Multi-Armed Bandits for Boolean Connectives in Hybrid System Falsification

Zhenya Zhang, Ichiro Hasuo, Paolo Arcaini

Hybrid system falsification is an actively studied topic, as a scalable quality assurance methodology for real-world cyber-physical systems. In falsification, one employs stochastic hill-climbing optimization to quickly find a counterexample input to a black-box system model. Quantitative *robust semantics* is the technical key that enables use of such optimization. In this paper, we tackle the so-called *scale problem* regarding Boolean connectives that is widely recognized in the community: quantities of different scales (such as speed [km/h] vs. rpm, or worse, rph) can mask each other's contribution to robustness. Our solution consists of integration of the *multi-armed bandit* algorithms in hill climbing-guided falsification frameworks, with a technical novelty of a new reward notion that we call *hill-climbing gain*. Our experiments show our approach's robustness under the change of scales, and that it outperforms a state-of-the-art falsification tool.

1 Introduction

Hybrid System Falsification

Quality assurance of *cyber-physical systems (CPS)* is attracting growing attention from both academia and industry, not only because it is challenging and scientifically interesting, but also due to the safety-critical nature of many CPS. The combination of physical systems (with continuous dynamics) and digital controllers (that are inherently discrete) is referred to as *hybrid systems*, capturing an important aspect of CPS. To verify hybrid systems is intrinsically hard, because the continuous dynamics therein leads to infinite search spaces.

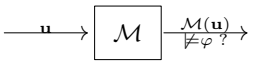
More researchers and practitioners are therefore turning to *optimization-based falsification* as a quality assurance measure for CPS. The problem is formalized as follows.

This work was originally published at 31st International Conference on Computer-Aided Verification.

Zhenya Zhang, Ichiro Hasuo, Paolo Arcaini, 国立情報学研究所, National Institute of Informatics.

The falsification problem

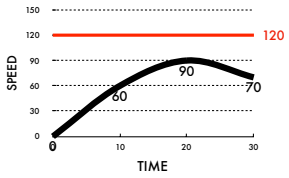
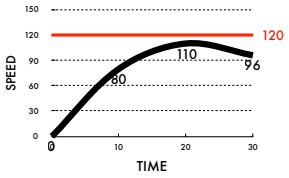
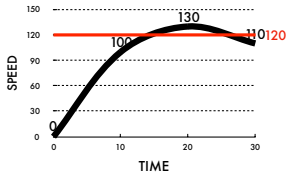
- **Given:** a *model* $\mathcal{M}$ (that takes an input signal $\mathbf{u}$ and yields an output signal $\mathcal{M}(\mathbf{u})$), and a *specification* φ (a temporal formula)
- **Find:** a *falsifying input*, that is, an input signal $\mathbf{u}$ such that the corresponding output $\mathcal{M}(\mathbf{u})$ violates φ



In optimization-based falsification, the above problem is turned into an optimization problem. It is *robust semantics* of temporal formulas [12, 17] that makes it possible. Instead of the Boolean satisfaction relation $\mathbf{v} \models \varphi$, robust semantics assigns a quantity $[\![\mathbf{v}, \varphi]\!] \in \mathbb{R} \cup \{\infty, -\infty\}$ that tells us, not only whether φ is true or not (by the sign), but also *how robustly* the formula is true or false. This allows one to employ hill-climbing optimization: we iteratively generate input signals, in the direction of decreasing robustness, hoping that eventually we hit negative robustness.

An illustration of robust semantics is in Table 1. We use *signal temporal logic (STL)* [12], a temporal logic that is commonly used in hybrid system specification.

表 1: Boolean satisfaction $\mathbf{w} \models \varphi$, and quantitative robustness values $[\![\mathbf{w}, \varphi]\!]$, of three signals of *speed* for the STL formula $\varphi \equiv \square_{[0,30]}(speed < 120)$

signal $\mathbf{w}$			
$\mathbf{w} \models \varphi$	True	True	False
$[\![\mathbf{w}, \varphi]\!]$	30	10	-10

The specification says the speed must always be below 120 during the time interval $[0, 30]$. In the search of an input signal $\mathbf{u}$ (e.g. of throttle and brake) whose corresponding output $\mathcal{M}(\mathbf{u})$ violates the specification, the quantitative robustness $[\![\mathcal{M}(\mathbf{u}), \varphi]\!]$ gives much more information than the Boolean satisfaction $\mathcal{M}(\mathbf{u}) \models \varphi$. Indeed, in Table 1, while Boolean satisfaction fails to discriminate the first two signals, the quantitative robustness indicates a tendency that the second signal is closer to violation of the specification.

In the falsification literature, stochastic algorithms are used for hill-climbing optimization. Examples include simulated annealing (SA), globalized Nelder-Mead (GNM [30]) and covariance matrix adaptation evolution strategy (CMA-ES [6]). Note that the system model $\mathcal{M}$ can be black-box: we have only to observe the correspondence between input $\mathbf{u}$ and output $\mathcal{M}(\mathbf{u})$. Observing an error $\mathcal{M}(\mathbf{u}')$ for some input $\mathbf{u}'$ is sufficient evidence for a system designer to know that the system needs improvement. Besides these practical advantages, optimization-based falsification is an interesting scientific topic: it combines two different worlds of formal reasoning and stochastic optimization.

Optimization-based falsification started in [17] and has been developed vigorously [1,3–5,9,11–13,15,27,28,34, 36,38]. See [26] for a survey. There are mature tools such as Breach [11] and S-Taliro [5]; they work with industry-standard Simulink models.

Challenge: The Scale Problem in Boolean Superposition

In the field of hybrid falsification—and more generally in search-based testing—the following problem is widely recognized. We shall call the problem *the scale problem (in Boolean superposition)*.

Consider an STL specification $\varphi \equiv \square_{[0,30]}(\neg(rpm > 4000) \vee (speed > 20))$ for a car; it is equivalent to $\square_{[0,30]}((rpm > 4000) \to (speed > 20))$ and says that the speed should not be too small whenever the rpm is over 4000. According to the usual definition in the literature [11, 17], the Boolean connectives $\neg$ and $\vee$ are interpreted by $-$ and the supremum $\sqcup$, respectively; and the "always" operator $\square_{[0,30]}$ is by infimum $\bigsqcap$. Therefore the robust semantics of φ under the signal $(rpm, speed)$, where $rpm, speed \colon [0, 30] \to \mathbb{R}$, is given as follows.

$$[\![(rpm, speed), \varphi]\!] = \bigsqcap_{t \in [0,30]} \big((4000 - rpm(t)) \sqcup (speed(t) - 20) \big) \tag{1}$$

A problem is that, in the supremum of two real values in (1), one component can totally *mask* the contribution of the other. In this specific example, the former (rpm) component can have values as big as thousands, while the latter $(speed)$ component will be in the order of tens. This means that in hill-climbing optimization it is hard to use the information of both signals, as one will be masked.

Another related problem is that the efficiency of a falsification algorithm would depend on the choice of units of measure. Imagine replacing rpm with rph in (1), which makes the constant 4000 into 240000, and make the situation even worse.

These problems—that we call the *scale problem*—occur in many falsification examples, specifically when a specification involves Boolean connectives. We do need Boolean connectives in specifications: for example, many real-world specifications in industry are of the form $\Box_I(\varphi_1 \rightarrow \varphi_2)$, requiring that an event φ_1 triggers a countermeasure φ_2 all the time.

One could use different operators for interpreting Boolean connectives. For example, in [21], $\vee$ and $\wedge$ are interpreted by $+$ and $\times$ over $\mathbb{R}$, respectively. However, these choices do not resolve the scale problem, either. In general, it does not seem easy to come up with a fixed set of operators over $\mathbb{R}$ that interpret Boolean connectives and are free from the scale problem.

図 1: A multi-armed bandit for falsifying $\Box_I(\varphi_1 \wedge \varphi_1)$

Contribution: Integrating Multi-Armed Bandits into Optimization-Based Falsification

As a solution to the scale problem in Boolean superposition that we just described, we introduce a new approach that does *not* superpose robustness values. Instead, we integrate *multi-armed bandits (MAB)* in the existing framework of falsification guided by hill-climbing optimization.

The MAB problem is a prototypical reinforcement learning problem: a gambler sits in front of a row of slot machines; their performance (i.e. average reward) is not known; the gambler plays a machine in each round and he continues with many rounds; and the goal is to optimize cumulative rewards. The gambler needs to play different machines and figure out their performance, at the cost of the loss of opportunities in the form of playing suboptimal machines.

In this paper, we focus on specifications of the form $\Box_I(\varphi_1 \wedge \varphi_2)$ and $\Box_I(\varphi_1 \vee \varphi_2)$; we call them *(conjunctive/disjunctive) safety properties*. We identify an instance of the MAB problem in the choice of the formula (out of φ_1, φ_2) to try to falsify by hill climbing. See Fig. 1. We combine MAB algorithms (such as ε-greedy and UCB1, see §3. 2) with hill-climbing optimization, for the purpose of coping with the scale problem in Boolean superposition. This combination is made possible by introducing a novel reward notion for MAB, called *hill-climbing gain*, that is tailored for this purpose.

