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

Zhenya Zhang, Deyun Lyu, Paolo Arcaini, Lei Ma, Ichiro Hasuo, Jianjun Zhao

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.

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 $u$ and outputting a signal $\mathcal{M}(u)$, and a specification $\varphi$ (a temporal formula), the falsification problem consists in finding a falsifying input, i.e., an input signal $u$ such that the corresponding output $\mathcal{M}(u)$ violates $\varphi$.

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

This work was originally published at 33rd International Conference on Computer-Aided Verification. [41]

Zhenya Zhang, Deyun Lyu, Jianjun Zhao, 九州大学システム情報科学研究科, Graduate School of Information Science and Electrical Engineering, Kyushu University.
Paolo Arcaini, Ichiro Hasuo, 国立情報学研究所, National Institute of Informatics.
Lei Ma, University of Alberta, Canada.
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 $[w, \varphi] \in \mathbb{R} \cup \{-\infty, +\infty\}$ (i.e., robustness) that tells not only whether $\varphi$ 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 $\varphi$. 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 $\varphi \equiv \Box_{[0,30]} (\varphi_1 \land \varphi_2)$, with $\varphi_1 \equiv \text{gear} < 6$ and $\varphi_2 \equiv \text{speed} < 130$. It is apparent that $\varphi_1$ is always satisfied (in any car model with 5 gears), and it has been added in the specification as redundant check.‡ According to robust semantics, the Boolean connective $\land$ is interpreted by minimum $\land$, and the “always” operator $\Box_{[0,30]}$ is interpreted by infimum $\prod$; the robustness of an atomic formula $f(\vec{x}) < c$ is given by the margin $c - f(\vec{x})$. Therefore, the robustness of $\varphi$ under the signal (gear, speed), where gear, speed: $[0,30] \rightarrow \mathbb{R}$, is $\left(\prod_{t \in [0,30]} \left((6 - \text{gear}(t)) \cup (130 - \text{speed}(t))\right)\right)$. Note that the robustness of $\varphi_1$ is always in the order of units, while the robustness of $\varphi_2$ is, in general, in the order of tens. It is not difficult to see that, if both $\varphi_1$ and $\varphi_2$ are satisfied, the robustness of $\varphi$ will only depend on $\varphi_1$ (because of the minimum in the robust semantics of the logical connective). In this case, we say that $\varphi_1$ masks $\varphi_2$. 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 $\varphi$ is to falsify $\varphi_2$, because $\varphi_1$ 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 $\varphi$, requires selecting a sub-formula $\varphi_k$ among its sub-formulas $\{\varphi_1, \ldots, \varphi_K\}$. For $\varphi_k$, 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 $\Sigma$ along the parse tree of the formula $\varphi$, where visited sub-formulas are those for which the quantita-

‡ 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.
tive robustness is computed. We prove that QB-Robustness, independently of the selected $\Sigma$, 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 subformulas 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 $\Sigma$ of subformulas 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 $\Sigma$ contains $\varphi_1$, we can encounter the problem that the quantitative robustness of $\varphi_1$ would not provide any guidance (i.e., no big variations in the robustness values would be observed). On the other hand, if $\Sigma$ contains $\varphi_2$, the quantitative robustness would have larger variations, providing more effective guidance to the search.

Then, the key problem is how to select the best $\Sigma$, that enables the hill-climbing optimization used in falsification to be more effective. In general, although it is often difficult to know the best $\Sigma$ in advance, it is still possible to learn it by observing sampling results using different $\Sigma$. Based on this intuition, we propose a novel falsification approach that identifies the sequences $\Sigma$ 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 $\Sigma$ and running optimization-based falsification in which QB-Robustness with $\Sigma$ 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 §2, we introduce the preliminaries of the optimization-based falsification. In §3, we introduce the novel STL semantics QB-Robustness, and, in §4, we describe the MCTS-based falsification approach that uses QB-Robustness. In §5, we describe the experiments and evaluation results. Finally, we discuss most relevant work to ours in §6, and conclude the paper in §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 $w: [0,T] \rightarrow \mathbb{R}^M$. 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 $M$ that takes an input signal $u: [0,T] \rightarrow \mathbb{R}^M$ and returns a signal $M(u): [0,T] \rightarrow \mathbb{R}^N$. Here the common time
We fix a set function initialize a placeholder while rb H a system model return rb u

