Falsification and Control of CPS using the Language Set of Discrete-Time Temporal Logic

Christian Abou-Mrad*
Oregon State University
Corvallis, OR, US
aboumrac@oregonstate.edu

Tanmay Khandait*
Arizona State University
Tempe, AZ, US
tkhandai@asu.edu

Giulia Pedrielli
Arizona State University
Tempe, AZ, US
Giulia.Pedrielli@asu.edu

Houssam Abbas
Oregon State University
Corvallis, OR, US
abbasho@oregonstate.edu

ABSTRACT

We present algorithms for Cyber-Physical Systems (CPS) falsification and control, which take advantage of knowing the entire language of the temporal logic specification - that is, the set of signals that satisfy the formula. In the design of CPS, falsification and control play key roles. Falsification is a testing task, where the goal is to find an input signal that causes the system's output trajectory to violate the correctness requirements. Control is the dual task, where the goal is to find an input signal that causes the system's output to satisfy the specification. When the specification is expressed in a temporal logic, most existing work relies on local optimization heuristics to perform both tasks. In this paper, we explore whether a different expression of the specification offers advantages when performing falsification and control. Recent work presented a method for computing a representation of the language of a formula in (discrete-time) Signal Temporal Logic (STL), showing that the language can be represented as a union of polytopes. We introduce new falsification algorithms which combine distance information to the different components of the language to accelerate the convergence to a falsifier. And we introduce a new algorithm for computing a satisfying control signal which works by repeatedly projecting violating output trajectories back onto the language's components. Moreover, these algorithms are trivially parallelizable to take advantage of multiple processors. Despite their relative simplicity, our algorithms demonstrate 10x to 100x speedups relative to the state-of-the-art.

CCS CONCEPTS

• **Theory of computation → Logic**; • **Computer systems organization → Embedded and cyber-physical systems**.

KEYWORDS

Temporal logic, falsification, control, System design

ACM Reference Format:
Christian Abou-Mrad, Tanmay Khandait, Giulia Pedrielli, and Houssam Abbas. 2025. Falsification and Control of CPS using the Language Set of Discrete-Time Temporal Logic. In *ACM/IEEE 16th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2025) (ICCPS '25), May 6–9, 2025, Irvine, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3716550.3722040

*Equal contribution.

ICCPS '25, May 6–9, 2025, Irvine, CA, USA

ACM ISBN 979-8-4007-1498-6/2025/05…$15.00
https://doi.org/10.1145/3716550.3722040

1 INTRODUCTION

When designing Cyber-Physical Systems (CPS), it has become common practice to formalize the system's requirements in a temporal logic, such as Signal and Metric Temporal Logics (STL and MTL) [16, 31]. A formal requirement in STL, say, has the advantage of being unambiguous, because the logic has mathematical semantics that are not open to interpretation. Perhaps more importantly, several system design tasks can be automated given a requirement in formal logic, whether falsification [2], control [48], runtime monitoring [8], temporal logic inference [22, 30], and example generation [6, 10, 24, 25, 43, 46].

This paper focuses on the first two of these: falsification and control. Falsification is a testing task, where the goal is to find an input control signal that causes the system's output trajectory to violate the formal specification. Control is the task of finding a control signal that causes the system's output to satisfy the specification. Some limited cases (like linear control systems) are amenable to exhaustive approaches for certain fragments of the applicable logic, e.g. [44, 52]. However, this exhaustiveness comes at the cost of solving expensive mixed-integer optimizations or abstraction refinements *at control time*. More generally, since most CPS defy full mathematical modeling, the CPS is usually treated as a black-box (or a combination of black-box and white-box components). Therefore, most works in the literature use search heuristics and optimizations to perform falsification and control, from simulated annealing [1] to Bayesian optimization [14] and others [36]. These heuristics have met varying degrees of success, with different success/runtime trade-offs depending on the formula and the system being falsified or controlled. The yearly ARCH workshop keeps track of the state-of-the-art in falsification [12].

Recent work opens the way to a different approach, which shifts the bulk of the computational burden to a one-time offline computation, while dramatically accelerating the tasks when it matters: at control time, or at falsification time. Specifically, in [5], an algorithm is given for approximating the *language* of a discrete-time STL formula – that is, the set of all signals that satisfy it. Their algorithm for approximating the language is expensive, as might be expected (although the authors of [5] communicated that theirs is an un-optimized implementation [4]). The conjecture we test in this paper is that the language can be used to perform both falsification and control faster than would be possible without access to the language, potentially much faster, thus amortizing the cost of approximating the language over multiple uses of the language to falsify and/or control many different systems.

Contributions. We work with specifications ϕ in discrete-time STL, or DT-STL. For such formulas, [5] showed that the language $\mathcal{L}(\phi)$ is a union of polytopes, and gave a way to approximate this union. In this paper, we show that by incorporating the distance of a candidate falsifying signal to each polytope in the language individually into an existing falsification algorithm, we can accelerate falsification runtime by up to 100x, and generally reduce the number of tests needed to reach a falsifier (though the amount of reduction varies). We also show that by treating each polytope of the language as its own objective, we can find a successful control signal for a tracking dynamical system in as little as two iterations. Despite the simplicity of our algorithms, the experimental evidence suggests that this is a promising approach to falsification and control of CPS, and that further theoretical analysis is needed to understand it and build upon it.

Related Work. We examine the related work across three primary themes at the basis of this work.

Language Generation This paper relies on the availability of a concise explicit description of the language of the temporal logic formula. The only work we know of that provides such an explicit description is [5], in which it is shown that the language of DT-STL formulas is a union of polytopes.

Search-based Test Generation (SBTG) for CPS: SBTG, also known as falsification, refers to the problem of searching the operational space of a system to identify anomalous behavior with respect to a set of expert-defined requirements [28]. Various methods, ranging from black-box optimization [2, 14] to tree exploration techniques [19, 55], have been proposed and effectively applied to address the falsification task. In this line of work, the search for a falsifying input is guided by a distance measure called robustness, which quantifies the degree of satisfaction or violation of the output relative to the defined requirements. However, the robustness function is notoriously difficult to optimize due to its highly non-linear, non-convex, and possibly discontinuous surface. This complexity has motivated significant research into statistical and global optimization-based methods [9, 11]. Increasingly, there is a focus on providing guarantees for finding such anomalous behavior [42, 50, 54]. In addition to algorithmic advances, several software tools have been developed to address the falsification problem, as highlighted in the falsification category of the ARCH competition [29]. In this work, we use Psy-TaLiRo [51] due to its ease of developing and integrating new monitors and optimizers.

Control Control algorithms with (DT-)STL objectives maximize robustness. State-of-the-art approaches use global optimizers that work with an analytic expression of the robustness [38, 39], while others encode the problem as a mixed-integer linear program [23, 44, 45]. Depending on the level of control (motion planning, tracking and actuation, or both), the robustness maximization is constrained by different dynamic feasibility constraints [40]. Recent work proposes decentralized control for different classes of formulas, surveyed in the upcoming monograph [32].

Organization. After preliminaries in Section 2, Section 3 provides new ways to do falsification when the formula's language is available. Section 4 provides a way to compute control signals for nonlinear dynamical systems given the language. Section 5 concludes.

2 PRELIMINARIES

We define the logic DT-STL and its robustness, give the properties of its language, and provide background on CPS falsification.

Notation. Let $\mathfrak{R} = (-\infty, \infty)$ and $\mathbb{N} = \{0, 1, 2, \ldots\}$. With two integers a and b, $[a : b] = \{a, a+1, \ldots, b\} \subset \mathbb{N}$. Given an integer interval $I \subset \mathbb{N}$ and $t \in \mathbb{N}$, $t + I := \{t + s \mid s \in I\}$. We write $E = \{\vec{e}_1, \ldots, \vec{e}_N\}$ for the Euclidian basis of $\mathfrak{R}^N$. Given two vectors $x, y \in \mathfrak{R}^N$, the L_2 distance between them is $\|x - y\|_2 := \sqrt{\sum_n (x(n) - y(n))^2}$.

A *polytope* P in $\mathfrak{R}^N$ is the bounded intersection of half-spaces. A *box* is a polytope of the form $[a_1, b_1] \times \ldots \times [a_N, b_N]$ with real numbers a_n, b_n.

