Towards Efficient Input Space Exploration for Falsification of Input Signal Class Augmented STL

Vinayak S. Prabhu, and Meetkumar Savaliya

***Abstract*—In recent years black-box optimization based search testing for Signal Temporal Logic (STL) specifications has been shown to be a promising approach for finding bugs in complex Cyber-Physical Systems (CPS) that are out of reach of formal analysis tools. The efficacy of this approach depends on efficiently exploring the input signal space, which for CPS is infinite. In this work, we present a framework for more efficient exploration of the input space for falsification of a class of engineering requirements. Our first contribution is a dimensionality reduction heuristic for optimization based falsification frameworks for dynamical systems over this augmented logic. This heuristic leverages the step response of the system – a standard system characteristic from Control engineering – to obtain a smaller time interval in which the optimizer needs to vary the inputs. Next, we note that system behaviors on a standard class of inputs such as on step inputs or sinusoids are often of paramount importance to engineers, and such inputs while easy to specify as functions, are difficult for temporal logics to capture. Our second contribution is a formalism to augment a commonly used fragment of Signal Temporal Logic (STL) to incorporate such signals for use in a black-box optimization based falsification framework. Finally, we demonstrate the effectiveness of our approach in falsification of temporal logic specifications on three case studies over complex Simulink models.**

I. Introduction

A key part of the development of Cyber-Physical Systems (CPS) is identification of the input space for which the system behavior is not satisfactory. This includes testing the system over a test suite, and more generally *falsification* which incorporates a search for test inputs that cause undesirable behavior. Due to the complex nature of CPS models, often incorporating differential equations and look-up tables, static analysis techniques are infeasible. Fortunately, the properties engineers are interested in involve signals that take *quantitative* values, and this has opened up the field of black-box based optimization testing. For example, a property of interest in the Air Fuel Control system from [24] is the requirement that the air fuel ratio be within some band. Black-box optimization testing loops execute the system under test, compute a quantitative metric based on the property of interest that indicates how far the system is from violating stated requirements, and then depending on the values computed so far, select a next input for the next iteration of the loop. Key developments that made this approach applicable to intricate requirements, such those expressed with Signal Temporal Logic (STL) specifications, were (a) development of a *robustness function* that quantified the degree of violation of a temporal formula for a given trace; and (b) development of efficient algorithms for computing the value of this function over traces [19], [14]. A positive robustness function value implies boolean satisfaction of the STL property, and a negative value implies violation. The goal of the optimizer is to generate inputs such that the robustness value of the system output with respect to the STL property is negative. Over the past decade tools such as Breach. S-TaLiRo, FalCAun have advanced this framework for falsification of complex CPS models [32], [20], [12]. Simultaneously, efforts are underway to develop a suite of CPS benchmarks which can guide falsification tool improvement [15].

Department of Computer Science, Colorado State University (USA);
Email: vinayak.prabhu@colostate.edu,
meetkumar.savaliya@gmail.com

A basic ingredient of these approaches is the translation of the search space over input signals – defined as functions $[0, T] \rightarrow \mathbb{R}^n$ over some time domain $[0, T]$ – to a finite parameter space over which the black-box optimizers typically operate. For example, one way to do this is to pick p time points (also known as control points) over $[0, T]$, specify the n-dimensional input signal on these p time points, with the full input signal being generated by a signal generator module according to some interpolation scheme. Such an encoding results in $p \cdot n$ parameters the optimizer has to search over. The selection of p has conflicting implications for the search process: (a) a smaller p value leads to input signals that are more slowly varying and might miss input signals that lead to the system output violating the given property; (b) a larger p increases the number of parameters for the black-box optimizer to search over thus making the search difficult. Orthogonally, for debugging it is desirable to have "simpler" signals that demonstrate property violation; thus one may wish to prioritize a smaller number of control points.

Our first contribution in this work is the exploration of a dimensionality reduction heuristic to reduce the number of parameters for more efficient search of the input signal space. Our heuristic takes inspiration from the *step response* concept from the theory of Control Systems. The step response of a system, from a given initial state, is the response of the system when given a step input. Step response behavior is a critical part of the analysis of control systems [26], and overshoot, rise time, and settling time data from the step response are key characteristics for behavior analysis of dynamical systems. Our heuristic is to vary the input signal not over the entire simulation horizon, but over a time duration equal to the *settling time* for the step response. For example, if the simulation horizon is 80 seconds, and we wish to have a uniform placement of control points at 2 second intervals, then this would result in 41 control points. If the settling time is 20 seconds, then our heuristic would require 22 control points (we fix a special control point to be at the beginning of the simulation). The placement of this 20 second interval which contains the control points is itself variable, with the offset being determined by the optimizer. Such a heuristic, if successful has three benefits: (1) it reduces the search space

for the optimizer, and (2) the falsifying inputs are simpler, as they change over a smaller number of time points; and (3) the falsifying inputs change over a smaller time horizon (the settling time). Points (2) and (3) lead to inputs signals that are more desirable for debugging.