We have implemented our MAB-based falsification framework in MATLAB, building on Breach [11].†1 Our experiments with benchmarks from [7, 24, 25] demonstrate that our MAB-based approach is a viable one against the scale problem. In particular, our approach is observed to be (almost totally) robust under the change of scaling (i.e. changing units of measure, such as from rpm to rph that we discussed after the formula (1)). Moreover, for the benchmarks taken from the previous works—they do not suffer much from the scale problem—our algorithm performs better than the state-of-the-art falsification tool `Breach` [11].

Related Work

Besides those we mentioned, we shall discuss some related works.

Formal verification approaches to correctness of hybrid systems employ a wide range of techniques, including model checking, theorem proving, rigorous numerics, nonstandard analysis, and so on [8, 14, 18, 20, 22, 23, 29, 32]. These are currently not very successful in dealing with complex real-world systems, due to issues like scalability and black-box components.

Our use of MAB in falsification exemplifies the role of the *exploration-exploitation trade-off*, the core problem in reinforcement learning. The trade-off has been already discussed for the verification of quantitative properties (e.g., [33]) and also in some works on falsification. A recent example is [36], where they use Monte Carlo

†1 Code obtained at https://github.com/decyphir/breach.

tree search to force systematic exploration of the space of input signals. Besides MCTS, *Gaussian process learning (GP learning)* has also attracted attention in machine learning as a clean way of balancing exploitation and exploration. The GP-UCB algorithm is a widely used strategy there. Its use in hybrid system falsification is pursued e.g. in [3, 34].

More generally, *coverage-guided falsification* [1, 9, 13, 28] aims at coping with the exploration-exploitation trade-off. One can set the current work in this context—the difference is that we force systematic exploration on the specification side, not in the input space.

There have been efforts to enhance expressiveness of MTL and STL, so that engineers can express richer intentions—such as time robustness and frequency—in specifications [2, 31]. This research direction is orthogonal to ours; we plan to investigate the use of such logics in our current framework.

A similar masking problem around Boolean connectives is discussed in [10, 19]. Compared to those approaches, our technique does not need the explicit declaration of *input vacuity* and *output robustness*, but it relies on the "hill-climbing gain" reward to learn the significance of each signal.

Finally, the interest in the use of deep neural networks is rising in the field of falsification (as well as in many other fields). See e.g. [4, 27].

2 Preliminaries: Hill Climbing-Guided Falsification

We review a well-adopted methodology for hybrid system falsification, namely the one guided by hill-climbing optimization. It makes essential use of quantitative *robust semantics* of temporal formulas, which we review too.

2.1 Robust Semantics for STL

Our definitions here are taken from [12, 17].

Definition 1 ((time-bounded) signal) *Let $T \in \mathbb{R}_+$ be a positive real. An M-*dimensional signal *with a time horizon T is a function $\mathbf{w} \colon [0, T] \to \mathbb{R}^M$.*

Let $\mathbf{w} \colon [0, T] \to \mathbb{R}^M$ and $\mathbf{w}' \colon [0, T'] \to \mathbb{R}^M$ be M-dimensional signals. Their concatenation *$\mathbf{w} \cdot \mathbf{w}' \colon [0, T + T'] \to \mathbb{R}^M$ is the M-dimensional signal defined by $(\mathbf{w} \cdot \mathbf{w}')(t) = \mathbf{w}(t)$ if $t \in [0, T]$, and $(\mathbf{w} \cdot \mathbf{w}')(t) = \mathbf{w}'(t - T)$ if $t \in (T, T + T']$.*

Let $0 < T_1 < T_2 \leq T$. The restriction *$\mathbf{w}|_{[T_1, T_2]} \colon [0, T_2 - T_1] \to \mathbb{R}^M$ of $\mathbf{w} \colon [0, T] \to \mathbb{R}^M$ to the interval $[T_1, T_2]$ is defined by $(\mathbf{w}|_{[T_1, T_2]})(t) = \mathbf{w}(T_1 + t)$.*

One main advantage of optimization-based falsification is that a system model can be a black box—observing the correspondence between input and output suffices. We therefore define a system model simply as a function.

Definition 2 (system model $\mathcal{M}$) *A* system model*, with M-dimensional input and N-dim. 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. Furthermore, we impose the following* causality *condition on $\mathcal{M}$: for any time-bounded signals $\mathbf{u} \colon [0, T] \to \mathbb{R}^M$ and $\mathbf{u}' \colon [0, T'] \to \mathbb{R}^M$, we require that $\mathcal{M}(\mathbf{u} \cdot \mathbf{u}')\big|_{[0,T]} = \mathcal{M}(\mathbf{u})$.*

Definition 3 (STL syntax) *We fix a set $\mathbf{Var}$ of variables. In STL,* atomic propositions *and* formulas *are defined as follows, respectively: $\alpha ::\equiv f(x_1, \dots, x_N) > 0$, and $\varphi ::\equiv \alpha \mid \bot \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \,\mathcal{U}_I\, \varphi$. Here f is an N-ary function $f : \mathbb{R}^N \to \mathbb{R}$, $x_1, \dots, x_N \in \mathbf{Var}$, 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 $a < b$.*

We omit subscripts I for temporal operators if $I = [0, \infty)$. Other common connectives such as $\rightarrow$, $\top$, $\Box_I$ (always) and $\Diamond_I$ (eventually), are introduced as abbreviations: $\Diamond_I \varphi \equiv \top \,\mathcal{U}_I\, \varphi$ and $\Box_I \varphi \equiv \neg\Diamond_I \neg\varphi$. An atomic formula $f(\vec{x}) \leq c$, where $c \in \mathbb{R}$, is accommodated using $\neg$ and the function $f'(\vec{x}) := f(\vec{x}) - c$.

Definition 4 (robust semantics [12]) *Let $\mathbf{w} \colon [0, T] \to \mathbb{R}^N$ be an N-dimensional signal, and $t \in [0, T)$. The t-*shift *of $\mathbf{w}$, denoted by $\mathbf{w}^t$, is the time-bounded signal $\mathbf{w}^t \colon [0, T - t] \to \mathbb{R}^N$ defined by $\mathbf{w}^t(t') := \mathbf{w}(t + t')$.*

Let $\mathbf{w} \colon [0, T] \to \mathbb{R}^{|\mathbf{Var}|}$ be a signal, and φ be an

STL formula. We define the robustness $[\![\mathbf{w}, \varphi]\!] \in \mathbb{R} \cup \{\infty, -\infty\}$ *as follows, by induction on the construction of formulas. Here* $\bigsqcap$ *and* $\bigsqcup$ *denote infimums and supremums of real numbers, respectively. Their binary version* $\sqcap$ *and* $\sqcup$ *denote minimum and maximum.*

$$
\begin{aligned}
&[\![\mathbf{w}, f(x_1, \cdots, x_n) > 0]\!] := f\big(\mathbf{w}(0)(x_1), \cdots, \mathbf{w}(0)(x_n)\big) \\
&[\![\mathbf{w}, \bot]\!] := -\infty \qquad [\![\mathbf{w}, \neg\varphi]\!] := -[\![\mathbf{w}, \varphi]\!] \\
&[\![\mathbf{w}, \varphi_1 \wedge \varphi_2]\!] := [\![\mathbf{w}, \varphi_1]\!] \sqcap [\![\mathbf{w}, \varphi_2]\!] \qquad (2) \\
&[\![\mathbf{w}, \varphi_1 \vee \varphi_2]\!] := [\![\mathbf{w}, \varphi_1]\!] \sqcup [\![\mathbf{w}, \varphi_2]\!] \qquad (3) \\
&[\![\mathbf{w}, \varphi_1 \,\mathcal{U}_I\, \varphi_2]\!] := \bigsqcup_{t \in I \cap [0,T]} \big([\![\mathbf{w}^t, \varphi_2]\!] \sqcap \bigsqcap_{t' \in [0,t)} [\![\mathbf{w}^{t'}, \varphi_1]\!] \big)
\end{aligned}
$$

For atomic formulas, $[\![\mathbf{w}, f(\vec{x}) > c]\!]$ stands for the vertical margin $f(\vec{x}) - c$ for the signal $\mathbf{w}$ at time 0. A negative robustness value indicates how far the formula is from being true. It follows from the definition that the robustness for the eventually modality is given by $[\![\mathbf{w}, \Diamond_{[a,b]}(x > 0)]\!] = \bigsqcup_{t \in [a,b] \cap [0,T]} \mathbf{w}(t)(x)$.

The above robustness notion taken from [12] is therefore *spatial*. Other robustness notions take *temporal* aspects into account, too, such as "how long before the deadline the required event occurs." See e.g. [2, 12]. Our choice of spatial robustness in this paper is for the sake of simplicity, and is thus not essential.

The original semantics of STL is Boolean, given as usual 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 [17, Prop. 16]. Optimization-based falsification via robust semantics hinges on this refinement.

