigma}_\varphi$, it holds that $\mathsf{QBRob}(\mathbf{w}, \varphi, \varSigma) > 0$ implies $\mathbf{w} \models \varphi$. Similarly, for any $\varSigma \in \boldsymbol{\varSigma}_\varphi$, it holds that $\mathsf{QBRob}(\mathbf{w}, \varphi, \varSigma) < 0$ implies $\mathbf{w} \not\models \varphi$.*

Proof. We first recall from [19, Prop. 16] that $[\![\mathbf{w}, \varphi]\!] < 0$ implies $\mathbf{w} \not\models \varphi$, and that $[\![\mathbf{w}, \varphi]\!] > 0$ implies $\mathbf{w} \models \varphi$. We prove Theorem 1 by induction on the structure of the formula.

- Case $\varphi = \alpha$. By Definition 7, $\mathsf{QBRob}(\mathbf{w}, \alpha, \varepsilon) > 0$ indicates that $[\![\mathbf{w}, \alpha]\!] > 0$ and hence $\mathbf{w} \models \alpha$, and $\mathsf{QBRob}(\mathbf{w}, \alpha, \varepsilon) < 0$ that $[\![\mathbf{w}, \alpha]\!] < 0$ and hence $\mathbf{w} \not\models \varphi$.
- For the following cases, let us assume that Theorem 1 holds for an arbitrary formula φ' and its sub-formula sequence $\varSigma'$ that $\mathsf{QBRob}(\mathbf{w}, \varphi', \varSigma') > 0$ implies $[\![\mathbf{w}, \varphi']\!] > 0$, and that $\mathsf{QBRob}(\mathbf{w}, \varphi', \varSigma') < 0$ implies $[\![\mathbf{w}, \varphi']\!] < 0$. We aim to prove that Theorem 1 also holds for φ, resulting from the application of the operator in each of the following cases to φ', and $\varSigma$, the sub-formula sequence of φ.
  - Case $\varphi = \varphi' \wedge \psi$, where ψ is an arbitrary formula. Let $\varSigma = \varphi' \cdot \varSigma'$, and let us consider the two cases in which $\mathsf{QBRob}(\mathbf{w}, \varphi, \varSigma)$ is negative and positive separately:
    - ∗ If $\mathsf{QBRob}(\mathbf{w}, \varphi, \varSigma) < 0$, there are two sub-cases:
      - · if $\mathsf{QBRob}(\mathbf{w}, \varphi', \varSigma') < 0$, then $[\![\mathbf{w}, \varphi']\!] < 0$ (by assumption). Then, by the robust semantics of conjunction, also $[\![\mathbf{w}, \varphi]\!] < 0$ holds, and so it does $\mathbf{w} \not\models \varphi$.

· if $\mathsf{QBRob}(\mathbf{w}, \varphi', \Sigma') > 0$, then $[\![\mathbf{w}, \varphi']\!] > 0$ (by assumption). Then, it holds $\mathbf{w} \not\models \psi$ by Definition 7, and, therefore, it holds $\mathbf{w} \not\models \varphi$.

$*$ If $\mathsf{QBRob}(\mathbf{w}, \varphi, \Sigma) > 0$, it means that $\mathsf{QBRob}(\mathbf{w}, \varphi', \Sigma') > 0$ and $\mathbf{w} \models \psi$ (by Definition 7). By assumption, if $\mathsf{QBRob}(\mathbf{w}, \varphi', \Sigma') > 0$, then $[\![\mathbf{w}, \varphi']\!] > 0$. Therefore, $\mathbf{w} \models \varphi$.

- Case $\varphi = \Box_I \varphi'$. Let $\Sigma = \Sigma'$, and let us consider the two cases in which $\mathsf{QBRob}(\mathbf{w}, \varphi, \Sigma)$ is negative and positive separately:

  $*$ By Definition 7, $\mathsf{QBRob}(\mathbf{w}, \varphi, \Sigma) < 0$ indicates that there exists a $t \in I$ such that $\mathsf{QBRob}(\mathbf{w}^t, \varphi', \Sigma) < 0$. By assumption, it holds that $\mathbf{w}^t \not\models \varphi'$. Then, by the semantics of the always operator $\Box$, it holds that $\mathbf{w} \not\models \varphi$.

  $*$ By Definition 7, $\mathsf{QBRob}(\mathbf{w}, \varphi, \Sigma) > 0$ indicates that for all $t \in I$ it holds that $\mathsf{QBRob}(\mathbf{w}^t, \varphi', \Sigma) > 0$. Then, by assumption, it holds that for all $t \in I$, $\mathbf{w}^t \models \varphi'$. So, by the semantics of the always operator $\Box$, it holds that $\mathbf{w} \models \varphi$.

- Case $\varphi = \neg\varphi'$. Let $\Sigma = \Sigma'$, and let us consider the two cases in which $\mathsf{QBRob}(\mathbf{w}, \varphi, \Sigma)$ is negative and positive separately:

  $*$ By Definition 7, $\mathsf{QBRob}(\mathbf{w}, \varphi, \Sigma) < 0$ indicates that $\mathsf{QBRob}(\mathbf{w}, \varphi', \Sigma') > 0$. By assumption, it holds that $\mathbf{w} \models \varphi'$, and therefore, $\mathbf{w} \not\models \varphi$.

  $*$ By Definition 7, $\mathsf{QBRob}(\mathbf{w}, \varphi, \Sigma) > 0$ indicates that $\mathsf{QBRob}(\mathbf{w}, \varphi', \Sigma') < 0$. By assumption, it holds that $\mathbf{w} \not\models \varphi'$, and, therefore, $\mathbf{w} \models \varphi$.

- Proofs for the cases of $\varphi = \varphi' \vee \psi$ and $\varphi = \Diamond_I \varphi'$ follow similar proof patterns, and so are left to the readers.

We use an example to illustrate how QB-Robustness is used for checking the satisfiability of an STL formula.

Example 2. Let $\mathbf{w} \colon [0, T] \to \mathbb{R}^2$ be a 2-dimensional signal and $\varphi = \Box_I(\varphi_1 \vee \varphi_2)$ be an STL formula where φ_1 and φ_2 are two atomic formulas. Intuitively, to make φ falsified, there must exist $t \in I$ such that $\mathbf{w}^t \not\models \varphi_1$ and $\mathbf{w}^t \not\models \varphi_2$. Let us consider a non-trivial falsification problem in which, for most of the signals $\mathbf{w}$, sets $\{t \in I \mid \mathbf{w}^t \not\models \varphi_1\}$ and $\{t \in I \mid \mathbf{w}^t \not\models \varphi_2\}$ are non-empty and disjoint.