Our second contribution is a formalization and use of typical engineering request-response constraints in the black-box optimization loop. Often requirements such as "if the input signal is of type θ_{ip} then output response must satisfy θ_{op} for some time duration" are of interest, where θ_{op} involves only output signals, and θ_{ip} only input signal variables, and θ_{ip} restricts input signals to be of certain special standard mathematical function classes, For instance, θ_{ip} could be a type of pulse input, or a sine wave segment. The way this has usually been handled previously has been to encode both θ_{ip} and θ_{op} as STL sub-formulae; however the temporal logic encodings of θ_{ip}, when mathematical functions such as sinusoids or step inputs, is usually cumbersome, and prone to errors. Our insight is that such an encoding of *input constraints* is unnecessary for such input-request output-response properties due to the following fact. We are only interested in searching over input signals satisfying the constraint θ_{ip}; the robustness function, if the input constraint θ_{ip} is satisfied should depend *only* on the output response condition θ_{op}. In addition, if the input constraint is not satisfied, we are not interested in *vacuous* satisfaction of the full requirement; this has been argued previously in [21], [33]. That is, the utility of θ_{ip} is *only* in ensuring the input signals that we search over satisfy a certain constraint, it does not have a quantitative utility unlike θ_{op} (which tells us the degree of violation of the system output). This means that if we can ensure that the input constraint is satisfied for generated inputs by some mechanism, then we can omit the θ_{ip} constraint from the robustness computation routine altogether, which in turn means that the encoding of θ_{ip} into STL is also unnecessary. Now we note that this adherence of generated input signals to θ_{ip} can be easily enforced as the signal generation module in an optimization framework is *under our control*, unlike the system output which depends on the complex dynamics of the system. For instance if θ_{ip} is a sinusoid signal requirement, we simply plug in a signal generator that generates a sinusoid. Since our signal generator ensures θ_{ip} being in the input, the optimizer can focus on using the robustness function for θ_{op} to search for an input (which is guaranteed to satisfy θ_{ip} by design) which violates θ_{op}. Usually, there are temporal constraints relating θ_{op} to θ_{ip} , for instance if θ_{ip} occurs then θ_{op} is activated for b time units, *e.g.*, $\square\left(\theta_{\mathsf{ip}} \rightarrow \square_{[0,b]}\theta_{\mathsf{op}}\right)$. We show in this work strategies for inferring when θ_{op} becomes active with an appropriately designed signal generator; and how to use this knowledge to modify the black-box optimization loop. With our proposed method, we are not restricted by STL to capture these input requirements, and we present a formal treatment to enlarge STL with these more general input signal classes. This augmentation and our proposed method for the robustness computation over this augmented logic formalizes the use of more complex antecedent input triggering requirements in falsification frameworks.

Finally, we implemented our heuristic in the Breach tool and evaluated over three benchmark models from the ARCH-COMP repository [17] for our experiments. The results demonstrate the effectiveness of our proposed approach.

Related Work. The most closely related works are [10], [18], [30]. The work in [10] investigated placing control points non-uniformly in order to reduce the number of parameters for black box search. More control points were introduced during the search based on the optimizer performance and the gap between control points. FALSTAR [18] is a tool that generates input signals in segments of varying lengths. The work in [30] investigates the idea that specialized input signal generator classes such as pulse generators could be more efficient in finding falsifying inputs. The general area of falsification is an active area of research, works [3], [29], [22] investigate alternative robustness definitions for STL that aggregate robustness rather than take extremal robustness values done commonly for STL. Works [6], [7] explore sampling of input traces satisfying patterns given by timed automata. Quantitative conformance notions are explored in [8], [27], [9]. Treating input and output signals differently for computing robustness is argued in [21]. In the context of request-response specifications, [11] employs a falsification loop to generate input signal portions satisfying input STL constraints, [2] uses a probabilistic approach to improve the chances of input antecedent satisfaction in the falsification process.

II. Problem Setting

A. Background

Systems, signals, traces. (finite) *trace* or a *signal* $\pi : [0,T] \rightarrow \mathbb{R}^n$ of arity n is a mapping from a finite closed interval $[0,T]$ of $\mathbb{R}_+$ to $\mathbb{R}^n$. The time-domain of π is the time interval $[0,T]$ over which it is defined. We partition signals into *input* signals, and *output* signals. A (continuous-time) *system* $\mathcal{S} : \left(\mathbb{R}_+^{[\,]} \rightarrow \mathbb{R}^{n_I}\right) \rightarrow \left(\mathbb{R}_+^{[\,]} \rightarrow \mathbb{R}^{n_O}\right)$, where $\mathbb{R}_+^{[\,]}$ is the set of finite closed intervals $[0,T]$ of $\mathbb{R}_+$, transforms input signals $\pi_{\mathsf{ip}} : [0,T] \rightarrow \mathbb{R}^{n_I}$ into output traces $\pi_{\mathsf{op}} : [0,T] \rightarrow \mathbb{R}^{n_O}$ (over the same time domain). At times we refer to the input-trace output-trace combination as $\pi : [0,T] \rightarrow \mathbb{R}^{n_I+n_O}$, with π_{ip} and π_{op} being the corresponding projections; that is if $\pi_{\mathsf{ip}}(t) = (a_1, \ldots, a_{n_I})$ and $\pi_{\mathsf{op}}(t) = (b_1, \ldots, b_{n_O})$, then $\pi(t) = (a_1, \ldots, a_{n_I}, b_1, \ldots, b_{n_O})$. For the signal value $\pi(t) = (a_1, \ldots, a_{n_I}, b_1, \ldots, b_{n_O})$, we refer to each a_j as the value of the j-th input *signal variable*, and similarly for output signal variables. Given a trace $\pi : [0,T] \rightarrow \mathbb{R}^k$, we refer to the sub-trace between times δ_1, δ_2 where $0 \leq \delta_1 \leq \delta_2 \leq T$ as π_{δ_1,δ_2}. Formally, $\pi_{\delta_1,\delta_2} : [0, \delta_2 - \delta_1] \rightarrow \mathbb{R}^k$ is a trace of length $\delta_2 - \delta_1$ such that $\pi_{\delta_1,\delta_2}(t) = \pi(t + \delta_1)$. We refer to the j-th dimension of the signal π as π_j, that is $\pi_j : [0,T] \rightarrow \mathbb{R}$ such that $\pi_j(t) = d_j$ where $\pi(t) = (d_1, \ldots, d_k)$.