2.2 Hill Climbing-Guided Falsification

As we discussed in the introduction, the falsification problem attracts growing industrial and academic attention. Its solution methodology by hill-climbing optimization is an established field, too: see [1, 3, 5, 9, 11–13, 15, 26, 28, 34, 38] and the tools Breach [11] and S-TaLiRo [5]. We formulate the problem and the methodology, for later use in describing our multi-armed bandit-based algorithm.

Definition 5 (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 6 (hill climbing-guided falsification) *Assume the setting in Def. 5. 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* $\mathbf{u}_k$*, aiming at minimizing the robustness* $[\![\mathcal{M}(\mathbf{u}_k), \varphi]\!]$*. It does so, learning from the previous observations* $\big(\mathbf{u}_l,\ [\![\mathcal{M}(\mathbf{u}_l), \varphi]\!]\big)_{l \in [1, k-1]}$ *of input signals* $\mathbf{u}_1, \ldots, \mathbf{u}_{k-1}$ *and their corresponding robustness values (cf. Table 1).*

The HILL-CLIMB function can be implemented by various stochastic optimization algorithms. Examples are CMA-ES [6] (used in our experiments), SA, and GNM [30].

Algorithm 1 Hill climbing-guided falsification

Require: a system model $\mathcal{M}$, an STL formula φ, and a budget K

function HILL-CLIMB-FALSIFY($\mathcal{M}, \varphi, K$)
  $\mathsf{rb} \leftarrow \infty$; $k \leftarrow 0$
  while $\mathsf{rb} \geq 0$ and $k \leq K$ **do**
    $k \leftarrow k + 1$
    $\mathbf{u}_k \leftarrow$ HILL-CLIMB$\Big(\big(\mathbf{u}_l,\ [\![\mathcal{M}(\mathbf{u}_l), \varphi]\!]\big)_{l \in [1,k-1]}\Big)$
    $\mathsf{rb}_k \leftarrow [\![\mathcal{M}(\mathbf{u}_k), \varphi]\!]$
    if $\mathsf{rb}_k < \mathsf{rb}$ **then**
      $\mathsf{rb} \leftarrow \mathsf{rb}_k$
  $\mathbf{u} \leftarrow \begin{cases} \mathbf{u}_k & \text{if } \mathsf{rb} < 0, \text{ i.e., } \mathsf{rb}_k = [\![\mathcal{M}(\mathbf{u}_k), \varphi]\!] < 0 \\ \text{Failure} & \text{otherwise} \end{cases}$
  return $\mathbf{u}$

3 Our Multi-Armed Bandit-Based Falsification Algorithm

In this section, we present our contribution, namely a falsification algorithm that addresses the scale problem in Boolean superposition (see §1). The main novelties in the algorithm are as follows.

1. **(Use of MAB algorithms)** For binary Boolean con-

nectives, unlike most works in the field, we do not superpose the robustness values of the constituent formulas φ_1 and φ_2 using a fixed operator (such as $\sqcap$ and $\sqcup$ in (2)). Instead, we view the situation as an instance of the multi-armed bandit problem (MAB): we use an algorithm for MAB to choose one formula φ_i to focus on (here $i \in \{1, 2\}$); and then we apply hill climbing-guided falsification to the chosen formula φ_i.

2. **(*Hill-climbing gain* as rewards in MAB)** For our integration of MAB and hill-climbing optimization, the technical challenge is find a suitable notion of reward for MAB. We introduce a novel notion that we call *hill-climbing gain*: it formulates the (downward) robustness gain that we would obtain by applying hill-climbing optimization, suitably normalized using the scale of previous robustness values.

Later, in §4, we demonstrate that combining those two features gives rise to falsification algorithms that successfully cope with the scale problem in Boolean superposition.

Our algorithms focus on a fragment of STL as target specifications. They are called *(disjunctive and conjunctive) safety properties*. In §3. 1 we describe this fragment of STL, and introduce necessary adaptation of the semantics. After reviewing the MAB problem in §3. 2, we present our algorithms in §3. 3–3. 4.

3. 1 Conjunctive and Disjunctive Safety Properties

Definition 7 (conjunctive/disjunctive safety property) *An STL formula of the form $\Box_I(\varphi_1 \wedge \varphi_2)$ is called a* conjunctive safety property*; an STL formula of the form $\Box_I(\varphi_1 \vee \varphi_2)$ is called a* disjunctive safety property.

It is known that, in industry practice, a majority of specifications is of the form $\Box_I(\varphi_1 \rightarrow \varphi_2)$, where φ_1 describes a trigger and φ_2 describes a countermeasure that should follow. This property is equivalent to $\Box_I(\neg\varphi_1 \vee \varphi_2)$, and is therefore a disjunctive safety property.

In §3. 3–3. 4, we present two falsification algorithms, for conjunctive and disjunctive safety properties respectively. For the reason we just discussed, we expect the disjunctive algorithm should be more important in real-world application scenarios. In fact, the disjunctive algorithm turns out to be more complicated, and it is best introduced as an extension of the conjunctive algorithm.

We define the restriction of robust semantics to a (sub)set of time instants. Note that we do not require $\mathcal{S} \subseteq [0, T]$ to be a single interval.

Definition 8 ($[\![\mathbf{w}, \psi]\!]_{\mathcal{S}}$, robustness restricted to $\mathcal{S} \subseteq [0, T]$) *Let $\mathbf{w} \colon [0, T] \to \mathbb{R}^{|\mathbf{Var}|}$ be a signal, ψ be an STL formula, and $\mathcal{S} \subseteq [0, T]$ be a subset. We define the* robustness *of $\mathbf{w}$ under ψ* restricted to $\mathcal{S}$ *by*

$$[\![\mathbf{w}, \psi]\!]_{\mathcal{S}} \;:=\; \textstyle\bigsqcap_{t \in \mathcal{S}} [\![\mathbf{w}^t, \psi]\!] \;. \tag{4}$$

Obviously, $[\![\mathbf{w}, \psi]\!]_{\mathcal{S}} < 0$ implies that there exists $t \in \mathcal{S}$ such that $[\![\mathbf{w}^t, \psi]\!]_{\mathcal{S}} < 0$. We derive the following easy lemma; it is used later in our algorithm.

Lemma 9 *In the setting of Def. 8, consider a disjunctive safety property $\varphi \equiv \Box_I(\varphi_1 \vee \varphi_2)$, and let $\mathcal{S} := \{t \in I \cap [0, T] \mid [\![\mathbf{w}^t, \varphi_1]\!] < 0\}$. Then $[\![\mathbf{w}, \varphi_2]\!]_{\mathcal{S}} < 0$ implies $[\![\mathbf{w}, \Box_I(\varphi_1 \vee \varphi_2)]\!] < 0$.*

3. 2 The Multi-Armed Bandit (MAB) Problem

The *multi-armed bandit* (MAB) problem describes a situation where,

- a gambler sits in front of a row $A_1, \ldots, A_n$ of slot machines;
- each slot machine A_i gives, when its arm is played (i.e. in each attempt), a reward according to a prescribed (but unknown) probability distribution μ_i;
- and the goal is to maximize the cumulative reward after a number of attempts, playing a suitable arm in each attempt.

The best strategy of course is to keep playing the best arm $A_{\max}$, i.e. the one whose average reward $\mathsf{avg}(\mu_{\max})$ is the greatest. This best strategy is infeasible, however, since the distributions $\mu_1, \ldots, \mu_n$ are initially unknown. Therefore the gambler must learn about $\mu_1, \ldots, \mu_n$ through attempts.

The MAB problem exemplifies the "learning by trying" paradigm of *reinforcement learning*, and is thus heavily studied. The greatest challenge is to balance between *exploration* and *exploitation*. A greedy (i.e. exploitation-only) strategy will play the arm whose empirical average reward is the maximum. However, since the rewards are random, this way the gambler can miss another arm whose real performance is even better but which is yet to be found so. Therefore one needs to mix exploration, too, occasionally trying empirically non-optimal arms, in order to identity their true performance.

The relevance of MAB to our current problem is as follows. Falsifying a conjunctive safety property $\Box_I(\varphi_1 \wedge \varphi_2)$ amounts to finding a time instant $t \in I$ at which either φ_1 or φ_2 is falsified. We can see the two subformulas (φ_1 and φ_2) as two arms, and this constitutes an instance of the MAB problem. In particular, playing an arm translates to a falsification attempt by hill climbing, and collecting rewards translates to spending time to minimize the robustness. We show in §3. 3–3. 4 that this basic idea extends to disjunctive safety properties $\Box_I(\varphi_1 \vee \varphi_2)$, too.

A rigorous formulation of the MAB problem is presented for the record.