Let J the following sense: The robust semantics refines the Boolean one in a binary relation J respectively: w

Definition 3 (Quantitative Robust Semantics)

Let w: [0, T] → R^N be an N-dimensional signal, and t ∈ [0, T). The t-shift w' of w is the signal w': [0, T - t] → R^N defined by w'(t') := w(t + t'). Let φ be an STL formula. We define the robustness [w, φ] ∈ R∪{∞} as follows, by induction on the construction of formulas. ∩ and ∪ denote infimaums and supremasums of real numbers, respectively. ∩, the binary version of ∩, denotes minimum.

[w, f(x_1, ..., x_n) > 0] := [f(w(0)(x_1), ..., w(0)(x_n))]
[w, ⊥] := −∞
[w, ¬φ] := −[w, φ]
[w, φ_1 ∨ φ_2] := [w, φ_1] ∪ [w, φ_2]
[w, φ_1 ∧ φ_2] := [φ_1](0) ∩ [t ∈ [0, T], w(t) ∈ [w, φ_1] ∩ [t ∈ [0, T], w(t) ∈ [w, φ_2]]

The original STL semantics is Boolean, given by a binary relation |= between signals and formulas. The robust semantics refines the Boolean one in the following sense: [w, φ] > 0 implies w |= φ, and [w, φ] < 0 implies w ̸|= φ, 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, 43] 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 4 (Falsifying Input) Let M be a system model, and φ be an STL formula. A signal u: [0, T] → R^|Var| is a falsifying input if [M(u), φ] < 0; the latter implies M(u) ̸|= φ.

The use of quantitative robust semantics [M(u), φ] ∈ R ∪ (−∞, −∞) in the above problem enables the use of hill-climbing optimization.

Definition 5 (Hill Climbing-Guided Falsification)

Assume the setting in Def. 4, for finding a falsifying input, the methodology of hill climbing-guided falsification is presented in Algo. 1.

Algorithm 1 Hill climbing-guided falsification

Require: a system model M, an STL formula φ, and a time budget
1: function Hill-Climb-Falsify(M, φ)
2: initialize a placeholder u and rb ← ∞ to the best input signal and robustness
3: H ← ∅ to sampling history of input signals and robustness
4: while rb ≥ 0 and within the time budget do
5: u' ← Hill-Climb(H) to run hill climbing based on sampling history
6: rb' ← [M(u'), φ] to compute robustness
7: H ← H ∪ {(u', rb')} to update sampling history
8: if rb' < rb then
9: rb ← rb', u ← u' to update the best input and robustness
10: return u if rb < 0 Failure otherwise

Here, the function Hill-Climb makes a guess of an input signal u', aiming at minimizing the robustness [M(u'), φ]. 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.

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 Def. 3), 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 $w$ may always (or often) mask the contribution of the others, and, therefore, the final robustness may be dominated by this signal $w$. 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 §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 $\varphi_1 U \varphi_2$, when $\varphi_1$ is an arbitrary formula. We still support it as “eventually” and “always” operators\*\*\*, 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 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 6 (Immediate Sub-Formulas)** Let $\varphi$ be an STL formula (see Def. 2). We define the set $\text{ISForm}(\varphi)$ of immediate sub-formulas of $\varphi$ as follows:

\[
\begin{align*}
\text{ISForm}(\alpha) & := \emptyset & \text{ISForm}(\bot) & := \emptyset \\
\text{ISForm}(\neg \varphi) & := \text{ISForm}(\varphi) \\
\text{ISForm}\left( \bigwedge_{i \in \{1, \ldots, K\}} \varphi_i \right) & := \{\varphi_1, \ldots, \varphi_K\} \\
\text{ISForm}\left( \bigvee_{i \in \{1, \ldots, K\}} \varphi_i \right) & := \{\varphi_1, \ldots, \varphi_K\} \\
\text{ISForm}(\square_I \varphi) & := \text{ISForm}(\varphi) \\
\text{ISForm}(\Diamond_I \varphi) & := \text{ISForm}(\varphi)
\end{align*}
\]