Signal Temporal Logic (STL). Signal Temporal Logic (STL), introduced in [28], extends Metric Interval Temporal Logic (MITL) [4] with real-time signals. We consider STL formulas with bounded-time temporal operators defined recursively according to the grammar $\varphi ::=$

$$\mathrm{T} \mid f(z_1, \ldots, z_n) \geq 0 \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \mathcal{U}_{[a,b]} \varphi_2,$$

where T is the *true* predicate; f is an n-ary function symbol; z_i are input/output signal variables; $\varphi, \varphi_1, \varphi_2$ are STL formulas; $\neg$ and $\wedge$ are Boolean connectives that respectively indicate negation and conjunction; and $\mathcal{U}_{[a,b]}$ with $a, b \in \mathbb{R}$ such that

$0 \leq a \leq b$ is the *until* operator. We define additional temporal operators in the standard way: the "eventually" operator $\Diamond_{[a,b]}\varphi$ stands for $\mathrm{T}\,\mathcal{U}_{[a,b]}\varphi$; and the "always" operator $\Box_{[a,b]}\varphi$ stands for $\neg\Diamond_{[a,b]}\neg\varphi$.

Definition 1 (STL Semantics)**.** Let $\pi : [0,T] \to \mathbb{R}^{n_I+n_O}$ be a signal, where $\pi(t) = (\pi_1(t), \ldots, \pi_{n_I+n_O}(t))$. The suffix of the trace from time $t \in [0,T]$, *i.e.*, $\pi_{t,T}$, is said to satisfy STL formulae as follows [14].

$(\pi,t) \models \mathrm{T}$; $\quad (\pi,t) \models \neg\varphi$ iff $(\pi,t) \not\models \varphi$;

$(\pi,t) \models f(z_1,..,z_{n_I+n_O}) \geq 0$ iff $f(\pi_1(t),..,\pi_{n_I+n_O}(t)) \geq 0$; where the variables z_j correspond to the j-th dimension of π;

$(\pi,t) \models \varphi_1 \wedge \varphi_2$ iff $(\pi,t) \models \varphi_1$ and $(\pi,t) \models \varphi_1$;

$(\pi,t) \models \varphi_1 \vee \varphi_2$ iff $(\pi,t) \models \varphi_1$ or $(\pi,t) \models \varphi_1$;

$(\pi,t) \models \Diamond_{[a,b]}\varphi$ iff there exists $t' \in [t,T] \cap [t+a, t+b]$ such that $(\pi,t') \models \varphi$;

$(\pi,t) \models \Box_{[a,b]}\varphi$ iff for all $t' \in [t,T] \cap [t+a,t+b]$ we have $(\pi,t') \models \varphi$;

$(\pi,t) \models \varphi_1 \mathcal{U}_{[a,b]}\varphi_2$ iff there exists $t' \in [t,T] \cap [t+a,t+b]$ such that $(\pi,t') \models \varphi_2$ and for all $t \leq t'' < t'$ we have $(\pi,t'') \models \varphi_1 \vee \varphi_2$.

We say a trace π satisfies an STL formula φ, denoted $\pi \models \varphi$, if $(\pi,0) \models \varphi$. □

In the above definition, we have specified the satisfaction relation for the derived operators $\Box$ and $\Diamond$ for simplicity (these two definitions can be obtained from the others).

As an example, consider a formula $\Diamond_{[1.1,3]}\big((z_1 \leq 10) \wedge (z_2 \geq 20)\big)$. Suppose we have a trace $\pi : [0,T] \to \mathbb{R}^2$. The formula has three ingredients: (1) $\Diamond$ says that at some point in the future, its inner formula must be satisfied; (2) the $\Diamond$ subscript $[1.1,3]$ says that this point in the future must be in the interval $[1.1,3]$ (provided it is within the signal horizon T; and (3) the constraint $(z_1 \leq 10) \wedge (z_2 \geq 20)$ says that at that future point, the first signal dimension value must be ≤ 10, and the second signal dimension value must be ≥ 20. The formula $\Box_{[1.1,3]}\big((z_1 \leq 10) \wedge (z_2 \geq 20)\big)$ similarly says that at all points in the future in the time interval $[1.1,3]$, the predicate $(z_1 \leq 10) \wedge (z_2 \geq 20)$ must hold.

B. Augmenting STL *with Input Classes for CPS*

A commonly used fragment of STL used for CPS property specification is of the form $\Box_{[a,b]}(\varphi_1 \to \varphi_2)$, where φ_1 and φ_2 are STL formulae. The formula φ_1 in such a case is called the triggering *antecedent* and the formula φ_2 is called the *consequent*. In many cases, the triggering condition involves only input signals, and the consequent involves only output signal variables. Often, for the engineering requirements we are interested in, the triggering antecedents can be easily specified as mathematical functions, but are cumbersome to specify in temporal logics. In this section we extend STL to incorporate these more general triggering antecedents.

Input Signal Classes. An input signal class, InSig given an arity n_I is a collection of input signals $\pi_p : [0,T_p] \to \mathbb{R}^{n_I}$. We leave the syntax of specifying signal classes open, with the understanding that the signal class can be unambiguously specified. While STL is one possible syntax, it is not the only one. For example, a signal class could be specified by the set $\{5\sin(\omega t + \theta) \mid 0 \leq \theta \leq 2\}$. Another example is a short pulse function of duration 2 seconds: define a pulse function of amplitude a as

$$f_a(t) = \begin{cases} 0 & \text{for } t < 1 \\ a & \text{for } 1 \leq t \leq 2 \end{cases}$$

A pulse signal class example is $\{f_a(t) \mid f_a(t)$ is a pulse signal, and $10 \leq a \leq 25\}$.