Definition 10 (the multi-armed bandit problem) *The* multi-armed bandit (MAB) *problem is formulated as follows.*

Input: *arms $(A_1, \dots, A_n)$, the associated probability distributions $\mu_1, \dots, \mu_n$ over $\mathbb{R}$, and a time horizon $H \in \mathbb{N} \cup \{\infty\}$.*

Goal: *synthesize a sequence $A_{i_1} A_{i_2} \dots A_{i_H}$, so that the cumulative reward $\sum_{k=1}^{H} \mathsf{rew}_k$ is maximized. Here the reward rew_k of the k-th attempt is sampled from the distribution μ_{i_k} associated with the arm A_{i_k} played at the k-th attempt.*

We introduce some notations for later use. Let $(A_{i_1} \dots A_{i_k}, \mathsf{rew}_1 \dots \mathsf{rew}_k)$ be a history*, i.e. the sequence of arms played so far (here $i_1, \dots, i_k \in [1, n]$), and the sequence of rewards obtained by those attempts (rew_l is sampled from μ_{i_l}).*

For an arm A_j, its visit count $N(j, A_{i_1} A_{i_2} \dots A_{i_k}, \mathsf{rew}_1 \mathsf{rew}_2 \dots \mathsf{rew}_k)$ *is given by the number of occurrences of A_j in $A_{i_1} A_{i_2} \dots A_{i_k}$. Its* empirical average reward $R(j, A_{i_1} A_{i_2} \dots A_{i_k}, \mathsf{rew}_1 \mathsf{rew}_2 \dots \mathsf{rew}_k)$ *is given by $\sum_{l \in \{l \in [1,k] \mid i_l = j\}} \mathsf{rew}_l$, i.e. the average return of the arm A_j in the history. When the history is obvious from the context, we simply write $N(j, k)$ and $R(j, k)$.*

3. 2. 1 MAB Algorithms

There have been a number of algorithms proposed for the MAB problem; each of them gives a *strategy* (also called a *policy*) that tells which arm to play, based on the previous attempts and their rewards. The focus here is how to resolve the exploration-exploitation trade-off. Here we review two well-known algorithms.

3. 2. 1. 1 The ε-Greedy Algorithm

This is a simple algorithm that spares a small fraction ε of chances for empirically non-optimal arms. The spared probability ε is uniformly distributed. See Algorithm 2.

Algorithm 2 The ε-greedy algorithm for MAB

Require: the setting of Def. 10, and a constant $\varepsilon > 0$ (typically very small)

At the k-th attempt, choose the arm A_{i_k} as follows

1: $j_{\text{emp-opt}} \leftarrow \arg\max_{j \in [1,n]} R(j, k-1)$ ▷ the arm that is empirically optimal

2: Sample $i_k \in [1, n]$ from the distribution

$$\left[\begin{array}{rcll} j_{\text{emp-opt}} & \longmapsto & (1-\varepsilon) + \frac{\varepsilon}{n} & \\ j & \longmapsto & \frac{\varepsilon}{n} & \text{for each } j \in [1,n] \setminus \{j_{\text{emp-opt}}\} \end{array}\right]$$

3: **return** i_k

3. 2. 1. 2 The UCB1 Algorithm

The UCB1 (*upper confidence bound*) algorithm is more complex; it comes with a theoretical upper bound for *regrets*, i.e. the gap between the expected cumulative reward and the optimal (but infeasible) cumulative reward (i.e. the result of keep playing the optimal arm $A_{\max}$). It is known that the UCB1 algorithm's regret is at most $O(\sqrt{nH \log H})$ after H attempts, improving the naive random strategy (which has the expected regret $O(H)$).

See Alg. 3. The algorithm is deterministic, and picks the arm that maximizes the value shown in Line 1. The first term $R(j, k-1)$ is the *exploitation* factor, reflecting

the arm's empirical performance. The second term is the *exploration* factor. Note that it is bigger if the arm A_j has been played less frequently. Note also that the exploration factor eventually decays over time: the denominator grows roughly with $O(k)$, while the numerator grows with $O(\ln k)$.

Algorithm 3 The UCB1 algorithm for MAB

Require: the setting of Def. 10, and a constant $c > 0$

At the k-th attempt, choose the arm A_{i_k} as follows

$i_k \leftarrow \arg\max_{j \in [1,n]} \left(R(j, k-1) + c\sqrt{\frac{2\ln(k-1)}{N(j,k-1)}} \right)$

return i_k

3.3 Our MAB-Guided Algorithm I: Conjunctive Safety Properties

Our first algorithm targets at conjunctive safety properties. It is based on our identification of MAB in a Boolean conjunction in falsification—this is as we discussed just above Def. 10. The technical novelty lies in the way we combine MAB algorithms and hill-climbing optimization; specifically, we introduce the notion of *hill-climbing gain* as a reward notion in MAB (Def. 11). This first algorithm paves the way to the one for disjunctive safety properties, too (§3.4).

Algorithm 4 Our MAB-guided algorithm I: *conjunctive* safety properties

Require: a system model $\mathcal{M}$, an STL formula $\varphi \equiv \Box_I(\varphi_1 \wedge \varphi_2)$, and a budget K

function MAB-FALSIFY-CONJ-SAFETY($\mathcal{M}, \varphi, K$)

$\mathsf{rb} \leftarrow \infty$; $k \leftarrow 0$

while $\mathsf{rb} \geq 0$ and $k \leq K$ **do**

$k \leftarrow k + 1$

$i_k \leftarrow \mathrm{MAB}\Big((\varphi_1, \varphi_2),\ \big(\mathcal{R}(\varphi_1), \mathcal{R}(\varphi_2)\big),\ \varphi_{i_1} \dots \varphi_{i_{k-1}},\ \mathsf{rew}_1 \dots \mathsf{rew}_{k-1} \Big)$

$\mathbf{u}_k \leftarrow \textsc{Hill-Climb}\big(\big((\mathbf{u}_l,\ \mathsf{rb}_l) \big) \big)$ where $l \in [1, k-1]$ such that $i_l = i_k$

$\mathsf{rb}_k \leftarrow [\![\mathcal{M}(\mathbf{u}_k), \Box_I \varphi_{i_k}]\!]$

if $\mathsf{rb}_k < \mathsf{rb}$ **then**

$\mathsf{rb} \leftarrow \mathsf{rb}_k$

$\mathbf{u} \leftarrow \begin{cases} \mathbf{u}_k & \text{if } \mathsf{rb} < 0 \\ \text{Failure} & \text{otherwise} \end{cases}$

return $\mathbf{u}$

Algorithm 5 Our MAB-guided algorithm II: *disjunctive* safety properties

Require: a system model $\mathcal{M}$, an STL formula $\varphi \equiv \Box_I(\varphi_1 \vee \varphi_2)$, and a budget K

function MAB-FALSIFY-DISJ-SAFETY($\mathcal{M}, \varphi, K$)

The same as Algorithm 4, except that Line 7 is replaced by the following Line 7'.

7': $\mathsf{rb}_k \leftarrow [\![\mathcal{M}(\mathbf{u}_k), \varphi_{i_k}]\!]_{\mathcal{S}_k}$ where $\mathcal{S}_k = \big\{ t \in I \cap [0,T] \,\big|\, [\![\mathcal{M}(\mathbf{u}_k^t), \varphi_{\overline{i_k}}]\!] < 0 \big\}$

▷ here $\varphi_{\overline{i_k}}$ denotes the other formula than φ_{i_k}, among φ_1, φ_2

The algorithm is in Algorithm 4. Some remarks are in order.

Algorithm 4 aims to falsify a conjunctive safety property $\varphi \equiv \Box_I(\varphi_1 \wedge \varphi_2)$. Its overall structure is to *interleave* two sequences of falsification attempts, both of which are hill climbing-guided. These two sequences of attempts aim to falsify $\Box_I\varphi_1$ and $\Box_I\varphi_2$, respectively. Note that $[\![\mathcal{M}(\mathbf{u}), \varphi]\!] \leq [\![\mathcal{M}(\mathbf{u}), \Box_I\varphi_1]\!]$, therefore falsification of $\Box_I\varphi_1$ implies falsification of φ; the same holds for $\Box_I\varphi_2$, too.

In Line 5 we run an MAB algorithm to decide which of $\Box_I\varphi_1$ and $\Box_I\varphi_2$ to target at in the k-th attempt. The function MAB takes the following as its arguments: 1) the list of arms, given by the formulas φ_1, φ_2; 2) their rewards $\mathcal{R}(\varphi_1), \mathcal{R}(\varphi_2)$; 3) the history $\varphi_{i_1} \dots \varphi_{i_{k-1}}$ of previously played arms ($i_l \in \{1, 2\}$); and 4) the history $\mathsf{rew}_1 \dots \mathsf{rew}_{k-1}$ of previously observed rewards. This way, the type of the MAB function in Line 5 matches the format in Def. 10, and thus the function can be instantiated with any MAB algorithm such as Algorithms 2–3.

The only missing piece is the definition of the rewards $\mathcal{R}(\varphi_1), \mathcal{R}(\varphi_2)$. We introduce the following notion, tailored for combining MAB and hill climbing.