2.1 DT-STL

Signal Temporal Logic (STL) [18, 33] is a logic that allows the succinct and unambiguous specification of a wide variety of desired system behaviors over time, such as "The vehicle reaches its destination within 10 time units while always avoiding obstacles" and "While the vehicle is in Zone 1, it must obey that zone's velocity constraints". STL is defined over continuous-time signals. We use a variant of STL which applies to discrete-time signals, and which we call DT-STL. DT-STL has been used frequently in control synthesis such as in [23, 35, 44]. Formally, let $X \subset \mathfrak{R}^d$ be the state-space. A *signal* x is a map from $\mathbb{N}$ to X, $x : \mathbb{N} \to X$. The set of all signals is denoted $X^{\mathbb{N}}$. Let $AP = \{p_1, \ldots, p_L\}$ be a set of atomic propositions, and let $\{\mu_p \mid p \in AP\}$ be a corresponding set of real-valued functions: $\mu_p : X \to \mathfrak{R}$.

Definition 2.1 (Discrete-Time STL (DT-STL)). The syntax of the logic is given by

$$\phi := \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \, \mathcal{U}_I \, \phi$$

where $p \in AP$ and $I \subseteq \mathbb{N}$ is an integer interval. The semantics are given relative to signals as follows.

$$\begin{aligned}
(x,t) &\models \top \\
(x,t) &\models p \text{ iff } \mu_p(x(t)) \leq 0 \\
(x,t) &\models \neg\phi \text{ iff } (x,t) \not\models \phi \\
(x,t) &\models \phi_1 \wedge \phi_2 \text{ iff } (x,t) \models \phi_1 \text{ and } (x,t) \models \phi_2 \\
(x,t) &\models \phi_1 \mathcal{U}_I \phi_2 \text{ iff } \exists t' \in t + I \,.\, (x,t') \models \phi_2 \text{ and} \\
&\qquad \forall t'' \in [t : t' - 1], \;\; (x,t'') \models \phi_1
\end{aligned}$$

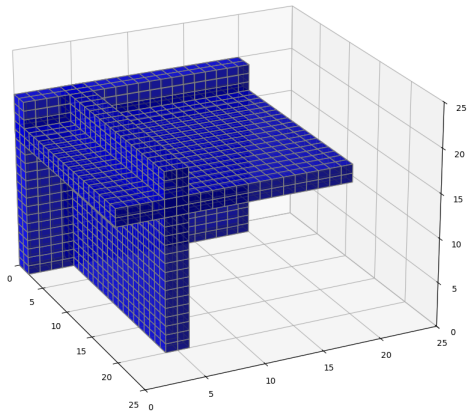

Figure 1: Example of a 3D projection of a DT-STL formula's language (ϕ_4 in Eq. (17)).

The *language of* ϕ is the set of all signals that satisfy it at $t = 0$:

$$\mathcal{L}(\phi) := \{x \in X^{\mathbb{N}} \mid (x, 0) \models \phi\}$$

The disjunction, Eventually and Always operators are derived from conjunction and Until in the usual way. It's worth emphasizing that the language does not depend on any system – it is entirely a function of the formula itself.

Robustness. Falsification for CPS with temporal logic requirements relies on the notion of a *robustness function* [2, 3, 16, 20]. Given a DT-STL formula ϕ with horizon N, the robustness function $\rho_\phi : X^N \to \mathfrak{R}$ approximates the signed distance between a signal x and the language $\mathcal{L}(\phi)$. Specifically, the distance between two signals is measured using the L_∞ metric $\|x-y\|_\infty := \max_t \|x(t)-y(t)\|_2$. The distance between a signal $x \in X^N$ and a set of signals $A \subset X^{\mathbb{N}}$ is $\mathbf{dist}(x, A) := \inf_{a\in A}\|x - a\|_\infty$. Write ∂A for the boundary of a set A. Then robustness has the following property:

Theorem 2.2 (Robustness approximates distance [2, 16]). *For any DT-STL formula ϕ and signal x, if $\rho_\phi(x) > 0$ then $x \in \mathcal{L}(\phi)$ and $\rho_\phi(x) < \mathbf{dist}(x, \partial\mathcal{L}(\phi))$, while if $\rho_\phi(x) < 0$ then $x \notin \mathcal{L}(\phi)$ and $\rho_\phi(x) > -\mathbf{dist}(x, \mathcal{L}(\phi))$.*

Thus robustness gives us a conservative measure of how far a satisfying signal ($x \in \mathcal{L}(\phi)$) is from the language's boundary, and a conservative measure of how far a violating signal ($x \notin \mathcal{L}(\phi)$) is from the language. We define the *signed distance* function

$$\mathbf{Dist}(x, A) := \begin{cases} \mathbf{dist}(x, \partial A), & x \in A \\ -\mathbf{dist}(x, A), & x \notin A \end{cases} \tag{1}$$

In particular if we define the *robustness-induced language* $\mathcal{L}_\rho(\phi) := \{x \in X^N \mid \rho_\phi(x) > 0\}$ we have that

$$\mathcal{L}_\rho(\phi) \subseteq \mathcal{L}(\phi)$$

2.2 Geometry of the Language of DT-STL

We will need the following results about the language of a DT-STL formula from [5]:

Theorem 2.3. *Suppose that (1) The state space X is a bounded axis-aligned box: $X = [a_1, b_1] \times \ldots \times [a_d, b_d]$, (2) The functions μ_p are affine, and (3) all formulas we consider have a finite horizon, that is, for every formula there exists a smallest integer hrz_ϕ s.t. every satisfying signal has a satisfying prefix of length at most hrz_ϕ [15]. Then the language of a DT-STL formula with horizon N is the union of polytopes in $\mathfrak{R}^{Nd}$.*

See Figure 1. If additionally the signal is one-dimensional ($d = 1$), each atom is of the form $\ell \le x \le r$ for some scalars ℓ, r such that $[\ell, r] \subset X$, and this corollary follows.

Corollary 2.4. *If the signals are 1-dimensional, then under the assumptions of Theorem 2.3, the language of any DT-STL formula ϕ is a union of axis-aligned boxes in $\mathfrak{R}^{hrz_\phi}$.*

In the rest of this paper we work with one-dimensional signals, so for a given formula ϕ, each signal can be treated as a vector in X^{hrz_ϕ}, and $\mathcal{L}_\rho(\phi)$ is a subset of X^{hrz_ϕ}. In [5] an algorithm is given for approximating $\mathcal{L}_\rho$. Their experiments demonstrate a high accuracy in the approximation. In the rest of the paper, whenever referring to the language, we mean the approximation returned by the algorithms of [5]. We will often denote the approximate language simply as $\mathcal{L}$.

2.3 Falsification

Given a System-under-test (SUT) $\mathcal{M}$ and a formula ϕ *falsification*, refers to the problem of finding inputs $\mathbf{u} \in \mathbf{U}$ to the SUT that causes its output $x = \mathcal{M}(\mathbf{u}) \in X^N$ to violate the formula. Here N is the signal length. The search for a falsifier is framed as a robustness optimization, namely

$$\mathbf{u}^* = \arg\min_{\mathbf{u}\in\mathbf{U}} \rho_\phi(x) \tag{2}$$

If, at $\mathbf{u}^*$, the $\rho_\phi(x) < 0$ then, as established in Thm. 2.2, x violates ϕ. Solving the optimization (4) requires two routines: a *robustness monitor* that computes $\rho_\phi(x)$ for a given signal x, and an optimization routine to minimize ρ_ϕ. Multiple tools for DT-STL robustness monitoring have been proposed [13, 17, 37, 53]. We use the state-of-the-art RTAMT [37] monitor.

3 FALSIFICATION WITH THE LANGUAGE

For any given DT-STL formula ϕ interpreted over length-N signals, Corollary 2.4 asserts that the language is a union of axis-aligned boxes and it computes an approximation to that union:

$$\mathcal{L}(\phi) = \mathcal{B}^\phi := \bigcup_{k=1}^{j} b_k^\phi \tag{3}$$

where j denotes the number of boxes, and $b_k^\phi \subset \mathfrak{R}^N$ is the k^{th} N-dimensional box for ϕ. The signal space $X^{\mathbb{N}}$ satisfies $X^{\mathbb{N}} = \mathcal{B}^\phi \cup \mathcal{B}^{\neg\phi}$. Because each finite horizon discrete time signal is represented as an N-dimensional point in $X^{\mathbb{N}}$, we will use the words 'signal' and 'point' interchangeably.