$\mathrm{STL}_{\mathsf{InSig}}$: Extending STL. If an STL formula φ_{op} involves only output signal variables, then we refer to it as an *output variable formula*. Given an input signal class InSig, the formulas of $\mathrm{STL}_{\mathsf{InSig}}$ are defined by the grammar

$$\psi := \quad \varphi_{\mathsf{op}} \mid \Box_{[l,u]}\left(\mathsf{InSig}_{\Delta_i} \to \Box_{[\Delta_{o_1},\Delta_{o_2}]}\varphi_{\mathsf{op}}\right) \mid \Box_{[l,u]}\left(\mathsf{InSig}_{\Delta_i} \to \Diamond_{[\Delta_{o_1},\Delta_{o_2}]}\varphi_{\mathsf{op}}\right) \tag{1}$$

where

- $0 \leq l \leq u$, and $\Delta_i \leq \Delta_{o_1} \leq \Delta_{o_2}$.
- φ_{op} is a bounded output variable STL formula, *i.e.*, an STL formula (with bounded temporal operator intervals) involving only output dimension variables of the combined input-output signal.

The additions to STL are the InSig primitives $\mathsf{InSig}_{\Delta_i}$. We handle these primitives as follows. InSig is an input signal class which contains input signals of interest as triggering antecedents. The subscript Δ_i in $\mathsf{InSig}_{\Delta_i}$ indicates only to look at those InSig signals that have length at most Δ_i. Thus $\mathsf{InSig}_{\Delta_i} \subseteq \mathsf{InSig}$. A formula $\Box_{[l,u]}\left(\mathsf{InSig}_{\Delta_i} \to \psi^*\right)$ intuitively specifies that whenever the input signal has a sub-portion that is in $\mathsf{InSig}_{\Delta_i}$, then the output signal from that place must satisfy the property ψ^*.

Definition 2 (Semantics of $\mathrm{STL}_{\mathsf{InSig}}$)**.** The satisfaction relation is defined as follows. The suffix of the trace input-output trace π (consisting of both input and output signals) from time t (for $t \leq T$ the trace horizon) is said to satisfy a $\mathrm{STL}_{\mathsf{InSig}}$ property ψ, written as $(\pi,t) \models \psi$ as follows.

- For $\psi = \varphi_{\mathsf{op}}$, a bounded STL formula over output signal variables. the satisfaction relation is as for STL.
- For $\psi = \Box_{[l,u]}(\psi')$, where ψ' has an InSig primitive, we require for all $\delta \in [l,u]$, such that $t+\delta \leq T$, the condition $(\pi, t+\delta) \models \psi'$.
  The satisfaction relation for $\psi' = \mathsf{InSig}_{\Delta_i} \to \psi^*$ is in turn defined as $(\pi,t') \models \mathsf{InSig}_{\Delta_i} \to \psi^*$ provided the following condition holds: $\forall \delta_i \leq \Delta_i$ such that $t' + \delta_i \leq T$, if $\pi_{\mathsf{ip}_{t',t'+\delta_i}} \in \mathsf{InSig}$, then $(\pi,t') \models \psi^*$. Here π_{ip} is the input signal portion of the input-output composite signal π.

A trace π satisfies an $\mathrm{STL}_{\mathsf{InSig}}$ property ψ if $(\pi,0) \models \psi$. □

In the above, the clause $\mathsf{InSig}_{\Delta_i}$ is referred to as the triggering antecedent, as the requirement ψ^* is to be fulfilled only upon the triggering of the antecedent. The role of Δ_i in the subscript is to specify that we are only interested in triggering antecedents of duration Δ_i.

Example 1. Consider the formula ψ defined as $\Box_{3,10}\left(\mathrm{Pulse}_2 \to \Diamond_{3,7}\varphi_{\mathsf{op}}\right)$ where Pulse is a class of

input pulses. A trace π over time $[0,T]$ satisfies ψ, if either (a) an input pulse of duration ≤ 2 does not start in the time interval $[3,10]$; or (b) whenever an input pulse of duration ≤ 2 starts in the time interval $[3,10]$, say at time t_p, then we have that some trace suffix $\pi_{t_p+\delta,T}$ for some $\delta \in [3,7]$ satisfies φ_{op}. Note that the antecedent trigger Pulse_2 leaves the input signal unconstrained apart from requiring that a pulse be present. For example, we could have oscillatory input from time 0 to 5, then a pulse from 5 to 6.2, and then again oscillatory input from 6.2 till the input trace horizon T. Such an input signal will still satisfy the triggering antecedent. □

In addition to this triggering antecedent in formulae, we also allow simple restrictions on the input signals *outside* the logic, such as "input signal $x(t)$ is a sinusoid of frequency ω". This differs from triggering antecedents in the formulae in that the triggers are restricted to a short time interval and the consequent requirement comes after the end of the trigger interval. Simple restrictions on the input signals outside the logic are over the entire simulation horizon, and in addition are simple enough to not require temporal logic machinery.

$\mathrm{STL}_{\mathsf{InSig}}$: Quantitative Robustness. The primary utility of quantitative robustness definitions for STL has been in the optimization based falsification framework in which the robustness function value is used to guide black-box falsifiers towards traces that violate a given specification [13], [20]. As the specification under consideration may involve triggering conditions, the framework has been used to search for inputs that satisfy the triggering constraints. It has been argued in [21], [33] that the case engineers are interested in is where the input triggering constraint is satisfied, and the consequent is violated, and hence we need to treat input and outputs differently when defining robustness. If we consider the $\mathrm{STL}_{\mathsf{InSig}}$ fragment, the only place input variables occur is in the antecedent, and this allows cleanly treating input and output variables differently. First, we define the robustness function $\rho^{\varphi}(\pi,t) \in \mathbb{R} \cup \{-\infty,+\infty\}$ for output variable STL formulae φ over a trace π at time t, where negative robustness values indicate violation of φ (this is the standard STL robusness definition [14]):