Definition 11 (hill-climbing gain) *In Algorithm 4, in Line 5, the reward $\mathcal{R}(\varphi_i)$ of the arm φ_i (where $i \in \{1, 2\}$) is defined by*

$$\mathcal{R}(\varphi_i) = \begin{cases} \frac{\mathsf{max\text{-}rb}(i,k-1) - \mathsf{last\text{-}rb}(i,k-1)}{\mathsf{max\text{-}rb}(i,k-1)} & \text{if } \varphi_i \text{ was played} \\ 0 & \text{otherwise} \end{cases}$$

Here $\mathsf{max\text{-}rb}(i, k-1) := \max\{\mathsf{rb}_l \mid l \in [1, k-$

$1], i_l = i\}$ *(i.e. the greatest* rb_l *so far, in those attempts where* φ_i *was played), and* $\mathsf{last\text{-}rb}(i, k-1) := \mathsf{rb}_{l_{\mathrm{last}}}$ *with* l_{last} *being the greatest* $l \in [1, k-1]$ *such that* $i_l = i$ *(i.e. the last* rb_l *for* φ_i*).*

Since we try to minimize the robustness values rb_l through falsification attempts, we can expect that rb_l for a fixed arm φ_i decreases over time. (In the case of the hill-climbing algorithm CMA-ES that we use, this is in fact guaranteed). Therefore the value $\mathsf{max\text{-}rb}(i, k-1)$ in the definition of $\mathcal{R}(\varphi_i)$ is the first observed robustness value. The numerator $\mathsf{max\text{-}rb}(i, k-1) - \mathsf{last\text{-}rb}(i, k-1)$ then represents how much robustness we have reduced so far by hill climbing—hence the name "hill-climbing gain." The denominator $\mathsf{max\text{-}rb}(i, k-1)$ is there for normalization.

In Algorithm 4, the value rb_k is given by the robustness $[\![\mathcal{M}(\mathbf{u}_k), \Box_I \varphi_{i_k}]\!]$. Therefore the MAB choice in Line 5 essentially picks i_k for which hill climbing yields greater effect (but also taking exploration into account—see §3. 2).

In Line 6 we conduct hill-climbing optimization—see §2. 2. The function HILL-CLIMB learns from the previous attempts $\mathbf{u}_{l_1}, \dots, \mathbf{u}_{l_m}$ regarding the same formula φ_{i_k}, and their resulting robustness values $\mathsf{rb}_{l_1}, \dots, \mathsf{rb}_{l_m}$. Then it suggests the next input signal $\mathbf{u}_k$ that is likely to minimize the (unknown) function that underlies the correspondences $\left[\, \mathbf{u}_{l_j} \mapsto \mathsf{rb}_{l_j} \,\right]_{j \in [1,m]}$.

Lines 6–8 read as follows: the hill-climbing algorithm suggests a single input $\mathbf{u}_k$, which is then selected or rejected (Line 8) based on the robustness value it yields (Line 7). We note that this is a simplified picture: in our implementation that uses CMA-ES (it is an evolutionary algorithm), we maintain a population of some ten particles, and each of them is moved multiple times (our choice is three times) before the best one is chosen as $\mathbf{u}_k$.

3. 4 Our MAB-Guided Algorithm II: Disjunctive Safety Properties

The other main algorithm of ours aims to falsify a *disjunctive* safety property $\varphi \equiv \Box_I(\varphi_1 \vee \varphi_2)$. We believe this problem setting is even more important than the conjunctive case, since it encompasses conditional safety properties (i.e. of the form $\Box_I(\varphi_1 \to \varphi_2)$). See §3. 1 for discussions.

In the disjunctive setting, the challenge is that falsification of $\Box_I \varphi_i$ (with $i \in \{1, 2\}$) does *not* necessarily imply falsification of $\Box_I(\varphi_1 \vee \varphi_2)$. This is unlike the conjunctive setting. Therefore we need some adaptation of Algorithm 4, so that the two interleaved sequences of falsification attempts for φ_1 and φ_2 are not totally independent of each other. Our solution consists of *restricting* time instants to those where φ_2 is false, in a falsification attempt for φ_1 (and vice versa), in the way described in Def. 8.

Algorithm 5 shows our MAB-guided algorithm for falsifying a disjunctive safety property $\Box_I(\varphi_1 \vee \varphi_2)$. The only visible difference is that Line 7 in Algorithm 4 is replaced with Line 7'. The new Line 7' measures the quality of the suggested input signal $\mathbf{u}_k$ in the way restricted to the region $\mathcal{S}_k$ in which the other formula is already falsified. Lem. 9 guarantees that, if $\mathsf{rb}_k < 0$, then indeed the input signal $\mathbf{u}_k$ falsifies the original specification $\Box_I(\varphi_1 \vee \varphi_2)$.

The assumption that makes Alg. 5 sensible is that, although it can be hard to find a time instant at which both φ_1 and φ_2 are false (this is required in falsifying $\Box_I(\varphi_1 \vee \varphi_2)$), falsifying φ_1 (or φ_2) individually is not hard. Without this assumption, the region $\mathcal{S}_k$ in Line 7' would be empty most of the time. Our experiments in §4 demonstrate that this assumption is valid in many problem instances, and that Alg. 5 is effective.

4 Experimental Evaluation

We name `MAB-UCB` and `MAB-`ϵ`-greedy` the two versions of MAB algorithm using strategies ε-Greedy (see Alg. 2) and UCB1 (see Alg. 3). We compared the proposed approach (both versions `MAB-UCB` and `MAB-`ϵ`-greedy`) with a state-of-the-art falsification framework, namely `Breach` [11]. `Breach` encapsulates several hill-climbing optimization algorithms, including

CMA-ES (covariance matrix adaptation evolution strategy) [6], *SA (simulated annealing)*, *GNM (global Nelder-Mead)* [30], etc. According to our experience, CMA-ES outperforms other hill-climbing solvers in `Breach`, so the experiments for both `Breach` and our approach rely on the CMA-ES solver.

Experiments have been executed using Breach 1.2.13 on an Amazon EC2 c4.large instance, 2.9 GHz Intel Xeon E5-2666, 2 virtual CPU cores, 4 GB RAM.

Benchmarks

We selected three benchmark models from the literature, each one having different specifications. The first one is the *Automatic Transmission* (AT) model [16, 24]. It has two input signals, $throttle \in [0, 100]$ and $brake \in [0, 325]$, and computes the car's *speed*, engine rotation in rounds per minute rpm, and the automatically selected $gear$. The specifications concern the relation between the three output signals to check whether the car is subject to some unexpected or unsafe behaviors. The second benchmark is the *Abstract Fuel Control* (AFC) model [16, 25]. It takes two input signals, *pedal angle* $\in [8.8, 90]$ and *engine speed* $\in [900, 1100]$, and outputs the critical signal *air-fuel ratio* (AF), which influences fuel efficiency and car performance. The value is expected to be close to a reference value $AFref$; $mu \equiv |AF - AFref| / AFref$ is the deviation of AF from $AFref$. The specifications check whether this property holds under both *normal mode* and *power enrichment mode*. The third benchmark is a model of a *magnetic levitation system with a NARMA-L2 neurocontroller* (NN) [7, 16]. It takes one input signal, $Ref \in [1, 3]$, which is the reference for the output signal Pos, the position of a magnet suspended above an electromagnet. The specifications say that the position should approach the reference signal in a few seconds when these two are not close.

We built the benchmark set `Bbench`, as shown in Table 2a that reports the name of the model and its specifications (ID and formula). In total, we found 11 specifications. In order to increase the benchmark set and obtain specifications of different complexity, we artificially modified a constant (turned into a parameter named τ if it is contained in a time interval, named ρ otherwise) of the specification: for each specification S, we generated m different versions, named as S_i with $i \in \{1, \ldots, m\}$; the complexity of the specification (in terms of difficulty to falsify it) increases with increasing i.[†2] In total, we produced 60 specifications. Column *parameter* in the table shows which concrete values we used for the parameters ρ and τ. Note that all the specifications but one are disjunctive safety properties (i.e., $\Box_I(\varphi_1 \vee \varphi_2)$), as they are the most difficult case and they are the main target of our approach; we just add AT5 as example of conjunctive safety property (i.e., $\Box_I(\varphi_1 \wedge \varphi_2)$).

Our approach has been proposed with the aim of tackling the scale problem. Therefore, to better show how our approach mitigates this problem, we generated a second benchmark set `Sbench` as follows. We selected 15 specifications from `Bbench` (with concrete values for the parameters) and, for each specification S, we changed the corresponding Simulink model by multiplying one of its outputs by a factor 10^k, with $k \in \{-2, 0, 1, 2, 3\}$ (note that we also include the original one using scale factor 10^0); the specification has been modified accordingly, by multiplying with the scale factor the constants that are compared with the scaled output. We name a specification S scaled with factor 10^k as S^k. Table 2b reports the IDs of the original specifications, the output that has been scaled, and the used scaled factors; in total, the benchmark set `Sbench` contains 60 specifications .

Experiment In our context, an *experiment* consists in the execution of an approach A (either `Breach`, `MAB-`ϵ`--greedy`, or `MAB-UCB`) over a specification S for 30 *trials*, using different initial seeds. For each experiment, we record the *success* SR as the number of trials in which a falsifying input was found, and average execution *time* of the trials. Complete experimental results are reported

†2 Note that we performed this classification based on the falsification results of `Breach`.

表 2: Benchmark sets Bbench and Sbench

(a) Bbench (here $\delta_{t'}(\mathbf{w})$ represents $\mathbf{w}^t(t') - \mathbf{w}^t(0)$).