\*\*\* Recall from Def. 2 that the “eventually” and “always” operators are defined in terms of the “until” operator.
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 7 (Sub-Formula Sequence)** Let $\varphi$ be an STL formula. A sub-formula sequence $\Sigma = \sigma_1 \cdot \ldots \cdot \sigma_L$ w.r.t. $\varphi$ is defined as follows:

$$
\sigma_1 \in \text{ISForm}(\varphi)
$$

$$
\sigma_{l+1} \in \text{ISForm}(\sigma_l) \quad \text{with} \ l = 1, \ldots, L - 1
$$

where the $\cdot$ is the concatenation operator in the sequence. We use $\Sigma_{k}$ to denote the $k$th element of $\Sigma$. Moreover, we denote the first element by $\Sigma_{\text{head}}$, and the last element by $\Sigma_{\text{rear}}$. We use $\Sigma_{\text{read}}$ to denote $\Sigma$ without $\Sigma_{\text{head}}$. We identify with $\varepsilon$ as its sub-formula sequence. We identify with $\Sigma_{\varphi}$ the set of all the sub-formula sequences rooted in $\varphi$.

To be specific, in a sub-formula sequence $\Sigma$, 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 predicing over a single signal can only appear as the final element of a sequence. We exploit these characteristics of $\Sigma$ 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 $\Sigma$, decides whether to compute the quantitative robust semantics or the Boolean semantics of a sub-formula, by considering whether the sub-formula belongs to $\Sigma$ or not. This implies that, in the case of conjunction and disjunction, we evaluate the quantitative robustness of the sub-formula in $\Sigma$ and the Boolean satisfaction of the other sub-formulas. Based on such intuition, we define the semantics of our proposed QB-Robustness in Def. 8, and demonstrate its usefulness in Thm. 9.

**Definition 8 (Semantics of QB-Robustness)**

Let $\varphi$ be an STL formula as defined in Def. 2, and $\Sigma$ be a sub-formula sequence w.r.t. $\varphi$. For $\varphi \equiv \land \varphi_i \lor \varphi_i$, let $\varphi_k \in \text{ISForm}(\varphi)$ be the first element $\Sigma_{\text{head}}$ of $\Sigma$, then we can represent these two cases as $\varphi \equiv \varphi_k \land \varphi_k' \lor \varphi_k \lor \varphi_k'$, where $\varphi_k'$ is the conjunction (or disjunction, respectively) of the other formulas in $\text{ISForm}(\varphi) \setminus \{\varphi_k\}$. The QB-Robustness $\text{QBRob}(w, \varphi, \Sigma)$ of $\varphi$ w.r.t. $\Sigma$ is defined as Fig. 1:

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 9** Let $\varphi$ be an STL formula. Given a signal $w$, for any $\Sigma \in \Sigma_{\varphi}$, it holds that $\text{QBRob}(w, \varphi, \Sigma) > 0$ implies $w \models \varphi$. Similarly, for any $\Sigma \in \Sigma_{\varphi}$, it holds that $\text{QBRob}(w, \varphi, \Sigma) < 0$ implies $w \not\models \varphi$.

**Proof 10** We first recall from [19, Prop. 16] that $[w, \varphi] < 0$ implies $w \not\models \varphi$, and that $[w, \varphi] > 0$ implies $w \models \varphi$. We prove Thm. 9 by induction on the structure of the formula.

- Case $\varphi = \alpha$. By Def. 8, $\text{QBRob}(w, \alpha, \varepsilon) > 0$ indicates that $[w, \alpha] > 0$ and hence $w \models \alpha$, and $\text{QBRob}(w, \alpha, \varepsilon) < 0$ that $[w, \alpha] < 0$ and hence $w \not\models \varphi$.

- For the following cases, let us assume that Thm. 9 holds for an arbitrary formula $\varphi'$ and its sub-formula sequence $\Sigma'$ that $\text{QBRob}(w, \varphi', \Sigma') > 0$ implies $[w, \varphi'] > 0$, and that $\text{QBRob}(w, \varphi', \Sigma') < 0$ implies $[w, \varphi'] < 0$.