Definition 3 (Robustness Function for Output-Variable STL)**.** Let $\pi : [0,T] \to \mathbb{R}^n$ where $\pi(t) = (\pi_1(t),\ldots,\pi_n(t))$ be an output variable trace, and let μ be the predicate $f(z_1,\ldots,z_n) \geq 0$ where each z_i is an output variable. For $t \in [0,T]$ we define the *robustness value* $\rho^{\varphi}(\pi,t)$ *of* φ *over* π *at time* t as:

$$\rho^{\mu}(\pi,t) = f(\pi_1(t),\ldots,\pi_n(t)); \quad \text{where the output variables } z_j \text{ correspond to the } j\text{-th dimension of } \pi;$$

$$\rho^{\neg\varphi}(\pi,t) = -\rho^{\varphi}(\pi,t); \qquad \rho^{\mathrm{T}}(\pi,t) = +\infty;$$

$$\rho^{\varphi_1 \wedge \varphi_2}(\pi,t) = \min\left(\rho^{\varphi_1}(\pi,t), \rho^{\varphi_2}(\pi,t)\right);$$

$$\rho^{\varphi_1 \vee \varphi_2}(\pi,t) = \max\left(\rho^{\varphi_1}(\pi,t), \rho^{\varphi_2}(\pi,t)\right);$$

$$\rho^{\varphi_1 \,\mathcal{U}_{\,[a,b]}\, \varphi_2}(\pi,t) = \sup_{\tau \in (\,[t+a,t+b] \cap [0,T]\,)} \left(\min \left(\begin{array}{c} \inf\limits_{s \in [t,\tau)} \max \begin{pmatrix} \rho^{\varphi_1}(\pi,s) \\ \rho^{\varphi_2}(\pi,s) \end{pmatrix}, \\ \rho^{\varphi_2}(\pi,\tau) \end{array} \right) \right)$$

$$\rho^{\Diamond_{[a,b]}\varphi}(\pi,t) = \sup_{\tau \in ([t+a,t+b]) \cap [0,T]} \rho^{\varphi}(\pi,\tau);$$

$$\rho^{\Box_{[a,b]}\varphi}(\pi,t) = \inf_{\tau \in ([t+a,t+b)) \cap [0,T]} \rho^{\varphi}(\pi,\tau).$$

The robustness $\rho^{\varphi}(\pi)$ of φ over π is defined as $\rho^{\varphi}(\pi,0)$. □

In the above definition, we require that $t+b \leq T$. We have specified the robustness function for the derived operators $\Box$ and $\Diamond$ in the definition for simplicity (these two definitions can be obtained from the others). The equations in Definition 3 are the quantitative analogues of the equations in Definition 2, where conjunctions have been replaced with $\inf$ (or $\min$) and disjunctions have been replaced with $\sup$ (or $\max$).

The robustness function for formulae involving input signal classes is defined as follows.

Definition 4 (Robustness Function for $\mathrm{STL}_{\mathsf{InSig}}$)**.** Let $n \in \mathbb{N}$, $T \in \mathbb{R}_+$, and let $\pi : [0,T] \to \mathbb{R}^n$ be an input-output trace, with π_{op} being the output signal portion of π. Given an $\mathrm{STL}_{\mathsf{InSig}}$ formula ψ, the robustness value $\rho^{\psi}(\pi,t)$ of ψ over π at time t is defined according to the following cases:

- If $\psi = \varphi_{\mathsf{op}}$, an output-variable formula, then $\rho^{\psi}(\pi,t) = \rho^{\psi}(\pi_{\mathsf{op}},t)$ according to Definition 3.
- If $\psi = \Box_{[l,u]}\left(\mathsf{InSig}_{\Delta_i} \to \psi^*\right)$, then the robustness value is defined as $\rho^{\psi}(\pi,t) =$

$$\begin{cases} \infty & \text{if } \forall \delta_1 \in [l,u] \text{ and } \forall \delta_2 \leq \Delta_i \\ & \text{we have } \pi_{t+\delta_1,t+\delta_1+\delta_2} \notin \mathsf{InSig}; \\ \inf\limits_{\substack{\delta_1 \in [l,u],\\ \delta_2 \leq \Delta_i,\\ \pi_{t+\delta_1,t+\delta_1+\delta_2} \in \mathsf{InSig}}} \rho^{\psi^*}(\pi_{\mathsf{op}}, t+\delta_1) & \text{otherwise,} \\ & \text{where } \rho^{\psi^*}(\pi_{\mathsf{op}},t+\delta_1) \\ & \text{is as in Definition 3.} \end{cases} \tag{2}$$

  The robustness value $\rho^{\psi}(\pi)$ of ψ over π is defined to be $\rho^{\psi}(\pi,0)$. □

We explain Equation 2. The first case refers to the vacuous satisfaction of the specification ψ due to the triggering antecedent never holding for any sub-trace $\pi_{t+\delta_1,t+\delta_1+\delta_2}$. In our falsification framework, our generated input signal will ensure such vacuity does not occur, hence we assign $+\infty$ to this case. The second case is for non-vacuous satisfaction/violation, and for this a quantitative value is relevant. We compute the lowest possible value of $\rho^{\psi^*}(\pi_{\mathsf{op}},t+\delta_1)$, the robustness value of the output-variable formula consequent for such δ_1 where the triggering antecedent holds at time $t+\delta_1$, that is the sub-trace $\pi_{t+\delta_1,t+\delta_1+\delta_2} \in \mathsf{InSig}$.

As in the case of standard STL, a positive robustness value implies satisfaction of the formula, and a negative value implies violation.