Bench	Specification		Parameter
	ID	Formula	
AT	AT1	$\Box_{[0,30]}((gear = 3) \rightarrow (speed > \rho))$	$\rho \in \{20.6, 20.4, 20.2, 20, 19.8\}$
	AT2	$\Box_{[0,30]}((gear = 4) \rightarrow (speed > \rho))$	$\rho \in \{43, 41, 39, 37, 35\}$
	AT3	$\Box_{[0,30]}((gear = 4) \rightarrow (rpm > \rho))$	$\rho \in \{700, 800, 900, 1000, 1100\}$
	AT4	$\Box_{[0,30-\tau]}((\delta_{10}(rpm) > 2000) \rightarrow (\delta_\tau(gear) > 0))$	$\tau \in \{15, 16, 17, 18, 19\}$
	AT5	$\Box_{[0,30]}((speed < \rho) \wedge (RPM < 4780))$	$\rho \in \{130, 131, 132, 133, 134, 135, 136, 137\}$
	AT6	$\Box_{[0,26]}((\delta_4(speed) > \rho) \rightarrow (\delta_4(gear) > 0))$	$\rho \in \{20, 25, 30, 35, 40\}$
	AT7	$\Box_{[0,30-\tau]}((\delta_\tau(speed) > 30) \rightarrow (\delta_\tau(gear) > 0))$	$\tau \in \{2, 3, 4, 5, 6, 7, 8\}$
AFC	AFC1	$\Box_{[11,50]}((controller\_mode = 0) \rightarrow (mu < \rho))$	$\rho \in \{0.16, 0.17, 0.18, 0.19, 0.2\}$
	AFC2	$\Box_{[11,50]}((controller\_mode = 1) \rightarrow (mu < \rho))$	$\rho \in \{0.222, 0.224, 0.226, 0.228, 0.23\}$
NN		$close \equiv \lvert Pos - Ref\rvert <= \rho + \alpha * \lvert Ref\rvert$	
		$reach \equiv \Diamond_{[0,2]}(\Box_{[0,1]}(close))$	
	NN1	$\Box_{[0,18]}(\neg close \rightarrow reach), \alpha = 0.04$	$\rho \in \{0.001, 0.002, 0.003, 0.004, 0.005\}$
	NN1	$\Box_{[0,18]}(\neg close \rightarrow reach), \alpha = 0.03$	$\rho \in \{0.001, 0.002, 0.003, 0.004, 0.005\}$

(b) Sbench

Spec ID	scaled output	factor 10^k
$AT1_1$		
$AT1_2$		
$AT1_3$	$speed$	$k \in$ {-2,0,1,3}
$AT1_4$		
$AT1_5$		
$AT5_4$		
$AT5_5$		
$AT5_6$	$speed$	$k \in$ {-2,0,1,3}
$AT5_7$		
$AT5_8$		
$AFC1_1$		
$AFC1_2$		
$AFC1_3$	mu	$k \in$ {0,1,2,3}
$AFC1_4$		
$AFC1_5$		

in Appendix A in the extended version [37][†3]. We report aggregated results in Table 3. For benchmark set Bbench, it reports aggregated results for each group of specifications obtained from S (i.e., all the different versions S_i obtained by changing the value of the parameter); for benchmark set Sbench, instead, results are aggregated for each scaled specification S^k (considering the versions S_i^k obtained by changing the parameter value). We report minimum, maximum and average number of successes SR, and time in seconds. For MAB-ϵ-greedy and MAB-UCB, both for SR and time, we also report the average percentage difference[†4] (Δ) w.r.t. to the corresponding value of Breach.

Comparison In the following, we compare two approaches $A_1, A_2 \in$ {Breach, MAB-ϵ-greedy, MAB-UCB } by comparing the number of their successes SR and average execution *time* using the non-parametric Wilcoxon signed-rank test with 5% level of significance[†5] [35]; the null hypothesis is that there is no difference in applying A_1 A_2 in terms of the compared measure (SR or time).

4.1 Evaluation

We evaluate the proposed approach with some research questions.

RQ1 *Which is the best MAB algorithm for our purpose?*

In § 3.2, we described that the proposed approach can be executed using two different strategies for choosing the arm in the MAB problem, namely MAB-ϵ-greedy and MAB-UCB. We here assess which one is better in terms of SR and time. From the results in Table 3, it seems that MAB-UCB provides slightly better performance in terms of SR; this has been confirmed by the Wilcoxon test applied over all the experiments (i.e., on the non-aggregated data reported in Appendix A in the extended version [37]): the null hypothesis that using anyone of the two strategies has no impact on SR is

†3 The code, models, and specifications are available online at https://github.com/ERATOMMSD/FalStar-MAB.

†4 Δ=$((m-b)*100)/(0.5*(m+b))$ where m is the result of MAB and b the one of Breach.

†5 We checked that the distributions are not normal with the non-parametric Shapiro-Wilk test.

表 3: Aggregated results for benchmark sets Bbench and Sbench (SR: # successes out 30 trials. Time in secs. Δ: percentage difference w.r.t. Breach). Outperformance cases are highlighted, indicated by positive Δ of SR, and negative Δ of time.

Spec.	Breach						MAB-ϵ-greedy								MAB-UCB							
ID	SR (/30)			time (sec.)			SR (/30)				time (sec.)				SR (/30)				time (sec.)			
	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Δ	Min	Max	Avg	Δ	Min	Max	Avg	Δ	Min	Max	Avg	Δ
AT1	14	25	20.2	125	361.2	223.1	24	30	28.6	35.7	62.7	213.4	106.4	−73.4	28	30	29.2	37.8	45.1	146.8	77.4	−97.1
AT2	11	30	20.2	14	390.6	209.8	30	30	30	43.9	11.9	126.3	54.5	−96.9	27	30	29.4	42.2	17.7	92.5	36.8	−112.1
AT3	29	30	29.4	2.3	22.2	14.2	30	30	30	2	2.5	7	3.5	−82.9	30	30	30	2	2.5	3.6	3	−88.6
AT4	18	30	25.8	19.5	265.3	109.6	29	30	29.8	16	7.8	45.1	24.4	−105	30	30	30	16.6	6.2	36.2	22.2	−113.5
AT5	6	23	14.1	203.1	525.9	366.2	26	30	28.5	72.1	35.2	149	93.7	−120.6	26	30	28.2	71.4	37.7	154.1	99.2	−116.8
AT6	5	29	22.8	30.1	509.5	157	21	30	27	28	2.3	300	95.1	−98.3	22	30	27	27.7	2.9	247.3	86.1	−99.4
AT7	15	30	26.6	12.2	314	81.5	20	30	28.6	8.4	2.9	283.9	49.9	−92	23	30	29	10.3	5.5	223.3	42.9	−88.3
AFC1	6	30	14.4	124.8	565.6	413.5	4	28	12	−28.4	171	568.4	446	10.8	5	30	16.4	9.7	98.7	559.8	389.9	−9.3
AFC2	2	30	18	80.7	582.3	343.4	5	30	20	23.8	43.2	547.8	301.9	−23.8	5	30	20	22.9	59.4	568.4	320.5	−11.1
NN1	17	25	20.8	212.9	384.7	292.9	14	27	20.2	−4.5	189.5	422.8	320.3	6.2	17	28	22.6	7.3	148.2	403.3	272.3	−11.8
NN2	27	28	27.2	55.5	93.4	73.1	30	30	30	9.8	11	39.3	26.3	−97.8	30	30	30	9.8	14.6	38.2	27.4	−92.3
$AT1^{-2}$	30	30	30	42.5	97.4	56.9	28	30	29	−3.4	75.6	178.3	118.7	68.7	28	30	29.4	−2.1	54.3	136.3	80.3	33.3
$AT1^{0}$	14	25	20.2	125	361.2	223.1	24	30	28.6	35.7	62.7	213.4	106.4	−73.4	28	30	29.2	37.8	45.1	146.8	77.4	−97.1
$AT1^{1}$	4	21	15.4	204.5	527.6	310.2	25	30	29	68.4	49	234.7	102.1	−108	27	29	28.2	64.5	77.5	128.7	105.1	−93
$AT1^{3}$	8	24	19.8	164	471.7	240.1	29	30	29.8	44.6	67.5	170.6	101.9	−77.3	29	30	29.4	43.4	55.4	104.8	80.6	−93.6
$AT5^{-2}$	29	30	29.6	61.1	163.7	102	25	30	27.8	−6.4	76.9	139.5	111.9	12.6	28	30	29.4	−0.7	48.5	131.9	85.7	−17
$AT5^{0}$	6	18	11.2	291.1	525.9	423.1	28	30	28.4	90.5	80.2	151.3	107.4	−117.7	26	30	28	89.4	68.3	154.1	114.9	−114.5
$AT5^{1}$	0	2	0.4	566.4	600	593.3	27	30	28.4	194.8	70.7	184.5	110.3	−138.5	25	30	27.6	194.1	83.1	150	123.7	−131.2
$AT5^{3}$	0	1	0.2	586.4	600	597.3	27	30	28.6	197.2	66.8	163.3	102.5	−142.3	27	29	28	197.2	80.4	160.9	111.9	−137.4
$AFC1^{0}$	6	30	14.4	124.8	565.6	413.5	4	29	16.4	8.5	115.1	559.9	411.1	−2.8	5	30	16.4	9.7	98.7	559.8	389.9	−9.3
$AFC1^{1}$	7	30	16.6	99	548.2	393.3	3	29	10.8	−60.9	198.1	587.6	465.8	24.6	7	29	17.8	10.3	105.7	527.3	354.3	−10.3
$AFC1^{2}$	0	12	5.2	434.4	600	535.8	3	28	11.6	96.2	180.8	577.6	463	−20.7	4	30	17	127	73.7	556.3	374.5	−47.3
$AFC1^{3}$	1	12	4.8	425.7	587.4	532.6	3	30	14.4	109	138	585.5	436.5	−28	7	30	15	113	77.1	553.4	403.7	−39.9

rejected with p-value equal to 0.005089, and the alternative hypothesis that SR is better is accepted with p-value=0.9975; in a similar way, the null hypothesis that there is no difference in terms of time is rejected with p-value equal to 3.495e-06, and the alternative hypothesis that is MAB-UCB is faster is accepted with p-value=1. Therefore, in the following RQs, we compare Breach with only the MAB-UCB version of our approach.