By Definition 7, given the sub-formula sequence $\Sigma = \varphi_1$ of φ, the corresponding QB-Robustness is $\mathsf{QBRob}(\mathbf{w}, \varphi, \Sigma) = \bigsqcap_{t \in I} \mathsf{QBRob}(\mathbf{w}^t, \varphi_1 \vee \varphi_2, \varphi_1)$, i.e., it takes the infimum of $\mathsf{QBRob}(\mathbf{w}^t, \varphi_1 \vee \varphi_2, \varphi_1)$ over $t \in I$. Again, by Definition 7, for any $t' \in I$, $\mathsf{QBRob}(\mathbf{w}^{t'}, \varphi_1 \vee \varphi_2, \varphi_1)$ is computed as follows:

– if for a $t' \in I$ it holds $\mathbf{w}^{t'} \models \varphi_2$, then $\mathsf{QBRob}(\mathbf{w}^{t'}, \varphi_1 \vee \varphi_2, \varphi_1) = \infty$. Then, it is impossible that $\bigsqcap_{t \in I} \mathsf{QBRob}(\mathbf{w}^t, \varphi_1 \vee \varphi_2, \varphi_1)$ is given by $\mathsf{QBRob}(\mathbf{w}^{t'}, \varphi_1 \vee \varphi_2, \varphi_1)$;
– if for a $t' \in I$ it holds $\mathbf{w}^{t'} \not\models \varphi_2$, then $\mathsf{QBRob}(\mathbf{w}^{t'}, \varphi_1 \vee \varphi_2, \varphi_1) = \mathsf{QBRob}(\mathbf{w}^{t'}, \varphi_1, \varepsilon) = [\![\mathbf{w}^{t'}, \varphi_1]\!]$. In this case, $\mathsf{QBRob}(\mathbf{w}^{t'}, \varphi_1 \vee \varphi_2, \varphi_1)$ has a chance to determine the value of $\bigsqcap_{t \in I} \mathsf{QBRob}(\mathbf{w}^t, \varphi_1 \vee \varphi_2, \varphi_1)$.

Therefore, when $\Sigma = \varphi_1$, it holds that $\mathsf{QBRob}(\mathbf{w}, \varphi, \Sigma) = \bigsqcap_{t \in S} [\![\mathbf{w}^t, \varphi_1]\!]$, where $S = \{t \in I \mid \mathbf{w}^t \not\models \varphi_2\}$, i.e., the infimum of the quantitative robustness of φ_1 on the interval when φ_2 is violated. Indeed, once this value is negative, it means that there exists a point $t \in I$ when both φ_1 and φ_2 are violated; by the Boolean semantics of *always* and *disjunction*, φ is violated.

4 MCTS-Based Falsification Guided by QB-Robustness

QB-Robustness never compares robustness values coming from signals with different magnitudes, and, therefore, it does not suffer from the scale problem. As such, it could be used in falsification approaches instead of the classical pure quantitative robustness.

However, a sub-formula sequence Σ is required when calculating QB-Robustness, and such sequence is not unique (see Definition 7). Note that the selection of the sequence can affect the performance of the numerical optimization algorithms used in falsification. Let us consider $\varphi \equiv \Box((\mathit{gear} < 6) \wedge (\mathit{speed} < 130))$ as an example. As explained in Sect.1, numerical optimization will perform better if guided by the robustness values coming from *speed* rather than by those coming from *gear*. Therefore, in a falsification approach using QB-Robustness, it is important to select an appropriate sub-formula sequence Σ.

By using the QB-Robustness, the problem of falsifying an STL formula φ consists in finding *both* a signal $\mathbf{w}$ and a sub-formula sequence Σ such that $\mathsf{QBRob}(\mathbf{w}, \varphi, \Sigma) < 0$. The selection of Σ is discrete, while the search for $\mathbf{w}$ is numerical. In order to combine these processes that are different in nature, we propose to adapt Monte Carlo Tree Search (MCTS) [8,28]. In the following, we firstly give a brief introduction to MCTS in Subsect. 4.1, and then present the application of MCTS to our falsification problem in Subsect. 4.2.

4.1 MCTS Background

MCTS exemplifies the "trial and error" philosophy, and has achieved a great success over the past decade, most notably in fields such as the computer Go game [35]. MCTS explores the action space given by the possible actions of the system; for example, in the Go game, these are the positions where to put the next stone. The approach builds a tree of sequences of actions, and assigns rewards to the different branches. MCTS performs the search by iteratively taking the following four steps. See Fig. 1, where the general scheme is adapted to our current setting, for illustration.

- *Selection.* It selects a node to expand or to reason about. Initially, selection has no other choice than the root. When a node has multiple expanded children, selection will be done according to the UCB1 [6] algorithm.
- *Expansion.* Child expansion happens after selection if the selected node has unexpanded children. A child will be added to the tree during expansion.
- *Playout.* After a node is just expanded or a leaf is reached, playout is performed for evaluating the node. The evaluation is given by a *reward*, which

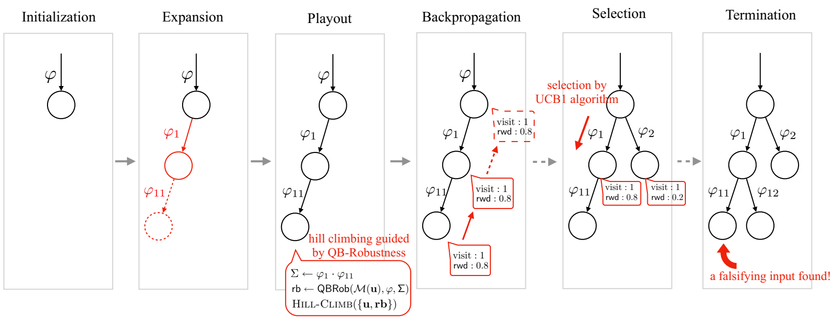
Initialization
Expansion
Playout
Backpropagation
Selection
Termination
φ
φ_1
φ_{11}
φ_2
φ_{12}
hill climbing guided by QB-Robustness
$\Sigma \leftarrow \varphi_1 \cdot \varphi_{11}$
rb ← QBRob($\mathcal{M}(\mathbf{u}), \varphi, \Sigma$)
HILL-CLIMB($\{\mathbf{u}, \mathbf{rb}\}$)
visit : 1 rwd : 0.8
visit : 1 rwd : 0.2
selection by UCB1 algorithm
a falsifying input found!

Fig. 1. The workflow of MCTS-based falsification guided by QB-Robustness. Let us consider the falsification of an STL formula $\varphi = \square_I\ (\varphi_1 \vee \varphi_2)$, where $\varphi_1 = \varphi_{11} \wedge \varphi_{12}$. Initially, there is only the root in the tree, so the algorithm selects it for expansion. Then, the algorithm keeps on randomly selecting a child of a non-fully expanded node, until a leaf node is reached. By reaching a leaf, a sub-formula sequence Σ has been constructed; the algorithm performs playout using Σ, by running hill-climbing optimization guided by the QB-Robustness with Σ, to estimate the reward of the path. After playout, the algorithm backpropagates the reward and the number of visits from the leaf to the root. When all the children of a node are expanded, selection is done based on the UCB1 algorithm. After many loops, the algorithm has explored all the possible sub-formula sequences in $\boldsymbol{\Sigma}_\varphi$, and it starts allocating more resources to those branches where hill-climbing optimization progresses more smoothly. The algorithm terminates either when a falsifying input is found, or when the budget is exhausted.

is a real number in $[0, 1]$. Reward can be interpreted differently in different contexts. For example, in the Go game, the reward of a position is measured by the winning rate when a stone is positioned there; this is estimated by randomly playing the game until the end for n times, and then taking the ratio $\frac{n_w}{n}$ of the number of winning as the winning rate.

– *Backpropagation.* Backpropagation updates the number of visits and the reward of the nodes along the path from the node of playout to the root. These data are used in subsequent loops to decide the branches to explore.

At the end, the action space will be sufficiently explored in an unbalanced manner, by focusing on the most promising sub-spaces giving the highest rewards.