Existing falsification algorithms minimize robustness, which amounts to minimizing (a conservative bound on) the distance between x and the language $\mathcal{L}(\phi)$ (Thm. 2.2) *using only oracle access to that distance.* Having the language in hand presents an intriguing alternative: compute $\mathcal{B}^{\neg\phi}$ and find points within this union. This

reduces the task of falsification to finding a trajectory within *any* box $b_k^{\neg\phi}$. Formally, eqn. (2) becomes:

$$\mathbf{u}^* = \arg\max_{\mathbf{u}\in\mathbf{U}} \mathbf{Dist}\left(x, b_k^{\neg\phi}\right), \tag{4}$$

where, if $\mathbf{Dist}\left(x, b_k^{\neg\phi}\right) > 0$, $\mathbf{u}^*$ is a falsifier. The remainder of this section provides the technical approach, followed by the experimental results on falsification task for CPS benchmarks.

3.1 Technical Approach

In the following subsection, we demonstrate two possible falsification frameworks that directly exploit the negated language reconstructed through the union of boxes $\mathcal{B}^{\neg\phi}$.

3.1.1 Parallelized Robustness Optimization for Falsification (PROF). Our new algorithm PROF is introduced to identify an approximate solution to Eq. (4) efficiently by exploiting the box structure of the language. PROF attempts to find falsifiers by working with each box in parallel, independently of the other boxes. For each box, the sampling distribution is incrementally updated using a Bayesian Optimization (BO) approach [21]. In the following, we describe such sampling performed in each of the j boxes.

Fix a box $b_k^{\neg\phi}$. PROF is initialized by sampling for each box, a set of n_0 input signals $\{\boldsymbol{u}_i\}$, resulting in n_0 output trajectories $\{x_i\}$. The distances $\mathbf{Dist}\left(x_i, b_k^{\neg\phi}\right)$ are computed, resulting into the initial PROF filtration $\mathcal{D}_k = \left\{\boldsymbol{u}_i, \mathbf{Dist}\left(x_i, b_k^{\neg\phi}\right)\right\}_{i=1}^{n_0}$.

In case $\mathbf{Dist}\left(x_i, b_k^{\neg\phi}\right)$ is non-positive for all i, the k-th box initiates the BO sampling phase. A Gaussian process (GP) is estimated using the filtration $\mathcal{D}_k$ as training set [49]. This surrogate model is used to build a sampling posterior (the core of the BO approach) in the input space referred to as an *acquisition function* [21]. In our implementation we use the Expected Improvement (EI) acquisition function and select the next input as its maximizer:

$$\begin{aligned} u^*_{iter+1,k} &\in \arg\max_u EI\,(u), \text{ where} \\ &EI\,(u) \\ &= \max\left[0, (y^* - \hat{y}_k(\boldsymbol{u}))\Phi\left(\frac{y^* - \hat{y}_k(\boldsymbol{x})}{\hat{s}_k(\boldsymbol{u})}\right) + \right. \\ &\qquad \left. + \hat{s}_k(\boldsymbol{u})\phi\left(\frac{y^* - \hat{y}_k(\boldsymbol{u})}{\hat{s}_k(\boldsymbol{u})}\right)\right], \end{aligned} \tag{5}$$

where, $\hat{y}_k\,(u)$ is the prediction for the signed distance produced by the GP surrogate at location u, $\hat{s}_k\,(u)$ is the associated model variance, while y^* is the best distance value within $\mathcal{D}_k$. Finally ϕ, and Φ refer here to the pdf and cdf of the standard normal distribution [27].

Obtained $\boldsymbol{u}^*_{iter+1,k}$, we simulate the associated trace $x^*_{iter+1,k}$, and evaluate the metric $\mathbf{Dist}\left(x^*_{iter+1,k}, b_k^{\neg\phi}\right)$. This results in the box-filtration $\mathcal{D}_k$ update. Unless the termination condition is met, the algorithm proceeds with the BO sampling stage.

At any iteration, if $\mathbf{Dist}\left(x^*_{iter+1,k}, b_k^{\neg\phi}\right) > 0$ for any box $k = 1, \ldots, j$, PROF terminates, otherwise PROF terminates once all boxes have sampled a maximum of n_{BO} inputs (user defined).

The pseudocode in Alg. 1 summarizes the approach. It should be noted that the choice of optimization algorithm to solve Eq.(4) can be replaced with any other optimizer.

Algorithm 1 Parallelized Robustness Optimization for Falsification (PROF)

Input: Search Space $\mathbf{U}$, distance $\mathbf{Dist}\left(x, b_k^{\neg\phi}\right)$, initialization budget n_0, total number of BO iterations n_{BO};
Output: location and distance value $\mathbf{u}^* \in \mathbf{U}$, $\mathbf{Dist}\left(x^*, b_k^{\neg\phi}\right) \geq 0$;

for b_k, $k = 1 \cdots j$ in parallel **do**
  $\mathcal{D}_k \leftarrow \left\{\boldsymbol{u}_i, \mathbf{Dist}\left(x_i, b_k^{\neg\phi}\right)\right\}_{i=1}^{n_0}$ ▷ Sample and evaluate n_0 number of inputs; Estimate the GP hyperparameters.
  $t_k \leftarrow 0$;
  while $t_k < n_{\text{BO}}$; **do**
    $\boldsymbol{u}^*_{t_k+1} \leftarrow \arg\max_{\boldsymbol{u}\in\mathbf{U}} \text{EI}\,(\boldsymbol{u})$ ▷ Select the next location
    $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \{\boldsymbol{u}^*_{t_k+1}, \mathbf{Dist}\left(x^*_{t_k+1}, b_k^{\neg\phi}\right)\}$ ▷ Update the dataset
    if $\mathbf{Dist}\left(x^*_{EI}, b_k^{\neg\phi}\right) \geq 0$ **then**
      Go to Line 17 ▷ Terminate search for all boxes
    else
      Estimate the GP hyperparameters using updated dataset $\mathcal{D}_k$.
      $t_k \leftarrow t_k + 1$
    end if
  end while
end for
return $\mathbf{u}^* \in \mathbf{U}$, $\mathbf{Dist}\left(x^*, b_k^{\neg\phi}\right)$

3.1.2 Serialized Multi-Robustness Falsification (SMuRF). When parallelization is not feasible, we can still leverage the distances to each of the boxes in the union $\mathcal{B}^{\neg\phi}$. In fact, we observe that the optimization in Eq. 4 can be formulated as:

$$\boldsymbol{u}^* = \arg\max_{\boldsymbol{u}\in\mathbf{U}} \left[\max_{k=1,\ldots,j} \mathbf{Dist}\left(x, b_k^{\neg\phi}\right)\right] \tag{6}$$

The MinBO algorithm, proposed in [34], provides an efficient approach to find approximations to Eq. 6 by utilizing a novel Expected Improvement (EI) criterion, denoted by EI_{MinBO}. Unlike traditional EI, which can only consider the maximum distance across the boxes, EI_{MinBO} can consider the distance to each box separately.

SMuRF is initialized by sampling a set of n_0 input signals $\{\boldsymbol{u_i}\}$, resulting in n_0 trajectories $\{x_i\}$. The distances to box $\mathbf{Dist}\left(x_i, b_k^{\neg\phi}\right)$, $k = 1, \cdots, j$ are computed to generate the initial SMuRF filtration dataset for $\mathcal{D}_k = \left\{\boldsymbol{u}_i, \mathbf{Dist}\left(x_i, b_k^{\neg\phi}\right)\right\}_{i=1}^{n_0}$. Like PROF, if $\mathbf{Dist}\left(x_i, b_k^{\neg\phi}\right)$ is non-positive for all i, SMuRF initializes the MinBO sampling phase. A Gaussian process (GP) is estimated using the filtration $\mathcal{D}_k$ as training set [49]. They key difference in MinBO is in the computation of the EI_{MinBO} acquisition function, which is computed as follows:

$$
\begin{aligned}
u^*_{iter+1} \in \arg\max_u EI_k(u), \quad k = 1, \cdots, j, \text{ where} \\
EI_k(u) \\
= E\left[\max\left((y^* - \hat{y}_k(\boldsymbol{u}))\Phi\left(\frac{y^* - \hat{y}_k(\boldsymbol{x})}{\hat{s}_k(\boldsymbol{u})}\right) + \right.\right. \\
\left.\left. + \hat{s}_k(\boldsymbol{u})\phi\left(\frac{y^* - \hat{y}_k(\boldsymbol{u})}{\hat{s}_k(\boldsymbol{u})}\right), 0\right)\right],
\end{aligned} \tag{7}
$$

where, $\hat{y}_k(u)$ is the prediction for the signed distance produced by the GP surrogate at location u, $\hat{s}_k(u)$ is the associated model variance, like in eq. (5). However, unlike eq. (5), y^* is the best distance value across all $\mathcal{D}_k$, $k = 1, \ldots, j$ [34]. Finally ϕ, and Φ refer here to the pdf and cdf of the standard normal distribution.