We aim to prove that Thm. 9 also holds for $\varphi$, resulting from the application of the operator in each of the following cases to $\varphi'$, and $\Sigma$, the
Case \( \varphi = \varphi' \land \psi \), where \( \psi \) is an arbitrary formula. Let \( \Sigma = \varphi' : \Sigma' \), and let us consider the two cases in which \( \text{QBRob}(w, \varphi, \Sigma) \) is negative and positive separately:

* If \( \text{QBRob}(w, \varphi, \Sigma) < 0 \), there are two sub-cases:
  * if \( \text{QBRob}(w, \varphi', \Sigma') < 0 \), then \([w, \varphi'] < 0 \) (by assumption).
    Then, by the robust semantics of conjunction, also \([w, \varphi] < 0 \) holds, and so it does \( w \not\vDash \varphi \).
  * if \( \text{QBRob}(w, \varphi', \Sigma') > 0 \), then \([w, \varphi'] > 0 \) (by assumption).
    Then, it holds \( w \not\vDash \psi \) by Def. 8, and, therefore, it holds \( w \not\vDash \varphi \).

* If \( \text{QBRob}(w, \varphi, \Sigma) > 0 \), it means that \( \text{QBRob}(w, \varphi', \Sigma') > 0 \) and \( w \vDash \psi \) (by Def. 8). By assumption, if \( \text{QBRob}(w, \varphi', \Sigma') > 0 \), then \([w, \varphi'] > 0 \).
  Therefore, \( w \vDash \varphi \).

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

* By Def. 8, \( \text{QBRob}(w, \varphi, \Sigma) < 0 \) indicates that there exists a \( t \in I \) such that \( \text{QBRob}(w', \varphi', \Sigma') < 0 \). By assumption, it holds that \( w' \not\vDash \varphi' \). Then, by the semantics of the always operator \( \Box \), it holds that \( w \not\vDash \varphi \).

* By Def. 8, \( \text{QBRob}(w, \varphi, \Sigma) > 0 \) indicates that for all \( t \in I \) it holds that \( \text{QBRob}(w', \varphi', \Sigma') > 0 \). Then, by assumption, it holds that for all \( t \in I \), \( w' \vDash \varphi' \). So, by the semantics of the always operator \( \Box \), it holds that \( w \not\vDash \varphi \).

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

* By Def. 8, \( \text{QBRob}(w, \varphi, \Sigma) < 0 \) indicates that \( \text{QBRob}(w, \varphi', \Sigma') < 0 \) and \( w \not\vDash \varphi' \) (by Def. 8). By assumption, it holds that \( w \vDash \varphi' \), and, therefore, \( w \not\vDash \varphi \).

* By Def. 8, \( \text{QBRob}(w, \varphi, \Sigma) > 0 \) indicates that \( \text{QBRob}(w, \varphi', \Sigma') > 0 \). By assumption, it holds that \( w \not\vDash \varphi' \). Then, by the semantics of the always operator \( \Box \), it holds that \( w \not\vDash \varphi \).

- Proofs for the cases of \( \varphi = \varphi' \lor \psi \) and \( \varphi = \varnothing_f \varphi' \) 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 11** Let \( w : \[0, T \] \rightarrow \mathbb{R}^2 \) be a 2-dimensional signal and \( \varphi = \Box_f (\varphi_1 \lor \varphi_2) \) be an STL formula where \( \varphi_1 \) and \( \varphi_2 \) are two atomic formu-
las. Intuitively, to make $\varphi$ falsified, there must exist $t \in I$ such that $w^t \not \models \varphi_1$ and $w^t \not \models \varphi_2$. Let us consider a non-trivial falsification problem in which, for most of the signals $w$, sets $\{ t \in I \mid w^t \not \models \varphi_1 \}$ and $\{ t \in I \mid w^t \not \models \varphi_2 \}$ are non-empty and disjoint.