4.2 Proposed QB-Robustness-Guided Falsification Approach

We here propose a falsification framework based on MCTS in which, during tree construction, we synthesize and select a sub-formula sequence that facilitates the falsification progress the most, and, at the bottom layer of the tree, we run numerical optimization to search for a falsifying input and provide feedback (i.e., backpropagation) to guide the sequence selection.

We formalize our algorithm in Algorithm 2 and visualize its execution in Fig. 1. In the following, we elaborate on our approach.

Algorithm 2. MCTS-based falsification guided by QB-Robustness

Require: a system model $\mathcal{M}$, an STL formula φ, and the following tunable parameters: a scalar c for UCB1, an MCTS budget B_M, and a playout budget B_P.

```
 1: function MCTS
 2:     $\Sigma^{\mathsf{init}} \leftarrow \varphi$                ▷ the root denoted as a sequence with $\varphi$ only
 3:     $\mathcal{T} \leftarrow \{\Sigma^{\mathsf{init}}\}$                ▷ the MCTS search tree, initially root only
 4:     $N \leftarrow (\Sigma^{\mathsf{init}} \mapsto 0)$                ▷ visit count $N$ initialized, defined only for root
 5:     $R \leftarrow (\Sigma^{\mathsf{init}} \mapsto 0)$                ▷ reward function $R$ initialized
 6:     $H \leftarrow (\Sigma^{\mathsf{init}} \mapsto \varnothing)$                ▷ the sampling history of hill climbing
 7:     while $\varphi$ not falsified and within the MCTS budget $B_M$ do
 8:         MCTSSearch($\Sigma^{\mathsf{init}}$)

 9: function MCTSSearch($\Sigma$)
10:     if $\mathsf{ISForm}(\Sigma_{\mathsf{rear}}) \neq \varnothing$ then                ▷ the node has children
11:         if $\Sigma \cdot \varphi_k \in \mathcal{T}$ for all $\varphi_k \in \mathsf{ISForm}(\Sigma_{\mathsf{rear}})$ then                ▷ all children expanded
12:             $\varphi_k \leftarrow \operatorname*{arg\,max}_{\varphi_i \in \mathsf{ISForm}(\Sigma_{\mathsf{rear}})} \left( R(\Sigma \cdot \varphi_i) + c\sqrt{\frac{2 \ln N(\Sigma)}{N(\Sigma \cdot \varphi_i)}} \right)$   ▷ selection by UCB1
13:         else                ▷ unexpanded children exist
14:             randomly select $\varphi_k$ from $\{\varphi_k \in \mathsf{ISForm}(\Sigma_{\mathsf{rear}}) \mid \Sigma \cdot \varphi_k \notin \mathcal{T}\}$
15:             $\mathcal{T} \leftarrow \mathcal{T} \cup \{\Sigma \cdot \varphi_k\}$                ▷ expand a new child
16:             $N(\Sigma \cdot \varphi_k) \leftarrow 0$
17:             $H(\Sigma \cdot \varphi_k) \leftarrow \varnothing$
18:         MCTSSearch($\Sigma \cdot \varphi_k$)                ▷ recursive call
19:         $R(\Sigma) \leftarrow \max_{\varphi_k \in \mathsf{ISForm}(\Sigma_{\mathsf{rear}})} R(\Sigma \cdot \varphi_k)$                ▷ back propagation for reward
20:     else                ▷ a leaf node reached
21:         while within playout budget $B_P$ do ▷ playout by hill-climbing falsification
22:             $\mathbf{u} \leftarrow$ Hill-Climb($H(\Sigma)$)                ▷ hill-climbing
23:             $\mathsf{rb} \leftarrow \mathsf{QBRob}(\mathcal{M}(\mathbf{u}), \varphi, \Sigma_{\overline{\mathsf{head}}})$
24:             if $\mathsf{rb} < 0$ then                ▷ falsifying input found
25:                 return $(\mathbf{u}, \mathsf{rb})$
26:             $H(\Sigma) \leftarrow H(\Sigma) \cup \{(\mathbf{u}, \mathsf{rb})\}$                ▷ record sampling history
27:         $R(w) \leftarrow \mathsf{Rwd}(\mathsf{rb}, H(\Sigma))$
28:     $N(\Sigma) \leftarrow N(\Sigma) + 1$                ▷ back propagation for visit count
```

We construct the tree in this way: each node represents a sequence of formulas, and each edge of a node is a sub-formula of the last element of the sequence represented by the node. The root is initialized with a sequence holding φ only (Lines 2–3) and some other properties including the number of visits to the different nodes (Line 4), the reward (Line 5), and the history of hill-climbing sampling (Line 6). The main process of MCTS consists in calling the MCTSSearch function iteratively with the root as argument (Line 8), until the exhaustion of the MCTS budget or a falsifying input is found (Line 7). The MCTSSearch function (Line 9) goes through the four phases, namely *selection*, *expansion*, *playout* and *backpropagation*, of the original MCTS algorithm.

Selection. Selection happens when a node has children (Line 10) and these have all been expanded (Line 11). It selects a child according to the UCB1 [6] algorithm (Line 12) to take a balance between exploration and exploitation. The exploitation is embodied by the reward $R(\Sigma \cdot \varphi_i)$—the higher the reward is, the more likely a falsifying input is found following that branch. Exploration, instead, is considered via $\sqrt{\frac{2 \ln N(\Sigma)}{N(\Sigma \cdot \varphi_i)}}$ that is negatively correlated to the number of visits to a child—the more the child was visited before, the less chance it will be visited again. The scalar c is a tunable parameter that balances the trade-off between exploration and exploitation. After a child $\Sigma \cdot \varphi_k$ is selected, it will be taken as the argument of the next MCTSSearch loop (Line 18).

Expansion. If not all the children of a node have been expanded (Line 13), a child will be expanded. Expansion consists in randomly selecting a child from the unexpanded child list (Line 14), adding it to the tree (Line 15), initilizing properties including the number of visits and history (Lines 16–17). After expansion, the newly expanded child will be taken as the argument of the recursive call to MCTSSearch (Line 18).

Playout. If a leaf node that has no children to expand is reached, the playout phase will start to devise a reward for evaluating the visited path. In our context, we define the reward based on the best robustness value that can be obtained with the path; specifically, playout consists in running hill-climbing guided falsification to search for a minimal robustness value (Line 22). Note that the sequence Σ represented by a leaf node is actually the concatenation between φ and a sub-formula sequence of φ. We extract the suffix of Σ, i.e., the sub-formula sequence, to compute the QB-Robustness as a guidance to the hill-climbing optimization (Line 23). If a negative QB-Robustness is found (Line 24), then the whole algorithm can be terminated and the input signal $\mathbf{u}$ that triggers the negative QB-Robustness can be returned as the falsifying input (Line 25); otherwise, the sampling history of hill climbing will be saved (Line 26) so that the future playout at the same leaf can be restored from that context. After playout, the reward of the leaf node will be updated based on the definition of the reward, which will be introduced below. *Reward* Since our goal is to find a sequence Σ with which hill-climbing optimization can minimize $\mathsf{QBRob}(\mathbf{w}, \varphi, \Sigma)$ smoothly, we connect the reward with the hill-climbing progress. Formally, given a sampling history H, our reward (Line 27) is defined as $\mathsf{Rwd}(\mathsf{rb}', H) := \frac{\max \mathsf{rb}_h - \min (\{\mathsf{rb}'\} \cup \mathsf{rb}_h)}{\max \mathsf{rb}_h}$, where rb_h is the history of robustness values in H.