Proposition 1. *Let $n \in \mathbb{N}$, $T \in \mathbb{R}_+$, let $\pi : [0,T] \to \mathbb{R}^n$ be an input-output trace, and let ψ be an* $\mathrm{STL}_{\mathsf{InSig}}$ *formula.*

1) *If $\rho^{\psi}(\pi) > 0$ then $(\pi,0) \models \psi$.*
2) *If $\rho^{\psi}(\pi) < 0$ then $(\pi,0) \not\models \psi$.* □

III. Falsification via Optimization

Models of CPS are complex, and are usually not amenable to formal analysis. In this work, we view CPS models (or actual physical systems themselves) as *black-box systems*: for

a system $\mathcal{S}$, given any input π_{ip}, we can execute $\mathcal{S}$ on π_{ip} and observe the output $\mathcal{S}(\pi_{\mathsf{ip}})$ (we assume the system output includes a copy of the system input), but we do not have further access to $\mathcal{S}$ and cannot statically analyze the system to compute an input for which the system output would violate a given specification. In recent years, a black-box *optimization* based approach has been shown to have promise for the task of searching for falsifying inputs to such complex Cyber-Physical Systems. These approaches leverage Proposition 1 which says that it suffices to get an input for which the system output has negative quantitative robustness as a negative robustness value implies an output trace which violates the corresponding logical specification.

Proposition 2. *Let $n \in \mathbb{N}$, $T \in \mathbb{R}_+$, let $\pi_{\mathsf{ip}} : [0,T] \to \mathbb{R}^{n_I}$ be an input signal such that the corresponding system input-output trace is $\mathcal{S}(\pi_{\mathsf{ip}})$. Let ψ be an* $\mathrm{STL}_{\mathsf{InSig}}$ *formula.*

1) *If $\rho^{\psi}(\mathcal{S}(\pi_{\mathsf{ip}})) > 0$ then the system $\mathcal{S}$ when given the input signal π_{ip} satisfies ψ.*
2) *If $\rho^{\psi}(\mathcal{S}(\pi_{\mathsf{ip}})) < 0$ then the system $\mathcal{S}$ when given the input signal π_{ip} violates ψ.* □

Thus this approach reduces the falsification problem to an optimization one. One can employ an optimizer to compute:

$$\min_{\pi_{\mathsf{ip}} \in \mathsf{InpSet}} \rho^{\psi}(\mathcal{S}(\pi_{\mathsf{ip}})) \tag{3}$$

where InpSet is the set of input signals to optimize over. Recall that an input signal is a function $\pi_{\mathsf{ip}} : [0,T] \to \mathbb{R}^{n_I}$. Thus the optimization search in Equation 3 is over a set InpSet of *functions*.

A *Black-Box* optimization based approach invokes optimizers that do not have access to the structure of the function f being optimized. Black-box optimizers such as Simulated Annealing based optimizers [1], and CMA-ES [23] have an optimization space that is different than the one in Equation 3. These optimizers optimize functions $f : D_1 \times \ldots \times D_p \to \mathbb{R}$, where each D_i is a subset of $\mathbb{R}$, usually a bounded interval $[l_i, u_i]$. Hence, those optimizers are heuristically solving $\min_{x_i \in D_i} f(x_1, \ldots, x_p)$.

In order to solve the optimization problem in Equation 3, we have to use a *parameterization function* $\mathsf{SigGen} : (D_1 \times \ldots \times D_p) \to ([0,T] \to \mathbb{R}^{n_I})$. The parameterization function SigGen takes a tuple of reals numbers $(d_1, \ldots, d_p)$ and generates a signal $\pi_{\mathsf{ip}} : [0,T] \to \mathbb{R}^{n_I}$. The most widespread parameterization strategy is that of *control points* [12], [5], [16], [15] Typically p time-points are chosen uniformly over the simulation horizon, and for each of these time-points t_j, there is a corresponding n_I dimensional $(d_1^j, \ldots d_{n_I}^j)$ vector where each d_k^j is in the range of the k-th dimension of the input signal. The signal π_{ip} is then constructed from these p n_I-tuples by some interpolation scheme, for example linear interpolation. Hence the optimization problem solved is.

$$\min_{x_i \in D_i} \rho^{\psi}(\mathcal{S}(\mathsf{SigGen}(x_1, \ldots, x_p))). \tag{4}$$

A uniform placement of control points is done in usually two manners.

1) A fixed number α of control points is chosen according to engineering intuition, and these α points are placed uniformly over the simulation horizon. As each control point corresponds to n_I values for the n_I dimensional input signal, this results in $\alpha \cdot n_I$ parameters in Equation 4 that the optimizer has to search over.
2) An inter control point time duration is chosen, and then control points are placed uniformly with this inter-point duration throughout the simulation horizon. For example, if the inter control point time duration is β time units, this results in $\left\lceil \frac{T}{\beta} \right\rceil + 1$ control points, and correspondingly $n_I \cdot \left(\left\lceil \frac{T}{\beta} \right\rceil + 1 \right)$ parameters for the optimizer.

It is desirable to have a strategy which uses as few a number of control points as possible as (1) a smaller number of control points gives a n_I multiplicative benefit for the number of parameters for the optimizer to search over – this significantly improves optimizer performance; and (2) a smaller number of control points increases the interpretability, and hence usefulness, of falsifying inputs if any, as this facilitates debugging of the system under test. However, a conflicting fact is that reducing the number of control points leads to signals that may be more slowly varying than needed to properly test the system.

A. Utilizing the Step Response

The *step response* of a system from a given initial state is the response of the system when given a step input $u(t)$ which is 0 for $t < 0$ and A for $t \geq 0$ for some constant A for single input single output systems, with appropriate extensions for multiple input multiple output (MIMO) systems. Step response behavior is a critical part of the analysis of control systems [26], and overshoot, rise time, and settling time are part of key characteristics for behavior analysis of dynamical systems.