The newly obtained $\boldsymbol{u}^*_{iter+1}$ is then simulated to produce the associated signal x^*_{iter+1}. The maximum of distance to each of the boxes

$$
\max_{k=1,\ldots,j}\left(\textbf{Dist}\left(x^*_{iter+1}, b_k^{\neg\phi}\right)\right)
$$

is then computed. This results in the box-filtration $\mathcal{D}_k$ update. Unless the termination condition is met, the algorithm proceeds with the MinBO sampling stage.

Like in PROF, at any iteration, if $\textbf{Dist}\left(x^*_{iter+1}, b_k^{\neg\phi}\right) > 0$ for any box $k = 1, \ldots, j$, SMuRF terminates, otherwise it terminates when all boxes have sampled a maximum of n_{MinBO} inputs (user defined). The MinBO algorithm is shown in Alg. 2. It should be noted, as before, that the choice of the optimizer to solve eq. 6 can be replaced with any other algorithm.

Algorithm 2 Serialized Multi-Robustness Falsification (SMuRF)

Input: Search Space $\mathbf{U}$, distance $\textbf{Dist}\left(x, b_k^{\neg\phi}\right)$, initialization budget n_0, total number of MinBO iterations n_{MinBO};
Output: location and distance value $\mathbf{u}^* \in \mathbf{U}$, $\textbf{Dist}\left(x^*, b_k^{\neg\phi}\right) \geq 0$;

$\mathcal{D}_k \leftarrow \left\{\boldsymbol{u}_i, \textbf{Dist}\left(x_i, b_k^{\neg\phi}\right)\right\}_{i=1}^{n_0} \forall k \in 1, \ldots, j$ ▷ Sample and evaluate n_0 inputs.
$t_{\text{MinBO}} \leftarrow 0$;
while $t_{\text{MinBO}} < n_{\text{MinBO}}$; **do**
  for $k = 1, \ldots, j$ **do**
    Estimate the GP hyperparameters using $\mathcal{D}_k$
    $\mathbf{u}^k \leftarrow \arg\max_{\mathbf{u}\in\mathbf{U}} \text{EI}_k(\mathbf{u})$ ▷ Compute the next location for box k
  end for
  $\mathbf{u}^* \leftarrow \arg\max_{k=1,\ldots,j} EI_k\left(\mathbf{u}^k\right)$
  $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \left\{\mathbf{u}^*, \textbf{Dist}\left(x^*, b_k^{\neg\phi}\right)\right\}_{i=1}^{n_0} \forall k \in 1, \ldots, j$
  if $\max_{k=1,\ldots,j}\left(\textbf{Dist}\left(x^*, b_k^{\neg\phi}\right)\right) \geq 0$ **then**
    return $\mathbf{u}^*, \textbf{Dist}\left(x^*, b_k^{\neg\phi}\right)$ ▷ Return the sampled, point and its evaluation
  else
    $t_{\text{MinBO}} \leftarrow t_{\text{MinBO}} + 1$
  end if
end while

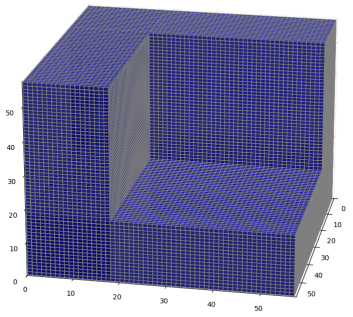

Figure 2: 3D projection of ϕ_5 in Eq. (12).

3.2 Experimental Results

To demonstrate the efficacy of the proposed approach for Cyber-Physical Systems (CPS), we use the Chasing Cars (CC) benchmark from the friendly ARCH competition [29]. The CC benchmark simulates an automatic chasing car system comprising five cars. The first car is controlled by two inputs: throttle and brake, which determine its movement. The remaining four cars follow the first car using an algorithm proposed by [26]. The system's output consists of the locations of all five cars, denoted as y_1, y_2, y_3, y_4, y_5. Both throttle and brake signals have a range of $[0, 1]$, representing the percentage of each applied. These signals are parameterized by 10 control points each, resulting in a total of 20 control points defining the system's input. The time horizon for the signal is 100 seconds, with a sampling frequency of 0.1 Hz, corresponding to 10 time steps. The benchmark aims to falsify the following properties, which constrain the longitudinal distance between the fifth and fourth cars:

$$\phi_1 = \square_{[0,100]}(y_5 - y_4 \leq 40) \tag{8}$$

$$\phi_2 = \square_{[0,70]}\left(\diamondsuit_{[0,30]}(y_5 - y_4 \geq 10)\right) \tag{9}$$

$$\phi_3 = \square_{[0,80]}\left(\left(\square_{[0,20]}(y_5 - y_4 \leq 30)\right) \vee \vee \left(\diamondsuit_{[0,20]}(y_5 - y_4 \geq 35)\right)\right) \tag{10}$$

$$\phi_4 = \square_{[0,80]}\left(\left(\square_{[0,20]}(y_5 - y_4 \leq 32)\right) \vee \vee \left(\diamondsuit_{[0,20]}(y_5 - y_4 \geq 33)\right)\right) \tag{11}$$

$$\phi_5 = \square_{[0,65]}\left(\diamondsuit_{[0,30]}\left(\square_{[0,5]}(y_5 - y_4 \geq 10)\right)\right) \tag{12}$$

Therefore, the actual signal is $x := y_5 - y_4$, and the language is a subset of $\mathfrak{R}^{10}$. Figure 2 illustrates the language of ϕ_5 in Eq. (12), by projecting it onto 3D subspaces. For generating the language, we used the code from [5], which they made available at https://github.com/Chris-amV/FindBox.git. For falsification, we use PyTaLiRo [51], the python counterpart of S-TaLiRo [7], because of its support for developing new monitors and optimization algorithms.

Monitoring accuracy. The first set of experiments answers the question: how accurate is robustness calculation using the language,

relative to robustness calculation with the state-of-the-art RTAMT? This needs to be verified since [5] *approximates* the language and does not provide a theoretical guarantee of approximation. Therefore $\mathbf{Dist}(x, \mathcal{L})$, where $\mathcal{L}$ is the approximate language, might be a poor estimate of the true robustness. In what follows we refer to $\mathbf{Dist}(x, \mathcal{L})$ as the *language monitor*.

Table 1 demonstrates the accuracy of the language monitor. A total of 10,000 input signals were generated using uniform sampling and simulated with the CC model. The *Value Agreement* row refers to the percentage of output trajectories where both RTAMT and the language monitor produced the same robustness value, and the *Sign-Only Agreement* row refers to those where only the signs agreed (so both counted a signal as being a falsifier or satisfier). The *Complete Disagreement* counts the remaining trajectories. Finally, we show the mean monitoring times for both methods. The two key takeaways from Table 1 are that (1) while the language monitor is sometimes inaccurate, in all but one case, it gives the correct robustness sign, justifying its use in falsification. To guarantee the results, when doing falsification, the *final* trajectory should be checked with RTAMT (or a similar monitor) before accepting it; and 2) the language monitor is up to two orders of magnitude faster than RTAMT, which can significantly reduce total falsification time when many tests are executed.

In Table 1, we also show the percentage of trajectories that are inside the language (*Satisfaction Ratios* row); this estimates the volume in the language in the signal space, and is an indirect measure of the difficulty of falsification. We also show the number of boxes for each language in the *Number of Boxes* row.

Table 1: Monitoring results showing satisfaction ratios, violation types, and computation times for different specifications. Mean RTAMT and monitoring times indicate performance efficiency across 10,000 simulated input signals

	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5
Value Agreement (%)	100.00	42.60	100.00	42.61	100.00
Sign-Only Agreement (%)	0.00	57.40	0.00	57.38	0.00
Complete Disagreement (%)	0.00	0.00	0.00	0.01	0.00
Mean RTAMT Time (ms)	8.69E-01	6.72E-01	1.38	1.37	1.05
Mean language monitor Time (ms)	3.64E-02	1.51E-02	3.51E-01	1.33E-02	3.68E-02
Satisfaction Ratios (%)	7.08	1.82	3.84	0.87	7.08
Number of Boxes	7	3	85	2	7

Falsification performance. The second set of experiments answers the question: how effective is using the language in finding falsifications on CPS benchmarks in comparison to the RTAMT monitor? This becomes important to analyze wether the additional information provided by the computing the language of the formulas aid the falsification process or not. To answer this question, we compare the performance of the proposed algorithms, PROF and SMuRF, against their RTAMT counterparts on falsification of properties ϕ_1 (Eq. 8) through ϕ_5 (Eq. 12). To falsify the requirements, we use the boxes computed in the earlier experiment.