Backpropagation. In MCTS, the playout result of a leaf is backpropagated to the higher layer nodes along the path, so that the future selection on the high layer is referred. Backpropagation updates two properties of each ancestor of the leaf till the root, the reward (Line 19) and the number of visits (Line 28).

Remark 1 (Approach Complexity). With respect to classical falsification, our approach introduces an exploration phase for searching the "best" sub-formula sequence to instantiate QB-Robustness. The number of these sequences corresponds to the number of atomic sub-formulas (and so the leaves of the

Table 1. Benchmarks – STL specifications

Spec. ID	Temporal specification in STL
AT1	$\Box_{[0,30]}\ (\mathit{gear} = 4 \rightarrow \mathit{speed} > 35)$
AT2	$\Box_{[0,30]}\ (\mathit{gear} = 4 \rightarrow \Diamond_{[0,5]}\ (\mathit{rpm} < 4300))$
AT3	$\Box_{[0,30]}\ (\mathit{speed} < 130 \wedge \mathit{gear} < 5)$
AT4	$\Box_{[0,30]}\ (\mathit{speed} < 135 \wedge \mathit{rpm} < 4780)$
AT5	$\Box_{[0,30]}\ (\mathit{rpm} < 600 \rightarrow \Diamond_{[0,10]}\ (\mathit{gear} > 1))$
AT6	$\Box_{[0,30]}\ (\Diamond_{[0,5]}(\mathit{speed} < 120 \vee \mathit{rpm} > 3500))$
AT7	$\Box_{[0,30]}\ (\mathit{rpm} < 4750 \wedge \mathit{gear} < 5)$
AT8	$\Box_{[0,10]}\ (\mathit{speed} < 50) \vee \Diamond_{[0,30]}\ (\mathit{rpm} > 2520)$
AT9	$\Diamond_{[10,30]}(\mathit{speed} < 50 \vee \mathit{speed} > 60 \vee \mathit{rpm} < 1000)$
AT10	$\Box_{[0,30]}\ (\mathit{gear} = 4 \rightarrow (\mathit{speed} > 35 \wedge \Diamond_{[0,5]}\ (\mathit{rpm} < 4000)))$
AT11	$\Box_{[0,30]}\ (\Diamond_{[0,8]}\ (\mathit{gear} = 1 \rightarrow (\mathit{speed} < 20 \wedge \mathit{rpm} < 600)))$
AT12	$\Box_{[0,30]}\ (\mathit{gear} < 3) \vee \Box_{[0,30]}\ (\mathit{speed} < 135 \wedge \mathit{rpm} < 4780)$
AT13	$\Box_{[0,30]}\ ((\mathit{gear} = 4 \rightarrow \Diamond_{[0,5]}\ (\mathit{rpm} < 4000)) \wedge \mathit{gear} < 5)$
AT14	$\Box_{[0,30]}\ (\mathit{throttle} = 0 \vee \mathit{brake} = 0) \rightarrow \Box_{[0,30]}\ (\mathit{speed} < 110)$
AT15	$\Box_{[0,30]}((\mathit{rpm} < 4770 \vee \Box_{[0,1]}\ (\mathit{rpm} > 1000)) \wedge \Diamond_{[0,5]}\ (\mathit{gear} < 5))$
AT16	$\Box_{[0,30]}\ (\mathit{gear} = 4 \rightarrow ((\Diamond_{[0,5]}\ (\mathit{rpm} < 3000) \wedge (\mathit{gear} = 2 \rightarrow \mathit{speed} < 20))))$
AT17	$\Box_{[0,5]}\ (\mathit{speed} < 70 \wedge \mathit{gear} < 4) \wedge \Box_{[10,20]}(\mathit{rpm} < 4780) \wedge \Box_{[25,30]}(\mathit{speed} < 130)$
AT18	$\Box_{[0,30]}\ ((\mathit{gear} = 4 \rightarrow \Diamond_{[0,5]}\ (\mathit{rpm} < 4250)) \wedge (\mathit{gear} = 3 \rightarrow \Diamond_{[0,5]}\ (\mathit{rpm} < 4700)) \wedge (\mathit{gear} = 2 \rightarrow \Diamond_{[0,5]}\ (\mathit{rpm} < 4800)))$
AT19	$\Box_{[0,30]}\ ((\mathit{gear} = 1 \rightarrow \mathit{speed} < 80) \wedge (\mathit{gear} = 2 \rightarrow \mathit{speed} < 90) \wedge (\mathit{gear} = 3 \rightarrow \mathit{speed} > 20) \wedge (\mathit{gear} = 4 \rightarrow \mathit{speed} > 30))$
AT20	$\Box_{[0,29]}\ (\mathit{speed} < 100) \vee \Box_{[29,30]}(\mathit{speed} > 64) \wedge \Box_{[0,30]}\ (\mathit{rpm} < 4770 \vee \Box_{[0,1]}\ (\mathit{rpm} > 700))$
AT21	$\Box_{[0,30]}\ (\mathit{throttle} > 90 \rightarrow \Diamond_{[0,10]}\ (\mathit{throttle} < 30)) \rightarrow \Box_{[0,30]}\ (\mathit{gear} = 4 \rightarrow \Diamond_{[0,5]}\ (\mathit{rpm} < 4000))$
AT22	$\Box_{[0,30]}\ (\mathit{throttle} > 70 \rightarrow \Diamond_{[0,10]}(\mathit{brake} > 50)) \rightarrow \Box_{[0,30]}(\mathit{gear} = 4 \rightarrow \mathit{speed} > 35)$
AFC1	$\Box_{[11,50]}\ (\mathit{mode} = 1 \rightarrow \mu < 0.228)$
AFC2	$\Diamond_{[0,50]}\ (\mathit{PedalAngle} > 40) \rightarrow \Box_{[11,50]}\ (\mu < 0.225)$
AFC3	$\Diamond_{[0,50]}\ (\mathit{EngineSpeed} > 1000) \rightarrow \Box_{[11,50]}\ (\mu < 0.225)$
AFC4	$\Box_{[0,50]}\ (\mathit{EngineSpeed} > 910 \vee \mathit{PedalAngle} > 25) \rightarrow \Box_{[11,50]}\ (\mu < 0.225)$
AFC5	$\Diamond_{[0,50]}\ (\mathit{PedalAngle} > 40) \rightarrow \Box_{[11,50]}\ (\Diamond_{[0,8]}\ (\mu < 0.06))$
AFC6	$\Diamond_{[0,50]}\ (\mathit{PedalAngle} > 40 \wedge \mathit{EngineSpeed} > 1000) \rightarrow \Box_{[11,50]}\ (\Diamond_{[0,8]}\ (\mu < 0.06))$
FFR1	$\Box_{[0,5]}((u1, u3 > 0 \vee u1, u3 < 0) \wedge (u2, u4 > 0 \vee u2, u4 < 0)) \rightarrow \Box_{[0,5]}(\neg(x1 > 3.9 \wedge x1 < 4.1) \wedge \neg(x3 > 3.9 \wedge x3 < 4.1))$
FFR2	$\neg(\Diamond_{[0,5]}(\Box_{[0,2]}(x1 > 1.5 \wedge x1 < 1.7 \wedge x3 > 1.5 \wedge x3 < 1.7)))$

tree). Considering that most of the time is spent on playout, the complexity of our approach grows linearly with the number of atomic sub-formulas.