In our work we propose a heuristic to reduce the number of control points based on the step response, notably based on the settling time of the system. The settling time is defined as the time elapsed from when a step input is applied to when the system enters and stays within a band of the final steady state value. Matlab® provides a function `stepinfo`, that computes settling time values (in addition to the other step response entities). Note that we do not propose *monitoring* the system over a settling time period; rather a settling time period is where we place the control points and hence *vary* the input signal. A factor to consider is the choice of where this settling time period should be. If the settling time is t_{settle} time units, the naive choice is have the control point placement interval as $[0, t_{\mathsf{settle}}]$. However, this choice is suboptimal, for instance when the energy needs to accumulate in the system affecting system dynamics later in the execution. We propose the control point placement interval as the floating interval $[t_\alpha, t_\alpha + t_{\mathsf{settle}}]$, where t_α is itself a variable, hence the offset of the t_{settle}-length control point placement interval is also determined by the optimizer. In addition to placing the control points in this interval, we also need to place two additional control points, one at 0 and one at the simulation horizon T so that the input signal over the entire simulation horizon can be constructed via interpolation.

For MIMO systems, we compute the settling time for each input variable, output variable pair (holding the other input

signals at some nominal value in their ranges), and then take the maximum of these settling times as t_{settle}. For systems with a fast response time, t_{settle} may be a small number, and placing control points in this interval may lead to input signals with high variance (compared to the simulation horizon); such input signals may not be realistic in an engineering context. In such a case, one may choose to place the given number of control points over a larger interval. We also choose to make the value of the input signal at time 0 also variable as the initial signal values can have long lasting effects. The last control point is at time $t_\alpha + t_{\mathsf{settle}}$, and the signal can be kept constant after that till the end of the simulation horizon. Note that if the input signal is over n_I variables, and there are α control points (including the one at time 0), then this strategy results in $\alpha \cdot n_I + 1$ parameters for the black box optimizer (the $+1$ is due to the additional variable t_α).

B. Handling InSig *Primitives in Formulae*

In this section we examine computing the robustness function $\rho^\psi\left(\mathcal{S}\left(\mathsf{SigGen}(x_1,\ldots,x_p)\right)\right)$ when the formula ψ has an InSig primitive. We note that this computation in our framework is only used in the optimization problem in Equation 4. This simplifies handling of the robustness computation as follows. First, we observe that in order to use a black box optimization framework, we need to construct a parameterization function SigGen. Since this construction is completely under our control, we can *by design* state what are the input signal portions $\pi_{t,t+\delta_2}$ such that the signal portion belongs to InSig as required in Equation 2. We explain with an example. Suppose our triggering input signal class is a *rise* signal that goes from a low value to a high value in a very short amount of time (*e.g.*, 0.05 time units), in almost a step fashion. It is easy to design a parameterization function using control points or other mechanisms that will by design (1) generate input signals $\pi_{\mathsf{ip}} = \mathsf{SigGen}(x_1,\ldots,x_p)$ that belong to this class; and (2) lead to inferable input signal portions $\pi_{\mathsf{ip}_{t,t+\delta_2}} \in \mathsf{InSig}$, *i.e.*, we know what are the t, δ_2 pairs that lead to the input signal being in InSig. In other words, we are by design generating $\pi_{\mathsf{ip}_{t,t+\delta_2}} \in \mathsf{InSig}$ events at desired time-points $t + \delta_2$. Since we are designing where the triggering InSig events happen, we can use this information to compute the robustness value of a formula having InSig primitives as follows.

Suppose we have a generated signal $\pi_{\mathsf{ip}} = \mathsf{SigGen}(x_1,\ldots,x_p)$ such that there is only one t, δ_2 pair such that $\pi_{\mathsf{ip}_{t,t+\delta_2}} \in \mathsf{InSig}$. Then for the formula $\psi = \Box_{[l,u]}\left(\mathsf{InSig}_{\Delta_i} \rightarrow \Box_{[\Delta_{o_1},\Delta_{o_2}]}\varphi_{\mathsf{op}}\right)$ we have $\pi \models \psi$ iff $t \in [l,u]$ and $\pi \models \Box_{[\Delta_{t+o_1},\Delta_{t+o_2}]}\varphi_{\mathsf{op}}$. Along similar lines, for the robustness value, we have

$$\rho^\psi\left(\mathcal{S}\left(\pi_{\mathsf{ip}}\right)\right) = \begin{cases} \infty & \text{if } t \notin [l,u] \\ \rho^{\Box_{[t+\Delta_{o_1},t+\Delta_{o_2}]}\varphi_{\mathsf{op}}}\left(\mathcal{S}\left(\pi_{\mathsf{ip}}\right)\right) & \text{otherwise} \end{cases} \tag{5}$$

If rather than one t, δ_2 pair such that $\pi_{\mathsf{ip}_{t,t+\delta_2}} \in \mathsf{InSig}$, we have several such pairs, we can similarly construct a procedure to eliminate the InSig primitive by taking a conjunction, provided the number of such pairs is finite. This was for the case of a particular input signal $\pi_{\mathsf{ip}} = \mathsf{SigGen}(x_1,\ldots,x_p)$. In the optimization problem in Equation 4, the input signals vary depending on the choice of the optimization variable. We can obtain an equivalent optimization problem leveraging the insight above.

Proposition 3. *Consider the formula* $\psi = \Box_{[l,u]}\left(\mathsf{InSig}_{\Delta_i} \rightarrow \Box_{[\Delta_{o_1},\Delta_{o_2}]}\varphi_{\mathsf{op}}\right)$. *Suppose* SigGen *is a parameterization strategy such that for any* $(x_1,\ldots,x_p) \in D_1 \times \ldots D_p$ *we have exactly one timepoint* t *(which we denote as* $\mathsf{tg}(x_1,\ldots,x_p)$*), and one* $\delta_2 \leq \Delta_i$ *such that the generated input signal* $\pi_{\mathsf{ip}} = \mathsf{SigGen}(x_1,\ldots,x_p)$ *contains the subsignal* $\pi_{\mathsf{ip}_{t,t+\delta_2}} \in \mathsf{InSig}$. *Then the the optimization problem in Equation 4 has the same value as the following optimization problem.*

$$\min_{(x_1^j,\in\ldots,x_p^j)\in D_1\times\ldots D_p} \rho^{\theta(x_1^j,\ldots,x_p^j)}\left(\mathcal{S}\left(\mathsf{SigGen}(x_1^j,\ldots,x_p^j)\right)\right). \tag{6}$$

where $\theta(x_1^j,\ldots,x_p^j) = \Box_{[\mathsf{tg}(x_1^j,\ldots,x_p^j)+\Delta_{o_1},\ \mathsf{tg}(x_1^j,\ldots,x_p^j)+\Delta_{o_2}]}\varphi_{\mathsf{op}}$. □