The PROF algorithm is evaluated against a parallelized Bayesian Optimization (BO) procedure that leverages RTAMT to compute robustness values. Specifically, if a formula $\neg\phi$ results in j generated boxes, the parallelized BO runs on j processors. This approach is referred to as PBO-RTAMT. In contrast, SMuRF is compared to a non-parallelized BO procedure, termed SBO-RTAMT. Both algorithms are initialized with the same set of five starting points and terminate either upon finding a falsifier for ϕ or after 200 iterations. Additionally, any encountered falsifier is validated using RTAMT to confirm its correctness before termination.

The results in Table 2 and Table 3 present the outcome of falsification for PROF and PBO-RTAMT, and SMuRF and SBO-RTAMT, respectively, for requirements ϕ_1 through ϕ_5 over 20 independent trials. The FR column represents the Falsification Rate, which is the number of trials in which an algorithm successfully achieved falsification, and $\overline{S}$ denotes the mean number of iterations an algorithm takes to find a falsification. Additionally, we report the timing statistics: the $\overline{T_T}$ column shows the average amount of time (in ms) spent in monitoring across the various trials, and $\overline{T_M}$ represents the mean time (in ms) spent monitoring a single component. A single component refers to computing the distance of a signal from the box in the context of PROF and SMuRF, while it refers to monitoring the formula for the algorithms using RTAMT. In the

Table 2: Falsification results for PROF and PBO-RTAMT over 20 independent trials. FR denotes the falsification rate, $\overline{S}$ is the average number of iterations, $\overline{T_T}$ represents the total time taken, and $\overline{T_M}$ is the mean time per iteration.

Req.	Method	FR	$\overline{S}$	$\overline{T_T}$	$\overline{T_M}$
ϕ_1	PBO-RTAMT	20/20	12.50	13.617	1.107
	PROF	20/20	11.30	0.355	0.034
ϕ_2	PBO-RTAMT	20/20	7.40	12.484	1.675
	PROF	20/20	7.50	0.484	0.067
ϕ_3	PBO-RTAMT	20/20	6.30	11.785	1.877
	PROF	20/20	6.05	2.294	0.380
ϕ_4	PBO-RTAMT	20/20	24.40	44.700	1.820
	PROF	20/20	31.10	0.672	0.031
ϕ_5	PBO-RTAMT	20/20	7.40	11.242	1.540
	PROF	20/20	7.90	0.468	0.063

case of PROF (Table 2), we observe comparable performance to PBO-RTAMT in terms of both the falsification rate and the mean number of iterations. However, PROF outperforms PBO-RTAMT in monitoring time by at least two orders of magnitude. This effect is even more pronounced with SMuRF. From Table 3, it can be seen that SMuRF not only surpasses its RTAMT counterpart, SBO-RTAMT, in monitoring time but also consistently requires fewer iterations to achieve a falsification. The number of iterations is directly proportional to the time spent on system simulations, and CPS simulations are notoriously costly. Given the high computational expense of simulations, the comparable performance of PROF with PBO-RTAMT and the superior performance of SMuRF with SBO-RTAMT in terms of mean number of iterations are crucial for reducing the total number of costly simulations required. Clearly, when considering both simulation and monitoring costs, the additional information obtained from computing the formula's language contributes to significantly faster overall runtimes.

Table 3: Falsification results for SMuRF and SBO-RTAMT over 20 independent trials. FR denotes the falsification rate, $\overline{S}$ is the average number of iterations, $\overline{T_T}$ represents the total time taken, and $\overline{T_M}$ is the mean time per iteration.

Req.	Method	FR	$\overline{S}$	$\overline{T_T}$	$\overline{T_M}$
ϕ_1	SBO-RTAMT	20/20	27.90	30.438	1.207
	SMuRF	20/20	16.85	2.789	0.203
ϕ_2	SBO-RTAMT	20/20	15.10	19.245	1.272
	SMuRF	20/20	12.95	3.055	0.324
ϕ_3	SBO-RTAMT	20/20	19.45	38.564	1.975
	SMuRF	20/20	18.65	11.150	0.631
ϕ_4	SBO-RTAMT	20/20	39.10	70.265	1.808
	SMuRF	20/20	25.50	4.121	0.186
ϕ_5	SBO-RTAMT	20/20	13.40	23.811	1.701
	SMuRF	20/20	11.65	3.254	0.353

For a given formula ϕ, computing the language of $\neg\phi$ yields a union of boxes, as shown in eq. (3). To determine whether a falsification resides in multiple boxes, we conduct an additional experiment with a modified PROF algorithm: this modified version exits only if a falsifier is found in all boxes or if the maximum iteration limit of 200 is reached. In Table 4, $\overline{S}_{\min}$ and $\overline{S}_{\max}$ refer to the mean number of minimum and maximum iterations over 20 independent trials computed using the modified PROF.

Additionally, for the set of boxes associated with the formula $\neg\phi$, we implement a sequential BO procedure. This procedure starts with a randomly selected box and samples signals within the box for up to 200 iterations before switching to another box. The process continues until a falsifier is found or all boxes have been searched, at which point the algorithm terminates. Results from this approach are summarized in Table 4.

An interesting observation is that some boxes within this union are relatively easy to falsify, while others are more challenging. For instance, in formula ϕ_4, which contains 85 boxes, we noted that 2 out of 85 boxes never yielded a falsification. Specifically, the ratios $\frac{\nu(b^{\neg\phi_4}k)}{\sum k=1,\ldots,85\,\nu(b_k^{\neg\phi_4})}$, where $\nu(\cdot)$ denotes volume, were 0 and 0.00129, respectively. This suggests that the order in which boxes are traversed (on a single processor) impacts the results, motivating future work to understand what makes certain boxes harder to falsify.

4 TRACKING CONTROL WITH THE LANGUAGE

In this section we demonstrate a way to use the explicit representation of the language as a union of boxes to perform control. We tackle the control problem in the common setup of a system that consists of a motion planner in line with a tracking controller. The planner produces a *reference trajectory* x_{ref} that (hopefully) satisfies the DT-STL formula, and the tracking controller attempts to track it – i.e. produce a trajectory that is as close to x_{ref} as possible.

Table 4: Summary of experimental results from random box sampling for the falsification on a single thread using BO over 20 independent trials. Here, $\overline{S}_{\min}$ and $\overline{S}_{\max}$ indicate the mean number of minimum and maximum iterations required for falsification. Results show variability in difficulty across boxes, with some boxes being significantly harder to falsify.

Req.	FR	$\overline{S}_{\min}$	$\overline{S}_{\max}$	$\overline{S}$	$\overline{T_T}$	$\overline{T_M}$
ϕ_1	20/20	11.3	47.7	31.8	5.009	1.670
ϕ_2	20/20	7.5	37.1	19.57	4.831	0.690
ϕ_3	20/20	6.05	200	46.35	24.925	0.293
ϕ_4	20/20	31.1	63.4	25.50	4.121	0.186
ϕ_5	20/20	7.9	37.7	19.7	5.023	0.718

Current motion planners optimize some quality measure to produce a reference x_{ref}. For DT-STL objectives, that is invariably the robustness, i.e. the planner solves the following to local optimality:

$$\mathbf{u}^* = \arg\max_{\mathbf{u}\in\mathbf{U}} \rho_\phi(\mathcal{M}_s(\mathbf{u})) \quad \text{and} \quad x_{ref} = \mathcal{M}_s(\mathbf{u}^*). \tag{13}$$

If $\rho_\phi(x_{ref}) > 0$ then this is a valid reference trajectory. The model $\mathcal{M}_s$ is a *simplified* model of the dynamics for which the above optimization can be solved. Unlike falsification, the speed at which this optimization is solved is critical since its output is used in the control loop, typically in a receding horizon fashion [38, 41].

4.1 Technical Approach

Our approach is inspired by projected gradient descent in constrained optimization. Suppose that a satisfying reference $x_{ref} \in \mathcal{L}$ is identified, but the control system fails to track it accurately enough: the system's attempt at tracking x_{ref} produces trajectory x_1 which violates ϕ. We project x_1 onto the language set, yielding $x_1^p \in \mathcal{L}(\phi)$. *This is the satisfying signal nearest to this actual system output trajectory.* This projection is now the new reference, which the system must track, yielding x_2: if x_2 is in the language, we are done; otherwise, we again project the new trajectory, and repeat.