5 Experimental Evaluation

In this section, we present the experiments we conducted to evaluate the effectiveness of the proposed approach. We first introduce the experiment setup in Subsect. 5.1, and then we present the experimental evaluation results by answering three research questions in Subsect. 5.2.

5.1 Experiment Setup

Simulink Models and Specifications As our benchmarks, we selected three Simulink models frequently used in the falsification community (i.e., in the falsification competitions [18]), and 30 specifications defined for them. All these models are complicated hybrid systems with multiple input and output signals. The specifications are STL formulas that formalize system requirements regarding safety, performance, etc. Since we are interested in assessing the influence of the scale problem to the performance of the compared falsification approaches, all the considered specifications predicate over, at least, two signals. Table 1 reports the 30 specifications under test. The IDs of the specifications identify which models they belong to. A description of the three models and of their specifications is as follows.

– *Automatic Transmission* (AT) [24] has two input signals, *throttle* $\in [0, 100]$ and *brake* $\in [0, 325]$, and three outputs signals including *gear*, *speed* and *rpm*. Most of the specifications we used formalize safety requirements of the system. For instance, AT2 requires that when *gear* is as high as 4, *rpm* should not be larger than 4300; AT3 is an adaptation of the example we used in Sect. 1; AT10-12 reason about the relationship among the three output signals; AT17 specifies three properties for three different time intervals; AT18 specifies different properties for different values of *gear*; AT14, AT21 and AT22 impose logical constraints on input signals, in addition to the property under consideration.
– *Abstract Fuel Control* (AFC) [25] takes two input signals, *PedalAngle* $\in [8.8, 70]$ and *EngineSpeed* $\in [900, 1100]$, and outputs a ratio μ reflecting the deviation of *air-fuel-ratio* from its reference value. The basic safety requirement to this system is that μ should not be deviated from the reference value too much (AFC1); in addition to that, our specifications also reason about the resilience of the system (AFC5 and AFC6), and impose input constraints (AFC2-6).
– *Free Floating Robot* (FFR) [11] models robot moving in a 2-dimentional space. It has four input signals $u_1, u_2, u_3, u_4 \in [-10, 10]$ that are boosters for a robot, and four output signals that are the position in terms of coordinate values x, y and their one-order derivatives $\dot{x}, \dot{y}$. The specifications regulate the kinetic properties of the robot: FFR1 requires the robot to pass an area around the point $(4, 4)$ under an input constraint, and FFR2 requires the robot to stay in an area for at least 2 s.

Baseline Approach and Our Proposed Approach. In our experiments, we compare the performances of our proposed approach with the baseline `Breach` approach. We implemented our approach in the tool `ForeSee`, which stands for **FOR**mula **E**xploitation by **S**equence tr**EE** for falsification.

`Breach` is a state-of-the-art falsification tool that implements the classic falsification workflow we introduced in Sect. 2. The quantitative robustness calculation in `Breach` is based on the robust semantics given in Definition 2. `Breach` also encapsulates several stochastic optimization algorithms, such as CMA-ES, Simulated Annealing, etc. The implementation of our `ForeSee` approach uses `Breach` only for interfacing with the Simulink model and for the calculation of quantitative robustness; instead, the calculation of QB-Robustness, and the implementation of the MCTS algorithm are novel. Since CMA-ES has proved to be the state-of-the-art stochastic algorithm [39], we select CMA-ES as our backend optimizer for the playout phase.[3]

[3] `ForeSee` is available at https://github.com/choshina/ForeSee.

We apply the two approaches, `ForeSee` and `Breach`, to each benchmark specification reported in Table 1. Since both approaches are based on stochastic optimization, we repeat each experiment for 30 times, as suggested by a guideline for conducting experiments with randomized algorithms [5]. For each experiment, both approaches have been given a total timeout B_M of 900 s (see Algorithm 2).

Evaluation Metrics. As first evaluation metric, we compute the *falsification rate (FR)* as the number of runs (out of 30) in which the approach returns a falsifying input. Therefore, FR is an indicator of the *effectiveness* of an approach, i.e., it reflects the ability of an algorithm to falsify the specification. As second evaluation metric, we compute the *average time* (seconds), as average execution time of the successful falsification runs. Therefore, the average time is an indicator of the *efficiency* of the approach. We do not report the number of simulations because these are consistent with the execution time.

Experiment Platform. In our experiments, we use `Breach` [13] (ver 1.2.13) with CMA-ES (the state of the art). `Breach` accepts piece-wise constant signals as input for the Simulink models; we use the same settings used in falsification competitions [18]: we use piece-wise constant signals with five control points for AT and AFC, and with four control points for FFR. As configuration of MCTS (see Algorithm 2), we set the UCB1 scalar c to 0.2, and the playout budget B_P to 10 generations. The experiments have been executed on an Amazon EC2 c4.2xlarge instance (2.9 GHz Intel Xeon E5-2666 v3, 15 GB RAM).

5.2 Evaluation

We here analyze the experimental results using three research questions (RQs).

RQ1. *Does the proposed approach perform better than state-of-the-art falsification approaches?*

In this RQ, we aim at assessing whether the proposed approach is indeed able to tackle the scale problem in falsification and performs better than state-of-the-art approaches. Table 2 reports, for each specification benchmark, the falsification rate FR and the average execution time of our proposed approach `ForeSee` and of the baseline `Breach`. The table further reports the difference of the two metrics between the two approaches. We highlight in gray the best results in which `ForeSee` has an FR of 15 units higher than `Breach`. We observe that for 25 benchmarks out of 30, `ForeSee` has a better FR, and in 15 of these the improvement is significant (selected in gray). Note that there are notable cases, such as AT3, AT13, AT16, and AT17, in which `Breach` only finds at most two falsifying inputs, while `ForeSee` finds always at least 29 falsifying inputs. In four cases, `Breach` has a better FR: while for AT8, AFC6, and FFR2 the difference is minimal, it is quite large for AT14. We further inspected such specification and its corresponding model (see Table 1); we noticed that all the sub-formulas in AT14 must be falsified to falsify the whole specification[4], and they are all

[4] Note that all binary connectives of AT14 are disjunctions; indeed, $A \rightarrow B$ is the syntactic sugar for $\neg A \bigsqcup_B$.

Table 2. Falsification performance comparison between `Breach` and `ForeSee` on benchmarks. Timeout: 900 s. FR in (/30), time in secs.