By Def. 8, given the sub-formula sequence $\Sigma = \varphi_1$ of $\varphi$, the corresponding QB-Robustness is $\text{QBRob}(w, \varphi, \Sigma) = \prod_{t \in I} \text{QBRob}(w^t, \varphi_1 \lor \varphi_2, \varphi_1)$, i.e., it takes the infimum of $\text{QBRob}(w^t, \varphi_1 \lor \varphi_2, \varphi_1)$ over $t \in I$. Again, by Def. 8, for any $t' \in I$, $\text{QBRob}(w', \varphi_1 \lor \varphi_2, \varphi_1)$ is computed as follows:

- if for a $t' \in I$ it holds $w^{t'} \models \varphi_2$, then $\text{QBRob}(w', \varphi_1 \lor \varphi_2, \varphi_1) = \infty$. Then, it is impossible that $\prod_{t \in I} \text{QBRob}(w^t, \varphi_1 \lor \varphi_2, \varphi_1)$ is given by $\text{QBRob}(w'^t, \varphi_1 \lor \varphi_2, \varphi_1)$;
- if for a $t' \in I$ it holds $w^{t'} \not \models \varphi_2$, then $\text{QBRob}(w', \varphi_1 \lor \varphi_2, \varphi_1) = \text{QBRob}(w'^t, \varphi_1, \epsilon) = [w'^t, \varphi_1]$. In this case, $\text{QBRob}(w', \varphi_1 \lor \varphi_2, \varphi_1)$ has a chance to determine the value of $\prod_{t \in I} \text{QBRob}(w^t, \varphi_1 \lor \varphi_2, \varphi_1)$.

Therefore, when $\Sigma = \varphi_1$, it holds that $\text{QBRob}(w, \varphi, \Sigma) = \prod_{t \in S} [w^t, \varphi_1]$, where $S = \{ t \in I \mid w^t \not \models \varphi_2 \}$, i.e., the infimum of the quantitative robustness of $\varphi_1$ on the interval when $\varphi_2$ is violated. Indeed, once this value is negative, it means that there exists a point $t \in I$ when both $\varphi_1$ and $\varphi_2$ are violated; by the Boolean semantics of always and disjunction, $\varphi$ 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 $\Sigma$ is required when calculating QB-Robustness, and such sequence is not unique (see Def. 8). Note that the selection of the sequence can affect the performance of the numerical optimization algorithms used in falsification. Let us consider $\varphi \equiv \Box ((\text{gear} < 6) \land (\text{speed} < 130))$ as an example. As explained in §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 $\Sigma$.

By using the QB-Robustness, the problem of falsifying an STL formula $\varphi$ consists in finding both a signal $w$ and a sub-formula sequence $\Sigma$ such that $\text{QBRob}(w, \varphi, \Sigma) < 0$. The selection of $\Sigma$ is discrete, while the search for $w$ 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 §4.1, and then present the application of MCTS to our falsification problem in §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. 2, 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 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 Algo. 2 and visualize its execution in Fig. 2. In the following, we elaborate on our approach.

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 $\varphi$ 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.

#### 4.2.0.1 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).

#### 4.2.0.2 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), initializing 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
Algorithm 2 MCTS-based falsification guided by QB-Robustness