We first note that *such projection is not possible at all if we only have the formula or its robustness.* Moreover, we can trivially parallelize this approach, by projecting onto each box in $\mathcal{L}(\phi)$ in parallel: from a control perspective, this gives a very easy way of exploring all the ways in which the formula might be satisfied, and allows us to pick among them based on secondary criteria (like minimal energy or jerk).

The pseudocode is shown as Algorithm 3. It takes in a union of boxes B (which can be the whole language, or a single box in case we parallelize over available processors) and performs the above project-then-track iterations.

4.2 Dynamical System and Its Objectives

In our experiments, we focused on tracking the x-position of an Uncrewed Aerial Vechile (UAV) using a dynamical model that incorporates translational, rotational, and kinematic equations [47]
.

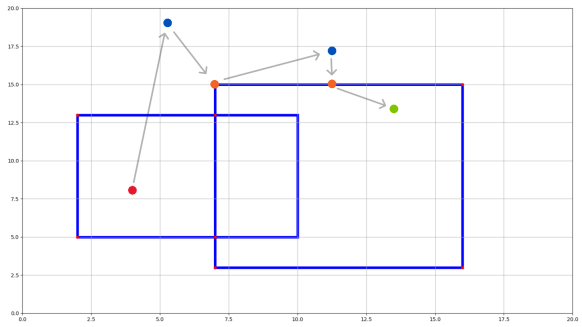

Figure 3: Representation of a run of Algorithm 3 in a 2-box language. Start point/trajectory is red, tracking points are blue, and projected points are orange. The final feasible point, which is successfully tracked, is green.

Algorithm 3 LangControl: Control with the Whole Language

Input: Union of boxes B (representing the language $\mathcal{L}(\phi)$ or a subset of it), signal length N, Failure limits M and K, control system tracker;
Output: Trajectory satisfying the formula and corresponding control signal

$m \leftarrow 0$
while $m \leq M$ **do**
  ▷ Sample a random signal from the constraint set without replacement
  $x_{ref} \leftarrow$ Sample(B)
  for ($count := 0; count \leq K; count++$) **do**
    ▷ Tracker returns trajectory and corresponding control signal
    $x, \mathbf{u} \leftarrow$ Track(x_{ref})
    if $x \in B$ **then**
      return $x, \mathbf{u}$ ▷ Feasible signal found
    else
      $x_{ref} \leftarrow$ Projection(x, B) ▷ Project x onto B
    end if
  end for
  ▷ If this attempt failed, start from a different initial reference trajectory
  $m \leftarrow m + 1$
end while
return $\perp$ ▷ No feasible signal found within the limits

- x_1, y_1, z_1 represent the position in the x, y, and z directions respectively, and x_2, y_2, z_2 be the corresponding velocities.
- $\phi_1, \phi_2, \theta_1, \theta_2, \psi_1, \psi_2$ represent the roll, pitch, and yaw angles and their corresponding angular velocities.
- f_t be the thrust.
- τ_x, τ_y, τ_z the the body torques.
- m the the mass of the quadrotor.
- I_x, I_y, I_z the the moments of inertia.
- g the the acceleration due to gravity.

The kinematic equations of motion are simply

$$\dot{x}_1 = x_2, \quad \dot{y}_1 = y_2, \quad \dot{z}_1 = z_2,$$
$$\dot{\phi}_1 = \phi_2, \quad \dot{\theta}_1 = \theta_2, \quad \dot{\psi}_1 = \psi_2$$

The translational dynamics are given by

$$\dot{x}_2 = \frac{f_t}{m}\left(\sin(\phi_1)\sin(\psi_1) + \cos(\phi_1)\cos(\psi_1)\sin(\theta_1)\right)$$
$$\dot{y}_2 = \frac{f_t}{m}\left(\cos(\phi_1)\sin(\psi_1)\sin(\theta_1) - \cos(\psi_1)\sin(\phi_1)\right)$$
$$\dot{z}_2 = -g + \frac{f_t}{m}\cos(\phi_1)\cos(\theta_1)$$

The rotational dynamics are given by

$$\dot{\phi}_2 = \frac{I_x}{I_y - I_z}\theta_2\psi_2 + \frac{\tau_x}{I_x}$$
$$\dot{\theta}_2 = \frac{I_y}{I_z - I_x}\phi_2\psi_2 + \frac{\tau_y}{I_y}$$
$$\dot{\psi}_2 = \frac{I_z}{I_x - I_y}\phi_2\theta_2 + \frac{\tau_z}{I_z}$$

For trajectories of length 10, we used the following DT-STL objectives:

$$\phi_1 = \Box(0 \leq x_1 \leq 10) \wedge \Box\neg(3 < x_1 < 5) \tag{14}$$
$$\phi_2 = \Diamond_{[0,7]}(0 \leq x_1 \leq 3) \wedge \Box_{[8,10]}(0 \leq x_1 \leq 3) \tag{15}$$
$$\phi_3 = \Diamond_{[0,10]}(0 \leq x_1 \leq 3) \wedge \Diamond_{[0,10]}(7 \leq x_1 \leq 10) \tag{16}$$
$$\phi_4 = \Diamond_{[0,2]}(0 \leq x_1 \leq 2) \wedge \Diamond_{[3,5]}(4 \leq x_1 \leq 6) \wedge \Diamond_{[6,9]}(15 \leq x_1 \leq 17) \tag{17}$$

For trajectories of length 30, we used the following objectives

$$\phi_5 = \Box(0 \leq x_1 \leq 100) \wedge \Box\neg(30 < x_1 < 50) \tag{18}$$
$$\phi_6 = \Diamond_{[0,20]}(40 \leq x_1 \leq 45) \wedge \Box_{[20,29]}(40 \leq x_1 \leq 45) \tag{19}$$
$$\phi_7 = \Diamond_{[0,29]}(30 \leq x_1 \leq 35) \wedge \Diamond_{[0,29]}(75 \leq x_1 \leq 80) \tag{20}$$
$$\phi_8 = \Diamond_{[0,9]}(x_1 \geq 0 \wedge x_1 \leq 5) \wedge \Diamond_{[10,19]}(x_1 \geq 95 \wedge x_1 \leq 100) \wedge \Diamond_{[20,29]}(x_1 \geq 0 \wedge x_1 \leq 5) \tag{21}$$

Figure 4 illustrates the languages of two of these formulas, by projecting them onto 3D subspaces.

4.3 Experimental Results

The results are shown in Table 5. Two variations of LangControl were tested: one where $B = \mathcal{L}(\phi)$ (so there's no parallelization, and each reference is projected onto the entire language), and one where B is a single box and the different boxes are distributed over the available processors for trivially parallel execution. We call these LCW and LCP, respectively. We note that in LCP, each tracking result is checked for whether it entered *any* box, thus ensuring that our method doesn't miss successful controls.

For a given formula, we run both algorithms LCW and LCP 1000 times, changing the initial reference signal x_{ref} between replications. We report the average number of iterations (averaged across replications).

We first note that both variations of LangControl perform a remarkably small number of iterations, compared to classical robustness-based methods like [41], which invoke a general-purpose global

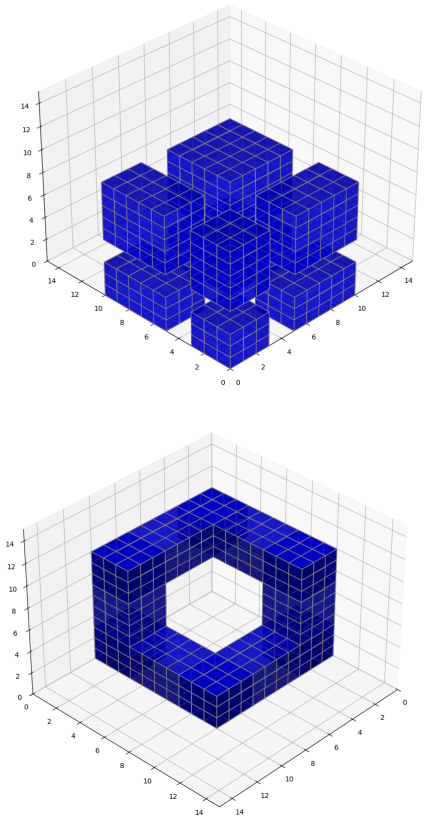

Figure 4: 3D projection of ϕ_1 in Eq. (14) (top) and ϕ_3 in Eq. (16) (bottom)

Table 5: Performance of LangControl variations LCW and LCP. $\overline{P}$ is the average number of iterations needed until reaching a feasible signal.