	`Breach`		`ForeSee`			
	FR	time	FR	time	ΔFR	Δtime
AT1	12	67.0	29	90.3	+17	+23.3
AT2	18	208.5	30	155.5	+12	-53.0
AT3	0	-	29	87.3	+29	-
AT4	8	414.0	30	376.6	+12	-37.4
AT5	13	44.7	30	159.0	+17	+114.3
AT6	14	630.5	20	545.9	+6	-84.6
AT7	20	24.9	30	5.8	+10	-19.1
AT8	17	418.5	13	547.0	-4	+128.5
AT9	9	298.6	29	208.0	+20	-90.6
AT10	14	99.4	30	89.7	+16	-9.7
AT11	17	58.1	30	39.6	+13	-18.5
AT12	5	379.2	28	381.4	+23	+2.2
AT13	2	75.2	29	98.3	+27	+23.1
AT14	24	184.9	1	601.5	-23	+416.6
AT15	1	66.1	9	331.8	+8	+265.7
AT16	1	13.0	30	6.7	+29	-6.3
AT17	0	-	30	208.8	+30	-
AT18	18	160.0	24	234.3	+6	+74.3
AT19	15	81.8	30	154.3	+15	+72.5
AT20	1	97.7	5	286.2	+4	+188.5
AT21	10	239.0	29	425.5	+19	+186.5
AT22	13	72.0	30	113.3	+17	+41.3
AFC1	10	532.2	12	458.0	+2	-74.2
AFC2	12	546.9	30	218.3	+18	-328.6
AFC3	8	727.6	28	232.5	+20	-495.1
AFC4	7	634.5	22	500.3	+15	-134.2
AFC5	8	576.9	9	322.0	+1	-254.9
AFC6	10	518.2	6	344.2	-4	-174.0
FFR1	7	132.1	7	399.3	+0	+267.2
FFR2	30	38.0	27	348.0	-3	+310.0

difficult to be falsified. In such a case, there is no best sub-formula sequence Σ: therefore, the time spent by `ForeSee` in exploring different Σ does not provide any improvement.

Regarding the time execution, there is no clear trend among the different results: sometimes `ForeSee` is faster, other times `Breach` is. However, even in the cases in which `ForeSee` is slower, it is still below the timeout by which it manages to find a falsifying input (so, leading to better falsification rates).

RQ2. *Does the proposed approach solve the scale problem effectively?*

The benchmarks reported in Table 1 and experimented in RQ1, predicate over signals having different scales and so they suffer from the scale problem. RQ1 showed that `ForeSee` is very efficient in falsifying them. In this RQ, we want to make a more systematic study of the effects of the scale problem; indeed, the scale problem could manifest itself in *different ways*, depending on the difference of the order of magnitudes of the different signals (e.g., speed [km/h] vs. rpm, or speed [km/h] vs. rph). To assess this, we take six specifications from Table 1 and we artificially modify their outputs: namely, we multiply by 10^k (with different k values depending on the specification) the *speed* of AT1, AT3, AT4,

Table 3. Falsification performance under different scales. Each rescaled signal is rescaled by 10^k.

(a) AT1: *speed* $\times 10^k$

	Breach		ForeSee	
k	FR	time	FR	time
-2	30	126.5	26	77.5
-1	25	64.4	29	107.9
0	12	67.0	29	90.3
1	9	92.4	28	81.8
2	9	131.9	30	94.2
min	9	64.4	26	77.5
max	30	131.9	30	107.9
mean	17	96.4	28	90.3

(b) AT3: *speed* $\times 10^k$

	Breach		ForeSee	
k	FR	time	FR	time
-3	30	124.9	30	81.2
-2	30	135.9	28	82.6
-1	1	136.7	28	101.6
0	0	-	29	87.3
1	0	-	30	103.4
min	0	124.9	28	81.2
max	30	136.7	30	103.4
mean	12	132.5	29	91.2

(c) AT4: *speed* $\times 10^k$

	Breach		ForeSee	
k	FR	time	FR	time
-2	29	247.2	29	329.4
-1	29	243.5	28	332.2
0	8	414.0	30	376.6
1	0	-	29	377.6
2	0	-	29	333.2
min	0	243.5	28	329.4
max	29	414.0	30	377.6
mean	13	301.6	29	349.8

(d) AT9: *speed* $\times 10^k$

	Breach		ForeSee	
k	FR	time	FR	time
-1	11	202.6	28	259.8
0	9	298.6	29	208.0
1	10	197.4	29	221.2
2	28	175.4	29	248.9
3	30	162.6	29	209.6
min	9	162.6	28	208.0
max	30	298.6	29	259.8
mean	18	207.3	29	229.5

(e) AT15: *rpm* $\times 10^k$

	Breach		ForeSee	
k	FR	time	FR	time
-5	20	138.3	6	222.3
-4	13	158.1	10	258.8
-3	4	144.6	5	313.6
-2	0	-	9	268.6
0	1	66.1	9	331.8
min	0	66.1	5	222.3
max	20	158.1	10	331.8
mean	10	126.8	8	279.0

(f) AFC3: *EngineSpeed* $\times 10^k$

	Breach		ForeSee	
k	FR	time	FR	time
0	8	727.6	28	232.5
-1	18	574.2	29	284.1
-2	29	401.2	29	211.5
-3	30	215.0	29	230.1
-4	29	198.2	30	236.2
min	8	198.2	28	211.5
max	30	727.7	30	284.1
mean	23	423.2	29	238.9

and AT9; the *rpm* of AT15; and the *EngineSpeed* of AFC3. For each artificial rescaling, both the Simulink model and the specification have been changed.[5] We run ForeSee and Breach on these rescaled benchmarks. Table 3 reports the experimental results for each k. The table also reports the minimum, maximum, and mean results for FR and execution time. We observe that the performance of Breach, in terms of FR, is very sensitive to the scale problem. Indeed, for all the specifications, FR decreases with increasing or decreasing k; notable examples are AT3 and AT4 in which Breach can (almost) always falsify with the minimum k, but never falsifies with the maximum two k. This is the demonstration of the effects of the scale problem on falsification approaches that only rely on quantitative robust semantics where the robustness values of different signals are compared. By looking at the results of ForeSee, instead, we observe that it is much more robust and its FR performance is independent of the applied rescaling. This clearly shows that our falsification approach guided by QB-Robustness is successful in avoiding the scale problem.

[5] Note that $k = 0$ corresponds to the experimental result in Table 2, and we report it again for reference.

Table 4. Falsification performance under different MCTS hyperparameters.

(a) Performance with varied c

	AT17		AT19		AT21	
c	FR	time	FR	time	FR	time
0	23	177.8	30	224.6	30	463.4
0.02	26	196.7	30	278.5	28	501.3
0.2	30	208.8	30	154.3	29	425.5
0.5	30	297.0	29	227.3	30	509.0
1.0	30	311.7	30	240.2	24	497.0

(b) Performance with varied B_P

	AT17		AT19		AT21	
B_P	FR	time	FR	time	FR	time
2	26	385.2	30	162.0	29	500.0
5	30	347.7	29	207.3	29	472.5
10	30	208.8	30	154.3	29	425.5
15	30	337.7	29	336.7	28	514.0
20	30	358.1	30	313.5	30	511.0

These results also allow us to show that the naive approach based on normalization for solving the scale problem does not work, as also reported in [41]. Indeed, one may think that a solution for tackling the scale problem could be to rescale the signals in a way to make them have the same order of magnitude. This is not a good approach. Let us consider the results in Table 3c for AT4 ($\Box_{[0,30]}$ (*speed* $< 135 \wedge$ *rpm* < 4780)). In this case, *speed* is multiplied by 10^k. We may think that the best falsification result should occur when *speed* is multiplied by 10^2, because this would make the two signals both in the order of thousands. However, this rescaling is the one giving the worst result. The best result is actually given by the rescaling making *speed* even smaller (i.e., $k = -2$ and $k = -1$). This means that the correct way for handling the scale problem cannot be identified in advance, but we need an approach as ours that *learns* during falsification the best strategy.