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

We here assess if our approach is effective in tackling the scale problem. Table 4 reports the complete experimental results over Sbench for Breach and MAB-UCB; for each specification S, all its scaled versions are reported in increasing order of the scaling factor. We observe that changing the scaling factor affects (sometimes greatly) the number of successes SR of Breach; for example, for $AT5_5$ and $AT5_7$ it goes from 30 to 0. For MAB-UCB, instead, SR is similar across the scaled versions of each specification: this shows that the approach is robust w.r.t. to the scale problem as the "hill-climbing gain" reward in Def. 11 eliminates the impact of scaling and UCB1 algorithm balances the exploration and exploitation of two sub-formulas. The observation is confirmed by the Wilcoxon test over SR: the null hypothesis is rejected with p-value=1.808e-09, and the alternative

表 4: Experimental results – Sbench (SR: # successes out of 30 trials. Time in secs)

Spec.	Breach		MAB-UCB		Spec.	Breach		MAB-UCB		Spec.	Breach		MAB-UCB	
ID	SR (/30)	time (sec.)	SR (/30)	time (sec.)	ID	SR (/30)	time (sec.)	SR (/30)	time (sec.)	ID	SR (/30)	time (sec.)	SR (/30)	time (sec.)
$AT1_1^{-2}$	30	51.3	30	54.3	$AT5_4^{-2}$	30	61.1	30	48.5	$AFC1_1^0$	30	124.8	30	98.7
$AT1_1^0$	25	125	29	75	$AT5_4^0$	18	291.1	28	94.5	$AFC1_1^1$	30	99	29	105.7
$AT1_1^1$	20	221.1	28	107.9	$AT5_4^1$	2	566.4	25	150	$AFC1_1^2$	12	434.4	30	73.7
$AT1_1^3$	23	170	29	55.4	$AT5_4^3$	1	586.4	28	96.2	$AFC1_1^3$	12	425.7	30	77.1
$AT1_2^{-2}$	30	49	29	67.5	$AT5_5^{-2}$	30	71.3	29	67.8	$AFC1_2^0$	16	421.5	23	346.8
$AT1_2^0$	22	187.5	30	45.1	$AT5_5^0$	15	369.1	27	114	$AFC1_2^1$	25	345.9	27	227.9
$AT1_2^1$	21	204.5	29	77.5	$AT5_5^1$	0	600	29	83.1	$AFC1_2^2$	8	497.2	25	320.5
$AT1_2^3$	24	164	30	61	$AT5_5^3$	0	600	27	113.8	$AFC1_2^3$	5	518.1	21	364
$AT1_3^{-2}$	30	42.5	30	62.4	$AT5_6^{-2}$	29	110.2	28	103.3	$AFC1_3^0$	11	457.7	15	442
$AT1_3^0$	19	239.5	29	62.5	$AT5_6^0$	10	438.2	30	68.3	$AFC1_3^1$	13	479.2	14	455.5
$AT1_3^1$	16	296.2	27	128.7	$AT5_6^1$	0	600	27	126.7	$AFC1_3^2$	2	590.7	15	453.2
$AT1_3^3$	21	209.8	30	93.4	$AT5_6^3$	0	600	29	80.4	$AFC1_3^3$	5	545.6	8	510.6
$AT1_4^{-2}$	30	44.5	30	80.8	$AT5_7^{-2}$	30	103.6	30	77.3	$AFC1_4^0$	9	498.2	9	502.1
$AT1_4^0$	21	202.2	30	57.4	$AT5_7^0$	7	491.4	26	154.1	$AFC1_4^1$	8	494	12	455
$AT1_4^1$	16	301.7	28	119.5	$AT5_7^1$	0	600	27	134.3	$AFC1_4^2$	4	556.8	11	468.7
$AT1_4^3$	23	185.1	29	88.3	$AT5_7^3$	0	600	29	108	$AFC1_4^3$	1	587.4	9	513.4
$AT1_5^{-2}$	30	97.4	28	136.3	$AT5_8^{-2}$	29	163.7	30	131.9	$AFC1_5^0$	6	565.6	5	559.8
$AT1_5^0$	14	361.2	28	146.8	$AT5_8^0$	6	525.9	29	143.6	$AFC1_5^1$	7	548.2	7	527.3
$AT1_5^1$	4	527.6	29	91.9	$AT5_8^1$	0	600	30	124.2	$AFC1_5^2$	0	600	4	556.3
$AT1_5^3$	8	471.7	29	104.8	$AT5_8^3$	0	600	27	160.9	$AFC1_5^3$	1	586	7	553.4

hypothesis accepted with p-value=1. Instead, the null hypothesis that there is no difference in terms of time cannot be rejected with p-value=0.3294.

RQ3 *How does the proposed process behave with not scaled benchmarks?*

In RQ2, we checked whether the proposed approach is able to tackle the scale problem for which it has been designed. Here, instead, we are interested in investigating how it behaves on specifications that have not been artificially scaled (i.e., those in Bbench). From Table 3 (upper part), we observe that MAB-UCB is always better than Breach both in terms of SR and time, which is shown by the highlighted cases. This is confirmed by Wilcoxon test over SR and time: null hypotheses are rejected with p-values equal to, respectively, 6.02e-08 and 1.41e-08, and the alternative hypotheses that MAB-UCB is better are both accepted with p-value=1. This means that the proposed approach can also handle specifications that do not suffer from the scale problem, and so it can be used with any kind of specification.

RQ4 *Is the proposed approach more effective than an approach based on rescaling?*

A naïve solution to the scale problem could be to rescale the signals used in specification at the same scale. Thanks to the results of RQ2, we can compare to this possible baseline approach, using the scaled benchmark set Sbench. For example, AT5 suffers from the scale problem as *speed* is one order of magnitude less than rpm. However, from Table 3, we observe that the scaling that would be done by the baseline approach (i.e., running Breach over $AT5^1$) is not effective, as SR is 0.4/30, that is much lower than the original SR 14.1/30 of the unscaled approach using Breach. Our approach, instead, raises SR to 28.4/30 and to 27.6/30 using the two proposed versions. By monitoring Breach execution, we notice that the naïve approach fails because it tries to falsify $rpm<4780$, which, however, is not falsifiable; our approach, instead, understands that it must try to falsify

$speed < \rho$. More details are given in the extended version [37].

5 Conclusion and Future work

In this paper, we propose a solution to the *scale problem* that affects falsification of specifications containing Boolean connectives. The approach combines multi-armed bandit algorithms with hill climbing-guided falsification. Experiments show that the approach is robust under the change of scales, and it outperforms a state-of-the-art falsification tool. The approach currently handles binary specifications. As future work, we plan to generalize it to complex specifications having more than two Boolean connectives.

謝辞 The authors are supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST.

参考文献

[1] Adimoolam, A., Dang, T., Donzé, A., Kapinski, J., and Jin, X.: Classification and Coverage-Based Falsification for Embedded Control Systems, *Computer Aided Verification*, Cham, Springer International Publishing, 2017, pp. 483–503.

[2] Akazaki, T. and Hasuo, I.: Time Robustness in MTL and Expressivity in Hybrid System Falsification, *Computer Aided Verification*, Cham, Springer International Publishing, 2015, pp. 356–374.

[3] Akazaki, T., Kumazawa, Y., and Hasuo, I.: Causality-Aided Falsification, *Proceedings First Workshop on Formal Verification of Autonomous Vehicles, FVAV@iFM 2017, Turin, Italy, 19th September 2017.*, EPTCS, Vol. 257, 2017, pp. 3–18.

[4] Akazaki, T., Liu, S., Yamagata, Y., Duan, Y., and Hao, J.: Falsification of Cyber-Physical Systems Using Deep Reinforcement Learning, *Formal Methods - 22nd International Symposium, FM 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 15-17, 2018, Proceedings*, Havelund, K., Peleska, J., Roscoe, B., and de Vink, E. P.(eds.), Lecture Notes in Computer Science, Vol. 10951, Springer, 2018, pp. 456–465.