Formula	ϕ_1	ϕ_2	ϕ_3	ϕ_4
$\overline{P}$ of LCW	1.88423	4.80769	1.25672	135.036
$\overline{P}$ of LCP	1	2.32152	1	41.2362

Formula	ϕ_5	ϕ_6	ϕ_7	ϕ_8
$\overline{P}$ of LCW	4.27235	6.31527	206.935	1965
$\overline{P}$ of LCP	2.37237	2.94731	89.7326	488.345

optimizer like IPOPT to maximize robustness. We perform preliminary experiments using RTAMT to compute robustness values and employing BO as the sampling strategy to maximize robustness. Compared to LCW, we observe that BO consistently requires 3-7x the number of iterations. The second thing to note is that LCP is consistently better than LCW. This is not evident, because even though LCP parallelizes, the individual processes work with one box. The results suggest that focusing on one box provides significant speedups, from 2x to 4x.

5 CONCLUSION

This paper provided ways to perform falsification and control by taking advantage of a recent method to approximate the language of a DT-STL formula. Our results indicate significant speedups can be obtained with relatively simple approaches, thus offsetting the cost of generating the language in the first place. They also point to interesting questions worth pursuing.

The first is to leverage the work in [5] and extend the methodology to accommodate multidimensional signals, thus ensuring the applicability of this work to general scenarios. The second is to come up with a metric to *a priori* quantify the likelihood of a box containing a falsifier. This metric not only guides the optimal order for attempting the boxes in a single-process falsification run but also aids in selecting the most probable subset of boxes, speeding up the parallelized algorithms when the number of boxes is large. The same analysis could be run when trying to control the system to satisfy the requirement. Related to that, if control attempts indicate that only a small subset of boxes is reachable by the system, this might indicate a need to tighten the formula which is overly lax – or, on the contrary, a need to actually debug the system which is only able to satisfy the requirement in very specific ways. Finally, it will be important to understand the performance of LangControl when working with one box, perhaps by formalizing it as a projected gradient optimization. These, and other more theoretical questions, should provide a sound theoretical basis for our results.

REFERENCES

[1] Houssam Abbas and Georgios Fainekos. 2012. Convergence proofs for Simulated Annealing falsification of safety properties. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 1594–1601. https://doi.org/10.1109/Allerton.2012.6483411

[2] Houssam Abbas, Georgios Fainekos, Sriram Sankaranarayanan, Franjo Ivančić, and Aarti Gupta. 2013. Probabilistic Temporal Logic Falsification of Cyber-Physical Systems. *ACM Trans. Embed. Comput. Syst.* 12, 2s, Article 95 (may 2013), 30 pages. https://doi.org/10.1145/2465787.2465797

[3] H. Abbas and R. Mangharam. 2018. Generalized Robust MTL Semantics for Problems in Cardiac Electrophysiology. In *2018 Annual American Control Conference (ACC)*. 1592–1597. https://doi.org/10.23919/ACC.2018.8431460

[4] Christian Abou-Mrad. 2024. Private Communication.

[5] Christian Abou-Mrad and Houssam Abbas. 2024. Approximating the Geometry of Temporal Logic Formulas. In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control* (Hong Kong SAR, China) *(HSCC '24)*. Association for Computing Machinery, New York, NY, USA, Article 16, 10 pages. https://doi.org/10.1145/3641513.3650139

[6] Étienne André, Masaki Waga, Natuski Urabe, and Ichiro Hasuo. 2022. Exemplifying Parametric Timed Specifications over Signals with Bounded Behavior. In *NASA Formal Methods*, Jyotirmoy V. Deshmukh, Klaus Havelund, and Ivan Perez (Eds.). Springer International Publishing, Cham, 470–488.

[7] Yashwanth Singh Rahul Annapureddy, Che Liu, Georgios E. Fainekos, and Sriram Sankaranarayanan. 2011. S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems. In *Tools and algorithms for the construction and analysis of systems (LNCS, Vol. 6605)*. Springer, 254–257.

[8] Ezio Bartocci, Luca Bortolussi, Michele Loreti, and Laura Nenzi. 2017. Monitoring mobile and spatially distributed cyber-physical systems. In *Proceedings of ACM MEMOCODE conference*. Association for Computing Machinery, New York, NY, USA, 146–155. https://doi.org/10.1145/3127041.3127050

[9] Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donzé, Georgios Fainekos, Oded Maler, Dejan Ničković, and Sriram Sankaranarayanan. 2018. *Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications*. Springer International Publishing, Cham, 135–175. https://doi.org/10.1007/978-3-319-75632-5_5

[10] Nicolas Basset, Thao Dang, Felix Gigler, Cristinel Mateis, and Dejan Ničković. 2021. Sampling of shape expressions with ShapEx. In *Proceedings of the 19th ACM-IEEE International Conference on Formal Methods and Models for System Design* (Virtual Event, China) *(MEMOCODE '21)*. Association for Computing Machinery, New York, NY, USA, 118–125. https://doi.org/10.1145/3487212.3487350

[11] Anthony Corso, Robert J. Moss, Mark Koren, Ritchie Lee, and Mykel J. Kochenderfer. 2020. A Survey of Algorithms for Black-Box Safety Validation. *ArXiv* abs/2005.02979 (2020). https://api.semanticscholar.org/CorpusID:218516698

[12] CPS Virtual Organization. 2024. ARCH Workshop Series.

[13] Joseph Cralley, Ourania Spantidi, Bardh Hoxha, and Georgios Fainekos. 2020. Tltk: A toolbox for parallel robustness computation of temporal logic specifications. In *Runtime Verification: 20th International Conference, RV 2020, Los Angeles, CA, USA, October 6–9, 2020, Proceedings 20*. Springer, 404–416.

[14] Jyotirmoy Deshmukh, Marko Horvat, Xiaoqing Jin, Rupak Majumdar, and Vinayak S. Prabhu. 2017. Testing Cyber-Physical Systems through Bayesian Optimization. *ACM Trans. Embed. Comput. Syst.* 16, 5s, Article 170 (sep 2017), 18 pages. https://doi.org/10.1145/3126521

[15] A. Dokhanchi, B. Hoxha, and G. Fainekos. 2014. Online Monitoring for Temporal Logic Robustness. In *Proc. of Runtime Verification*. Springer.

[16] Alexandre Donzé. 2010. Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems. In *Computer Aided Verification*, Tayssir Touili, Byron Cook, and Paul Jackson (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 167–170.

[17] Alexandre Donzé, Thomas Ferrere, and Oded Maler. 2013. Efficient robust monitoring for STL. In *Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25*. Springer, 264–279.

[18] Alexandre Donzé and Oded Maler. 2010. Robust satisfaction of temporal logic over real-valued signals. In *Proceedings of the 8th International Conference on Formal Modeling and Analysis of Timed Systems* (Klosterneuburg, Austria) *(FORMATS'10)*. Springer-Verlag, Berlin, Heidelberg, 92–106.

[19] T. Dreossi, T. Dang, A. Donze, J. Kapinski, X. Jin, and J. V. Deshmukh. 2015. A Trajectory Splicing Approach to Concretizing Counterexamples for Hybrid Systems. In *NASA Symposium on Formal Methods*.

[20] G. Fainekos and G. Pappas. 2009. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science* (2009).

[21] Peter I Frazier. 2018. Bayesian optimization. In *Recent advances in optimization and modeling of contemporary problems*. Informs, 255–278.

[22] Nicole Fronda and Houssam Abbas. 2022. Differentiable Inference of Temporal Logic Formulas. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 11 (2022), 4193–4204. https://doi.org/10.1109/TCAD.2022.3197506

[23] Shromona Ghosh, Dorsa Sadigh, Pierluigi Nuzzo, Vasumathi Raman, Alexandre Donzé, Alberto L. Sangiovanni-Vincentelli, S. Shankar Sastry, and Sanjit A. Seshia. 2016. Diagnosis and Repair for Synthesis from Signal Temporal Logic Specifications. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control* (Vienna, Austria) *(HSCC '16)*. Association for Computing Machinery, New York, NY, USA, 31–40. https://doi.org/10.1145/2883817.2883847

[24] Bardh Hoxha, Hoang Bach, Houssam Abbas, Adel Dokhanchi, Yoshihiro Kobayashi, and Georgios Fainekos. 2014. Towards formal specification visualization for testing and monitoring of cyber-physical systems. In *Int. Workshop on Design and Implementation of Formal Tools and Systems*.