RQ3. *How do the hyperparameters of MCTS influence the performance of the proposed approach?*

Our proposed approach is an instantiation of the Monte Carlo Tree Search (MCTS) method [8,28] that can be configured with some hyperparameters, namely the scalar c used by UCB1 (Line 12 in Algorithm 2), and the playout budget B_P (Line 21 in Algorithm 2), both used for balancing between exploration and exploitation. Therefore, the performance of MCTS could be affected by the values used for these hyperparameters. In this RQ, we try to assess this. We selected three benchmarks specifications (AT17, AT19, and AT21) and varied one hyperparameter while keeping the other fixed. Namely, we experimented with $c \in \{0, 0.02, 0.2, 0.5, 1\}$ and budget $B_P = 10$ (see Table 4a), and with $B_P \in \{2, 5, 10, 15, 20\}$ and budget $c = 0.2$ (see Table 4b). Looking at the results of Table 4a for AT17 and AT21, there seems to be some influence by the scalar c. In AT17, the worst result in terms of FR is obtained when c is 0, meaning that MCTS only focuses on *exploitation*. AT17 is a specification that suffers from the scale problem, as shown by the very bad performance of `Breach` in Table 2; for such a specification, we need to perform some *exploration* to find the best Σ: this explain the low performance of MCTS with $c = 0$. On the other hand, the worst FR performance of AT21 is given by the highest value $c = 1$ that requires MCTS to spend a lot of time in exploration. Since AT21 is not an extremely difficult specification (indeed `Breach` has FR of 10 in Table 2), such very conservative approach does not pay off, while more greedy approaches (i.e., with lower c) have better performance.

Looking at the results of Table 4b related to B_P, it seems that there is no too much influence. The only difference is given in AT17 with $B_P = 2$ where the FR is slightly lower than the other cases. This means that, provided that a sufficiently large value for B_P is given, `ForeSee` is not too sensitive to it.

6 Related Work

Quality assurance of CPS has been actively studied, due to its great significance. Different approaches, including but not limited to model checking, theorem proving, rigorous numerics, and nonstandard analysis [9,16,20,22,23,31,33], have been proposed to solve the problem. However, due to the scalability issue and existence of black-box components, those approaches are not widely applied in the real-world systems.

The optimization-based falsification approach inherits the search-based testing methodology, and is much more scalable than pure verification-based approaches. The key issue of search-based testing is the *exploration-exploitation trade-off*. This issue has been discussed for the verification of quantitative properties (e.g., [34]). In the falsification community, there have also been a lot of works focusing on that, and these works tackle the problem from different perspectives. *Metaheuristics* refers to high-level heuristic strategies that utilize heuristics to improve the search efficiency. Several metaheuristic strategies have been applied in falsification, such as *Simulated Annealing* [1], *tabu search* [10], and so on. *Coverage-guided falsification* [2,10,15,29] aims to guide the search using some coverage metrics, so that the search space is sufficiently explored. Recently, machine learning techniques have also been applied to falsification to enhance the search ability. For instance, *Bayesian optimization* [3,11,36] utilizes an *acquisition function* to balance exploration and exploitation; *Reinforcement learning* [27,37] naturally emphasizes on exploration.

The scale problem is a recognized issue [12,21,40] that is known to severely affect the performance of falsification. In [40], we proposed a multi-armed bandit approach to solve the problem in a specific setting, that is, safety properties with Boolean connectives: $\Box_I\ (\varphi_1 \wedge \varphi_2)$ and $\Box_I\ (\varphi_1 \vee \varphi_2)$. The approach is not applicable to formulas having more nested sub-formulas, or even connectives having more operands; therefore many of the benchmarks we used in Subsect. 5.2 fall out of the scope of [40]. The techniques introduced in [12,21] rely on explicit declaration of *input vacuity* and *output robustness*. Compared to their approaches, our method does not need that, but we learn the significance of each signal through tree exploration and reward computation.

MCTS, as an effective search framework, has been applied in testing hybrid systems. In [30], the authors applied an adaption of MCTS in testing, namely, adaptive press testing, to detect the potential dangerous cases of airborne collision. A recent study of MCTS on hybrid system falsification is [39]. There, the authors discretized the search space to construct the search tree, and then applied MCTS to explore different sub-spaces. Compared to their approach, our work aims to tackle the scale problem and so we exploit the structure of specification formulas to construct the tree search framework.

7 Conclusion and Future Work

Optimization-based falsification is a widely used approach for quality assurance of CPS, that tries to find an input violating a Signal Temporal Logic (STL) specification. It does this by exploiting the quantitative robust semantics of the specification, trying to minimize its robustness. The performance of falsification is affected by the *scale problem* in the presence of the comparison of robustness values of sub-formulas predicating over signals having different scales. In this paper, we propose QB-Robustness, a new STL semantics that does not suffer from the scale problem, because it avoids such comparison. The computation of QB-Robustness requires to specify a sub-formula sequence telling for which sub-formulas the quantitative robustness must be computed. We then propose a Monte Carlo Tree Search (MCTS)-based falsification approach that synthesizes a sub-formula sequence for QB-Robustness, and uses this for guiding numerical optimization. Experimental results show that the proposed approach achieves better falsification results than a state-of-the-art falsification tool that uses standard quantitative robust semantics.

In the analysis of RQ1, we observed that, when the specifications have a particular structure, our approach has no advantage and, actually, it could decrease the performance by trying to find a best sub-formula sequence that does not exist for the current initial sampling. As future work, we plan to devise some heuristics that could handle these cases: for example, we could perform a better initial sampling (see Subsect. 2.1) that could provide a better initial guidance.

References

1. Abbas, H., Fainekos, G.: Convergence proofs for simulated annealing falsification of safety properties. In: 50th Annual Allerton Conference on Communication, Control, and Computing, pp. 1594–1601. IEEE (2012)
2. Adimoolam, A., Dang, T., Donzé, A., Kapinski, J., Jin, X.: Classification and coverage-based falsification for embedded control systems. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 483–503. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_24
3. Akazaki, T., Kumazawa, Y., Hasuo, I.: Causality-aided falsification. In: Proceedings First Workshop on Formal Verification of Autonomous Vehicles, FVAV@iFM 2017, Turin, Italy, 19th September 2017. EPTCS, vol. 257, pp. 3–18 (2017)
4. Annpureddy, Y., Liu, C., Fainekos, G., Sankaranarayanan, S.: S-TaLiRo: a tool for temporal logic falsification for hybrid systems. In: Abdulla, P.A., Leino, K.R.M. (eds.) TACAS 2011. LNCS, vol. 6605, pp. 254–257. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19835-9_21