Require: a system model \( M \), an STL formula \( \varphi \), 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: \( \Sigma^{\text{init}} \leftarrow \varphi \)\quad \triangleright \text{the root denoted as a sequence with } \varphi \text{ only}
3: \( \mathcal{T} \leftarrow \{ \Sigma^{\text{init}} \} \)\quad \triangleright \text{the MCTS search tree, initially root only}
4: \( N \leftarrow (\Sigma^{\text{init}} \rightarrow 0) \)\quad \triangleright \text{visit count } N \text{ initialized, defined only for root}
5: \( R \leftarrow (\Sigma^{\text{init}} \rightarrow 0) \)\quad \triangleright \text{reward function } R \text{ initialized}
6: \( H \leftarrow (\Sigma^{\text{init}} \rightarrow \emptyset) \)\quad \triangleright \text{the sampling history of hill climbing}
7: while \( \varphi \) not falsified and within the MCTS budget \( B_M \) do
8: \hspace{1em} MCTSSearch(\( \Sigma^{\text{init}} \))
9: function MCTSSearch(\( \Sigma \))
10: if ISForm(\( \Sigma \)) \&\& \( \Sigma \nk = \emptyset \) then \quad \triangleright \text{the node has children}
11: \hspace{1em} if \( \Sigma \varphi \mathcal{T} \text{ for all } \varphi \mathcal{T} \) ISForm(\( \Sigma \)) \text{ then} \quad \triangleright \text{all children expanded}
12: \hspace{2em} \varphi_k \leftarrow \arg \max_{\varphi \in \text{ISForm}(\( \Sigma \))} \left( R(\Sigma \varphi) + c \sqrt{\frac{2 \ln N(\Sigma)}{N(\Sigma \varphi)}} \right) \quad \triangleright \text{selection by UCB1}
13: \hspace{1em} else \quad \triangleright \text{unexpanded children exist}
14: \hspace{2em} randomly select \( \varphi_k \) from \( \{ \varphi \in \text{ISForm}(\( \Sigma \)) \mid \Sigma \varphi \mathcal{T} \} \)
15: \hspace{2em} \mathcal{T} \leftarrow \mathcal{T} \cup \{ \Sigma \varphi \} \quad \triangleright \text{expand a new child}
16: \hspace{2em} N(\Sigma \varphi) \leftarrow 0 \quad \triangleright \text{visit count } N \text{ initialized}
17: \hspace{2em} H(\Sigma \varphi) \leftarrow \emptyset \quad \triangleright \text{reward function } R \text{ initialized}
18: \hspace{1em} MCTSSearch(\( \Sigma \varphi \)) \quad \triangleright \text{recursive call}
19: \hspace{1em} R(\Sigma) \leftarrow \max_{\varphi \mathcal{T} \in \text{ISForm}(\( \Sigma \))} R(\Sigma \varphi) \quad \triangleright \text{back propagation for reward}
20: \hspace{1em} else \quad \triangleright \text{a leaf node reached}
21: \hspace{2em} while within playout budget \( B_P \) do \quad \triangleright \text{playout by hill-climbing falsification}
22: \hspace{3em} u \leftarrow \text{Hill-Climb}(H(\Sigma)) \quad \triangleright \text{hill-climbing}
23: \hspace{3em} rb \leftarrow \text{QBRob}(M(u, \varphi, \Sigma_{\text{head}})) \quad \triangleright \text{falsifying input found}
24: \hspace{3em} if rb < 0 then \quad \triangleright \text{record sampling history}
25: \hspace{4em} return (u, rb) \quad \triangleright \text{falsifying input found}
26: \hspace{3em} H(\Sigma) \leftarrow H(\Sigma) \cup \{ (u, rb) \} \quad \triangleright \text{record sampling history}
27: \hspace{3em} R(w) \leftarrow \text{Rwd}(rb, H(\Sigma)) \quad \triangleright \text{back propagation for visit count}
28: \hspace{2em} N(\Sigma) \leftarrow N(\Sigma) + 1 \quad \triangleright \text{back propagation for visit count}

MCTSSearch (Line 18).

4.2.0.3 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 \( \Sigma \) represented by a leaf node is actually the concatenation between \( \varphi \) and a sub-formula sequence of \( \varphi \). We extract the suffix of \( \Sigma \), 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 \( u \) 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 \( \Sigma \)
Initialization

Expansion

Playout

Backpropagation

Selection

Termination

with which hill-climbing optimization can minimize $\text{QBRob}(w, \varphi, \Sigma)$ smoothly, we connect the reward with the hill-climbing progress. Formally, given a sampling history $H$, our reward (Line 27) is defined as $\text{Rwd}(rb', H) := \max_{rb_h} \min_{\Sigma} \left( \frac{rb_h - \min_{rb_h} (rb')}{} \right)$, where $rb_h$ is the history of robustness values in $H$.