[25] Bardh Hoxha, Nikolaos Mavridis, and Georgios Fainekos. 2015. VISPEC: A graphical tool for elicitation of MTL requirements. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 3486–3492. https://doi.org/10.1109/IROS.2015.7353863

[26] Jianghai Hu, John Lygeros, and Shankar Sastry. 2000. Towards a theory of stochastic hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 160–173.

[27] Donald R. Jones, Matthias Schonlau, and William J. Welch. 1998. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization* 13 (1998), 455–492. https://api.semanticscholar.org/CorpusID:263864014

[28] James Kapinski, Jyotirmoy V. Deshmukh, Xiaoqing Jin, Hisahiro Ito, and Kenneth R. Butts. 2016. Simulation-Based Approaches for Verification of Embedded Control Systems: An Overview of Traditional and Advanced Modeling, Testing, and Verification Techniques. *IEEE Control Systems* 36 (2016), 45–64. https://api.semanticscholar.org/CorpusID:34303066

[29] Tanmay Khandait, Federico Formica, Paolo Arcaini, Surdeep Chotaliya, Georgios Fainekos, Abdelrahman Hekal, Atanu Kundu, Ethan Lew, Michele Loreti, Claudio Menghi, et al. 2024. ARCH-COMP 2024 Category Report: Falsification. In *Proceedings of the 11th Int. Workshop on Applied*, Vol. 103. 122–144.

[30] Zhaodan Kong, Austin Jones, Ana Medina Ayala, Ebru Aydin Gol, and Calin Belta. 2014. Temporal Logic Inference for Classification and Prediction from Data. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control* (Berlin, Germany) *(HSCC '14)*. Association for Computing Machinery, New York, NY, USA, 273–282. https://doi.org/10.1145/2562059.2562146

[31] R. Koymans. 1990. Specifying Real-Time Properties with Metric Temporal Logic. *Real-Time Systems* 2, 4 (1990), 255–299.

[32] Lars Lindemann and Dimos Dimarogonas. 2025. *Formal Methods for Multi-Agent Feedback Control Systems*. MIT Press.

[33] Oded Maler and Dejan Nickovic. 2004. *Monitoring Temporal Properties of Continuous Signals*. Springer Berlin Heidelberg.

[34] Logan Mathesen, Giulia Pedrielli, and Georgios Fainekos. 2021. Efficient Optimization-Based Falsification of Cyber-Physical Systems with Multiple Conjunctive Requirements. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)* (Lyon, France). IEEE Press, 732–737. https://doi.org/10.1109/CASE49439.2021.9551474

[35] Noushin Mehdipour, Cristian Ioan Vasile, and Calin Belta. 2019. Average-based Robustness for Continuous-Time Signal Temporal Logic. In *58th IEEE Conference on Decision and Control, CDC 2019, Nice, France, December 11-13, 2019*. IEEE, 5312–5317. https://doi.org/10.1109/CDC40024.2019.9029989

[36] Tarik Nahhal and Thao Dang. 2007. Test Coverage for Continuous and Hybrid Systems. In *CAV (LNCS, Vol. 4590)*. Springer, 449–462.

[37] D. Ničković and Tomoya Yamaguchi. 2020. RTAMT: Online Robustness Monitors from STL. In *Automated Technology for Verification and Analysis*. https://api.semanticscholar.org/CorpusID:218869539

[38] Yash Vardhan Pant, Houssam Abbas, and Rahul Mangharam. 2017. Smooth operator: Control using the smooth robustness of temporal logic. In *Control Technology and Applications (CCTA), 2017 IEEE Conference on*. IEEE.

[39] Yash Vardhan Pant, Houssam Abbas, and Rahul Mangharam. 2017. Smooth Operator: Control using the Smooth Robustness of Temporal Logic. In *IEEE Conference on Control Technology and Applications,*.

[40] Yash Vardhan Pant, Houssam Abbas, Rhudii A. Quaye, and Rahul Mangharam. 2018. Fly-by-logic: Control of Multi-drone Fleets with Temporal Logic Objectives. In *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems* (Porto, Portugal) *(ICCPS '18)*. IEEE Press, Piscataway, NJ, USA, 186–197. https://doi.org/10.1109/ICCPS.2018.00026

[41] Yash Vardhan Pant, Houssam Abbas, Rhudii A Quaye, and Rahul Mangharam. 2018. Fly-by-logic: Control of multi-drone fleets with temporal logic objectives. In *Proceedings of the 9th ACM/IEEE International Conference on Cyber-Physical Systems*. IEEE Press.

[42] Giulia Pedrielli, Tanmay Khandait, Yumeng Cao, Quinn Thibeault, Hao Huang, Mauricio Castillo-Effen, and Georgios Fainekos. 2023. Part-x: A family of stochastic algorithms for search-based test generation with probabilistic guarantees. *IEEE Transactions on Automation Science and Engineering* (2023).

[43] Pavithra Prabhakar, Ratan Lal, and James Kapinski. 2018. Automatic Trace Generation for Signal Temporal Logic. In *2018 IEEE Real-Time Systems Symposium (RTSS)*. 208–217. https://doi.org/10.1109/RTSS.2018.00038

[44] V. Raman, A. Donze, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia. 2014. Model predictive control with signal temporal logic specifications. In *53rd IEEE Conference on Decision and Control*. 81–87. https://doi.org/10.1109/CDC.2014.7039363

[45] Alëna Rodionova, Yash Vardhan Pant, Connor Kurtz, Kuk Jin Jang, Houssam Abbas, and Rahul Mangharam. 2021. Learning-'N-Flying: A Learning-Based, Decentralized Mission-Aware UAS Collision Avoidance Scheme. *ACM Trans. Cyber Phys. Syst.* 5, 4 (2021), 35:1–35:26. https://doi.org/10.1145/3447624

[46] Hendrik Roehm, Thomas Heinz, and Eva Charlotte Mayer. 2017. STLInspector: STL Validation with Guarantees. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 10426)*, Rupak Majumdar and Viktor Kuncak (Eds.). Springer, 225–232. https://doi.org/10.1007/978-3-319-63387-9_11

[47] Francesco Sabatino. 2019. *Verification and Control of Temporal Logic Specifications for Hybrid Systems*. Master's thesis. KTH Royal Institute of Technology. https://www.kth.se/polopoly_fs/1.588039.1550155544!/Thesis%20KTH%20-%20Francesco%20Sabatino.pdf

[48] Sadra Sadraddini and Calin Belta. 2015. Robust Temporal Logic Model Predictive Control. In *Allerton conference*.

[49] TJ Santner. 2003. The Design and analysis of computer experiments.

[50] Simone Silvetti, Alberto Policriti, and Luca Bortolussi. 2017. An active learning approach to the falsification of black box cyber-physical systems. In *Integrated Formal Methods: 13th International Conference, IFM 2017, Turin, Italy, September 20-22, 2017, Proceedings 13*. Springer, 3–17.

[51] Quinn Thibeault, Jacob Anderson, Aniruddh Chandratre, Giulia Pedrielli, and Georgios Fainekos. 2021. PSY-TaLiRo: A Python Toolbox for Search-Based Test Generation for Cyber-Physical Systems. In *Formal Methods for Industrial Critical Systems: 26th International Conference, FMICS 2021, Paris, France, August 24–26, 2021, Proceedings* (Paris, France). Springer-Verlag, Berlin, Heidelberg, 223–231. https://doi.org/10.1007/978-3-030-85248-1_15

[52] Eric M. Wolff, Ufuk Topcu, and Richard M. Murray. 2013. Automaton-guided controller synthesis for nonlinear systems with temporal logic. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 4332–4339. https://doi.org/10.1109/IROS.2013.6696978

[53] Hengyi Yang. 2013. *Dynamic programming algorithm for computing temporal logic robustness*. Technical Report. Arizona State University.

[54] Zhenya Zhang and Paolo Arcaini. 2021. Gaussian process-based confidence estimation for hybrid system falsification. In *International Symposium on Formal Methods*. Springer, 330–348.

[55] Zhenya Zhang, Deyun Lyu, Paolo Arcaini, Lei Ma, Ichiro Hasuo, and Jianjun Zhao. 2021. Effective Hybrid System Falsification Using Monte Carlo Tree Search Guided by QB-Robustness. In *Computer Aided Verification: 33rd International Conference, CAV 2021, Virtual Event, July 20–23, 2021, Proceedings, Part I*. Springer-Verlag, Berlin, Heidelberg, 595–618. https://doi.org/10.1007/978-3-030-81685-8_29