5. Arcuri, A., Briand, L.: A practical guide for using statistical tests to assess randomized algorithms in software engineering. In: Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, pp. 1–10. ACM New York (2011)
6. Auer, P.: Using confidence bounds for exploitation-exploration trade-offs. J. Mach. Learn. Res. **3**, 397–422 (2002)
7. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005, pp. 1769–1776. IEEE (2005)
8. Browne, C., et al.: A survey of monte carlo tree search methods. IEEE Trans. Comput. Intellig. AI Games **4**(1), 1–43 (2012)
9. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: an analyzer for non-linear hybrid systems. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 258–263. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_18
10. Deshmukh, J., Jin, X., Kapinski, J., Maler, O.: Stochastic local search for falsification of hybrid systems. In: Finkbeiner, B., Pu, G., Zhang, L. (eds.) ATVA 2015. LNCS, vol. 9364, pp. 500–517. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24953-7_35
11. Deshmukh, J.V., Horvat, M., Jin, X., Majumdar, R., Prabhu, V.S.: Testing cyber-physical systems through bayesian optimization. ACM Trans. Embedded Comput. Syst. **16**(5), 170:1–170:18 (2017)
12. Dokhanchi, A., Yaghoubi, S., Hoxha, B., Fainekos, G.E.: Vacuity aware falsification for MTL request-response specifications. In: 13th IEEE Conference on Automation Science and Engineering, CASE 2017, Xi'an, China, 20–23 August 2017, pp. 1332–1337. IEEE (2017)
13. Donzé, A.: Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 167–170. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14295-6_17
14. Donzé, A., Maler, O.: Robust satisfaction of temporal logic over real-valued signals. In: Chatterjee, K., Henzinger, T.A. (eds.) FORMATS 2010. LNCS, vol. 6246, pp. 92–106. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15297-9_9
15. Dreossi, T., Dang, T., Donzé, A., Kapinski, J., Jin, X., Deshmukh, J.V.: Efficient guiding strategies for testing of temporal properties of hybrid systems. In: Havelund, K., Holzmann, G., Joshi, R. (eds.) NFM 2015. LNCS, vol. 9058, pp. 127–142. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17524-9_10
16. Dreossi, T., Dang, T., Piazza, C.: Parallelotope bundles for polynomial reachability. In: Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control, HSCC 2016 pp. 297–306. ACM, New York (2016)
17. Dreossi, T., Donzé, A., Seshia, S.A.: Compositional falsification of cyber-physical systems with machine learning components. In: Barrett, C., Davies, M., Kahsai, T. (eds.) NFM 2017. LNCS, vol. 10227, pp. 357–372. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57288-8_26
18. Ernst, G., et al.: 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)
19. Fainekos, G.E., Pappas, G.J.: Robustness of temporal logic specifications for continuous-time signals. Theor. Comput. Sci. **410**(42), 4262–4291 (2009)

20. Fan, C., Qi, B., Mitra, S., Viswanathan, M., Duggirala, P.S.: Automatic reachability analysis for nonlinear hybrid models with C2E2. In: Chaudhuri, S., Farzan, A. (eds.) CAV 2016. LNCS, vol. 9779, pp. 531–538. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41528-4_29
21. 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, 16–18 April 2019, pp. 57–66 (2019)
22. Frehse, G., et al.: SpaceEx: scalable verification of hybrid systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 379–395. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_30
23. Hasuo, I., Suenaga, K.: Exercises in nonstandard static analysis of hybrid systems. In: Madhusudan, P., Seshia, S.A. (eds.) CAV 2012. LNCS, vol. 7358, pp. 462–478. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31424-7_34
24. Hoxha, B., Abbas, H., Fainekos, G.E.: Benchmarks for temporal logic requirements for automotive systems. In: 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems, ARCH@CPSWeek 2014, Berlin, Germany, 14 April 2014/ARCH@CPSWeek 2015, Seattle, USA, 13 April 2015. EPiC Series in Computing, vol. 34, pp. 25–30. EasyChair (2014)
25. Jin, X., Deshmukh, J.V., Kapinski, J., Ueda, K., Butts, K.: Powertrain control verification benchmark. In: Proceedings of the 17th Int. Conf. on Hybrid Systems: Computation and Control, HSCC 2014, pp. 253–262. ACM, NY, USA (2014)
26. 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 Syst. **36**(6), 45–64 (2016)
27. Kato, K., Ishikawa, F.: Learning-based falsification for model families of cyber-physical systems. In: 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), pp. 236–245 (December 2019)
28. Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In: Fürnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) ECML 2006. LNCS (LNAI), vol. 4212, pp. 282–293. Springer, Heidelberg (2006). https://doi.org/10.1007/11871842_29
29. Kuřátko, J., Ratschan, S.: Combined global and local search for the falsification of hybrid systems. In: Legay, A., Bozga, M. (eds.) FORMATS 2014. LNCS, vol. 8711, pp. 146–160. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10512-3_11
30. Lee, R., Kochenderfer, M.J., Mengshoel, O.J., Brat, G.P., Owen, M.P.: Adaptive stress testing of airborne collision avoidance systems. In: Digital Avionics Systems Conference, 2015 IEEE/AIAA 34th, pp. 6C2-1. IEEE (2015)
31. Liebrenz, T., Herber, P., Glesner, S.: Deductive verification of hybrid control systems modeled in Simulink with KeYmaera X. In: Sun, J., Sun, M. (eds.) ICFEM 2018. LNCS, vol. 11232, pp. 89–105. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02450-5_6
32. Luersen, M.A., Le Riche, R.: Globalized Nelder-mead method for engineering optimization. Comput. Struct. **82**(23), 2251–2260 (2004)
33. Platzer, A.: Logical Foundations of Cyber-Physical Systems. Springer, Cham (2018) https://doi.org/10.1007/978-3-319-63588-0
34. Seshia, S.A., Rakhlin, A.: Quantitative analysis of systems using game-theoretic learning. ACM Trans. Embed. Comput. Syst. **11**(S2), 55:1–55:27 (2012)
35. Silver, D., et al.: Mastering the game of Go with deep neural networks and tree search. Nature **529**, 484–489 (2015)

36. Silvetti, S., Policriti, A., Bortolussi, L.: An active learning approach to the falsification of black box cyber-physical systems. In: Polikarpova, N., Schneider, S. (eds.) IFM 2017. LNCS, vol. 10510, pp. 3–17. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66845-1_1
37. Akazaki, T., Liu, S., Yamagata, Y., Duan, Y., Hao, J.: Falsification of cyber-physical systems using deep reinforcement learning. In: Havelund, K., Peleska, J., Roscoe, B., de Vink, E. (eds.) FM 2018. LNCS, vol. 10951, pp. 456–465. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-95582-7_27
38. Zhang, Z., Arcaini, P., Hasuo, I.: Hybrid system falsification under (in)equality constraints via search space transformation. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **39**(11), 3674–3685 (2020)
39. Zhang, Z., Ernst, G., Sedwards, S., Arcaini, P., Hasuo, I.: Two-layered falsification of hybrid systems guided by monte carlo tree search. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **37**(11), 2894–2905 (2018)
40. Zhang, Z., Hasuo, I., Arcaini, P.: Multi-armed bandits for Boolean connectives in hybrid system falsification. In: Dillig, I., Tasiran, S. (eds.) CAV 2019. LNCS, vol. 11561, pp. 401–420. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25540-4_23
41. Zhang, Z., Lyu, D., Arcaini, P., Ma, L., Hasuo, I., Zhao, J.: On the effectiveness of signal rescaling in hybrid system falsification. In: Dutle, A., Moscato, M.M., Titolo, L., Muñoz, C.A., Perez, I. (eds.) NFM 2021. LNCS, vol. 12673, pp. 392–399. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-76384-8_24
42. Zutshi, A., Deshmukh, J.V., Sankaranarayanan, S., Kapinski, J.: Multiple shooting, cegar-based falsification for hybrid systems. In: 2014 International Conference on Embedded Software, EMSOFT 2014, New Delhi, India, 12–17 October 2014, pp. 5:1–5:10. ACM (2014)