4.2.0.4 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 12 (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 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 §5.1, and then we present the experimental evaluation results by answering three research questions in §5.2.
5.1 Experiment Setup

5.1.0.1 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, \( \text{throttle} \in [0, 100] \) and \( \text{brake} \in [0, 325] \), and three outputs signals including \( \text{gear} \), \( \text{speed} \) and \( \text{rpm} \). Most of the specifications we used formalize safety requirements of the system. For instance, AT2 requires that when \( \text{gear} \) is as high as 4, \( \text{rpm} \) should not be larger than 4300; AT3 is an adaptation of the example we used in §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 \( \text{gear} \); 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, \( \text{PedalAngle} \in [8.8, 70] \) and \( \text{EngineSpeed} \in [900, 1100] \), and outputs a ratio \( \mu \) reflecting the deviation of \( \text{air-fuel-ratio} \) from its reference value. The basic safety requirement to this system is that \( \mu \) 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 seconds.

5.1.0.2 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 FORMula Exploitation by Sequence trEE for falsification. Breach is a state-of-the-art falsification tool that implements the classic falsification workflow we introduced in §2. The quantitative robustness calculation in Breach is based on the robust semantics given in Def. 3. 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.

We apply the two approaches, ForeSee and
Therefore, FR is an indicator of the in which the approach returns a falsifying input.

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 seconds (see Alg. 2).

### 5.1.0.3 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.

### 5.1.0.4 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 Alg. 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 subformulas in AT14 must be falsified to falsify the whole specification, and they are all difficult to be falsified. In such a case, there is no best subformula sequence $\Sigma$: therefore, the time spent by ForeSee in exploring different $\Sigma$ 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, 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. 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 perfor-

---

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

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

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 [42]. 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 ($[0,30]$ ($\text{speed} < 135 \land \text{rpm} < 4780$)). 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 Alg. 2), and the playout budget $B_P$ (Line 21 in Alg. 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
表 4: Falsification performance under different MCTS hyperparameters.

(a) Performance with varied $c$

<table>
<thead>
<tr>
<th></th>
<th>AT17</th>
<th>AT19</th>
<th>AT21</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c$</td>
<td>FR time</td>
<td>FR time</td>
<td>FR time</td>
</tr>
<tr>
<td>0</td>
<td>23 177.8</td>
<td>30 224.6</td>
<td>30 463.4</td>
</tr>
<tr>
<td>0.02</td>
<td>26 196.7</td>
<td>30 278.5</td>
<td>28 501.3</td>
</tr>
<tr>
<td>0.2</td>
<td>30 208.8</td>
<td>30 154.3</td>
<td>29 425.5</td>
</tr>
<tr>
<td>0.5</td>
<td>30 297.0</td>
<td>29 227.3</td>
<td>30 509.0</td>
</tr>
<tr>
<td>1.0</td>
<td>30 311.7</td>
<td>30 240.2</td>
<td>24 497.0</td>
</tr>
</tbody>
</table>

(b) Performance with varied $B_P$

<table>
<thead>
<tr>
<th></th>
<th>AT17</th>
<th>AT19</th>
<th>AT21</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_P$</td>
<td>FR time</td>
<td>FR time</td>
<td>FR time</td>
</tr>
<tr>
<td>2</td>
<td>26 385.2</td>
<td>30 162.0</td>
<td>29 500.0</td>
</tr>
<tr>
<td>5</td>
<td>30 347.7</td>
<td>29 207.3</td>
<td>29 472.5</td>
</tr>
<tr>
<td>10</td>
<td>30 208.8</td>
<td>30 154.3</td>
<td>29 425.5</td>
</tr>
<tr>
<td>15</td>
<td>30 337.7</td>
<td>29 336.7</td>
<td>28 514.0</td>
</tr>
<tr>
<td>20</td>
<td>30 358.1</td>
<td>30 313.5</td>
<td>30 511.0</td>
</tr>
</tbody>
</table>

Breach in Table 2: for such a specification, we need to perform some exploration to find the best $\Sigma$: this explains 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 a 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 non-standard analysis [9,16,20,22,31,33], 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 con-
connectives: $\square (\varphi_1 \land \varphi_2)$ and $\square (\varphi_1 \lor \varphi_2)$. 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 §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 §2.1) that could provide a better initial guidance.

Acknowledgment

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.

参考文献