[5] Annpureddy, Y., Liu, C., Fainekos, G., and Sankaranarayanan, S.: *S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems*, Springer, 2011, pp. 254–257.

[6] Auger, A. and Hansen, N.: A restart CMA evolution strategy with increasing population size, *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005*, IEEE, 2005, pp. 1769–1776.

[7] Beale, M. H., Hagan, M. T., and Demuth, H. B.: Neural Network Toolbox™ User's Guide, *The Mathworks Inc*, (1992).

[8] Chen, X., Ábrahám, E., and Sankaranarayanan, S.: Flow*: An Analyzer for Non-linear Hybrid Systems, *Computer Aided Verification*, Berlin, Heidelberg, Springer Berlin Heidelberg, 2013, pp. 258–263.

[9] Deshmukh, J., Jin, X., Kapinski, J., and Maler, O.: Stochastic Local Search for Falsification of Hybrid Systems, *Automated Technology for Verification and Analysis*, Cham, Springer International Publishing, 2015, pp. 500–517.

[10] Dokhanchi, A., Yaghoubi, S., Hoxha, B., and Fainekos, G. E.: Vacuity aware falsification for MTL request-response specifications, *13th IEEE Conference on Automation Science and Engineering, CASE 2017, Xi'an, China, August 20-23, 2017*, IEEE, 2017, pp. 1332–1337.

[11] Donzé, A.: Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems, *Computer Aided Verification, 22nd Int. Conf., CAV 2010*, LNCS, Vol. 6174, Springer, 2010, pp. 167–170.

[12] Donzé, A. and Maler, O.: Robust Satisfaction of Temporal Logic over Real-Valued Signals, *Formal Modeling and Analysis of Timed Systems - 8th Int. Conf., FORMATS 2010*, LNCS, Vol. 6246, Springer, 2010, pp. 92–106.

[13] Dreossi, T., Dang, T., Donzé, A., Kapinski, J., Jin, X., and Deshmukh, J. V.: Efficient Guiding Strategies for Testing of Temporal Properties of Hybrid Systems, *NASA Formal Methods*, Cham, Springer International Publishing, 2015, pp. 127–142.

[14] Dreossi, T., Dang, T., and Piazza, C.: Parallelotope Bundles for Polynomial Reachability, *Proc. of the 19th International Conference on Hybrid Systems: Computation and Control*, HSCC '16, New York, NY, USA, ACM, 2016, pp. 297–306.

[15] Dreossi, T., Donzé, A., and Seshia, S. A.: Compositional Falsification of Cyber-Physical Systems with Machine Learning Components, *NASA Formal Methods*, Cham, Springer International Publishing, 2017, pp. 357–372.

[16] Ernst, G., Arcaini, P., Donzé, A., Fainekos, G., Mathesen, L., Pedrielli, G., Yaghoubi, S., Yamagata, Y., and Zhang, Z.: ARCH-COMP 2019 Category Report: Falsification, *ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems*, Frehse, G. and Althoff, M.(eds.), EPiC Series in Computing, Vol. 61, EasyChair, 2019, pp. 129–140.

[17] Fainekos, G. E. and Pappas, G. J.: Robustness of temporal logic specifications for continuous-time signals, *Theor. Comput. Sci.*, Vol. 410, No. 42(2009), pp. 4262–4291.

[18] Fan, C., Qi, B., Mitra, S., Viswanathan, M., and Duggirala, P. S.: Automatic Reachability Analysis for Nonlinear Hybrid Models with C2E2, *Computer Aided Verification*, Cham, Springer International Publishing, 2016, pp. 531–538.

[19] Ferrère, T., Nickovic, D., Donzé, A., Ito, H., and Kapinski, J.: Interface-aware signal temporal logic, *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, Montreal,*

QC, Canada, April 16-18, 2019., Ozay, N. and Prabhakar, P.(eds.), ACM, 2019, pp. 57–66.

[20] Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., and Maler, O.: SpaceEx: Scalable Verification of Hybrid Systems, *Computer Aided Verification*, Berlin, Heidelberg, Springer, 2011, pp. 379–395.

[21] Fu, Z. and Su, Z.: XSat: A Fast Floating-Point Satisfiability Solver, *Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II*, Chaudhuri, S. and Farzan, A.(eds.), Lecture Notes in Computer Science, Vol. 9780, Springer, 2016, pp. 187–209.

[22] Gao, S., Avigad, J., and Clarke, E. M.: δ-Complete Decision Procedures for Satisfiability over the Reals, *Automated Reasoning*, Berlin, Heidelberg, Springer, 2012, pp. 286–300.

[23] Hasuo, I. and Suenaga, K.: Exercises in Nonstandard Static Analysis of Hybrid Systems, *Computer Aided Verification*, Berlin, Heidelberg, Springer, 2012, pp. 462–478.

[24] Hoxha, B., Abbas, H., and Fainekos, G. E.: Benchmarks for Temporal Logic Requirements for Automotive Systems, *1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems, ARCH@CPSWeek 2014, Berlin, Germany, April 14, 2014 / ARCH@CPSWeek 2015, Seattle, USA, April 13, 2015.*, Frehse, G. and Althoff, M.(eds.), EPiC Series in Computing, Vol. 34, EasyChair, 2014, pp. 25–30.

[25] Jin, X., Deshmukh, J. V., Kapinski, J., Ueda, K., and Butts, K.: Powertrain Control Verification Benchmark, *Proc. of the 17th Int. Conf. on Hybrid Systems: Computation and Control*, HSCC '14, NY, USA, ACM, 2014, pp. 253–262.

[26] Kapinski, J., Deshmukh, J. V., Jin, X., Ito, H., and Butts, K.: Simulation-Based Approaches for Verification of Embedded Control Systems: An Overview of Traditional and Advanced Modeling, Testing, and Verification Techniques, *IEEE Control Systems*, Vol. 36, No. 6(2016), pp. 45–64.

[27] Kato, K., Ishikawa, F., and Honiden, S.: Falsification of Cyber-Physical Systems with Reinforcement Learning, *3rd Workshop on Monitoring and Testing of Cyber-Physical Systems, MT@CPSWeek 2018, Porto, Portugal, April 10, 2018*, IEEE, 2018, pp. 5–6.

[28] Kuřátko, J. and Ratschan, S.: Combined Global and Local Search for the Falsification of Hybrid Systems, *Formal Modeling and Analysis of Timed Systems*, Cham, Springer International Publishing, 2014, pp. 146–160.

[29] Liebrenz, T., Herber, P., and Glesner, S.: Deductive Verification of Hybrid Control Systems Modeled in Simulink with KeYmaera X, *Formal Methods and Software Engineering - 20th International Conference on Formal Engineering Methods, ICFEM 2018, Gold Coast, QLD, Australia, November 12-16, 2018, Proceedings*, Sun, J. and Sun, M.(eds.), Lecture Notes in Computer Science, Vol. 11232, Springer, 2018, pp. 89–105.

[30] Luersen, M. A. and Le Riche, R.: Globalized Nelder–Mead method for engineering optimization, *Computers & Structures*, Vol. 82, No. 23(2004), pp. 2251–2260.

[31] Nguyen, L. V., Kapinski, J., Jin, X., Deshmukh, J. V., Butts, K., and Johnson, T. T.: Abnormal Data Classification Using Time-Frequency Temporal Logic, *Proc. of the 20th International Conference on Hybrid Systems: Computation and Control*, HSCC '17, New York, NY, USA, ACM, 2017, pp. 237–242.

[32] Platzer, A.: *Logical Foundations of Cyber-Physical Systems*, Springer, 2018.

[33] Seshia, S. A. and Rakhlin, A.: Quantitative Analysis of Systems Using Game-Theoretic Learning, *ACM Trans. Embed. Comput. Syst.*, Vol. 11, No. S2(2012), pp. 55:1–55:27.

[34] Silvetti, S., Policriti, A., and Bortolussi, L.: An Active Learning Approach to the Falsification of Black Box Cyber-Physical Systems, *Integrated Formal Methods*, Cham, Springer International Publishing, 2017, pp. 3–17.

[35] Wohlin, C., Runeson, P., Hst, M., Ohlsson, M. C., Regnell, B., and Wessln, A.: *Experimentation in Software Engineering*, Springer Publishing Company, Incorporated, 2012.

[36] Zhang, Z., Ernst, G., Sedwards, S., Arcaini, P., and Hasuo, I.: Two-Layered Falsification of Hybrid Systems Guided by Monte Carlo Tree Search, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 37, No. 11(2018), pp. 2894–2905.

[37] Zhang, Z., Hasuo, I., and Arcaini, P.: Multi-Armed Bandits for Boolean Connectives in Hybrid System Falsification (Extended Version), *CoRR*, Vol. abs/1905.07549(2019).

[38] Zutshi, A., Deshmukh, J. V., Sankaranarayanan, S., and Kapinski, J.: Multiple shooting, CEGAR-based falsification for hybrid systems, *2014 International Conference on Embedded Software, EMSOFT 2014, New Delhi, India, October 12-17, 2014*, ACM, 2014, pp. 5:1–5:10.