Note that in Equation 6, the formula for the robustness function *depends in each iteration of the optimizer*, that is when the optimizer changes its parameters $x_1^j,\ldots,x_p^j$ in the j-th iteration, the formula $\theta(x_1^j,\ldots,x_p^j)$ *also changes*. In comparison, the usual approach for black-box optimization based falsification has been to keep the formula the same during the optimization process. Similar results hold for $\Box_{[l,u]}\left(\mathsf{InSig}_{\Delta_i} \rightarrow \Diamond_{[\Delta_{o_1},\Delta_{o_2}]}\varphi_{\mathsf{op}}\right)$, and for cases where we have a multiple, but finite number of InSig generation events in the same input trace.

Proposition 3 can be used in a tool framework such as Breach as follows. In the normal optimization flow, a black-box optimizer repeatedly computes parameterization instantiations $(x_1^1,\ldots,x_p^1),\ldots,(x_1^j,\ldots,x_p^j)$ and Breach computes the robustness values for each instantiation in a loop, $\rho^\psi\left(\mathcal{S}\left(\mathsf{SigGen}(x_1^1,\ldots,x_p^1)\right)\right),\ldots,$ $\rho^\psi\left(\mathcal{S}\left(\mathsf{SigGen}(x_1^j,\ldots,x_p^j)\right)\right)$. The formula for the robustness computation remains the same – ψ – for each parameterization instantiation. In the case of formulas in $\mathrm{STL}_{\mathsf{InSig}}$ involving InSig primitives, Proposition 3 implies that we can use the existing implemented routines for standard STL in Breach provided we change the formula appropriately for different parameter instantiations.

There is another method to take advantage of the fact that the design of the parameterization function is under our control. In contrast to the previous method, this second method does not require changing the formula for different parameterization instantiations.

Proposition 4. *Consider an* $\mathrm{STL}_{\mathsf{InSig}}$ *formula* $\Box_{[l,u]}\left(\mathsf{InSig}_{\Delta_i} \rightarrow \psi^*\right)$ *where* ψ^* *is* $\Box_{[\Delta_{o_1},\Delta_{o_2}]}\varphi_{\mathsf{op}}$, *or* $\Diamond_{[\Delta_{o_1},\Delta_{o_2}]}\varphi_{\mathsf{op}}$. *Suppose* SigGen *is a parameterization scheme for which there is an* STL *formula* φ_{ip} *over only input variables (not having any* InSig *primitives) such that the following conditions hold for all* $(x_1,\ldots,x_p) \in D_1 \times \ldots D_p$ *where* D_i *is the range for* x_i*:*

1) *there exists* t *and* $\delta_2 \leq \Delta_i$ *such that* $\mathsf{SigGen}_{t,t+\delta_2}(x_1,\ldots,x_p) \in \mathsf{InSig}$*; and*
2) $\mathsf{SigGen}_{t,t+\delta_2}(x_1,\ldots,x_p) \in \mathsf{InSig}$ *for* $\delta_2 \leq \Delta_i$ *iff* $\mathsf{SigGen}_{t,T}(x_1,\ldots,x_p) \models \varphi_{\mathsf{ip}}$.

Then $\mathsf{SigGen}(x_1,\ldots,x_p) \models \Box_{[l,u]}\left(\mathsf{InSig}_{\Delta_i} \rightarrow \psi^*\right)$ *iff* $\mathsf{SigGen}(x_1,\ldots,x_p) \models \Box_{[l,u]}\left(\varphi_{\mathsf{ip}} \rightarrow \psi^*\right)$. □

Proposition 4 states that if a input signal formula φ_{ip} exists such that *if we restrict input signal to be those generated from the parameterization scheme* satisfying the conditions of the proposition for the formula φ_{ip}, then the the subtrace $\mathsf{SigGen}_{t,t+\delta_2}(x_1,\ldots,x_p)$ is in InSig iff $\mathsf{SigGen}_{t,T}(x_1,\ldots,x_p) \models \varphi_{\mathsf{ip}}$; that is, the formula φ_{ip} captures the signal class InSig *provided* we have a signal parameterization strategy SigGen satisfying the stated conditions. In this case, one can replace the formula $\square_{[l,u]}\left(\mathsf{InSig}_{\Delta_i} \rightarrow \psi^*\right)$ with $\square_{[l,u]}\left(\varphi_{\mathsf{ip}} \rightarrow \psi^*\right)$ in the black-box falsification loop, provided we use a robustness definition ρ that treats input antecedents differently from output consequents as in Definition 4; that is the ρ is such that $\rho^{\varphi_{\mathsf{ip}} \rightarrow \psi^*}(\pi,t)$ equals ∞ if $(\pi_{\mathsf{ip}},t) \not\models \varphi_{\mathsf{ip}}$ and equals $\rho^{\psi^*}(\pi_{\mathsf{op}},t)$ as in Definition 3 when $(\pi_{\mathsf{ip}},t) \models \varphi_{\mathsf{ip}}$; where π_{ip} and π_{op} are the input and output portions of the input-output trace π. Note that such an input formula φ_{ip} might not always exist for a given input signal class InSig, in which case we have to rely on Proposition 3.

Example 2. Consider a InSig class defined as a set of signals that have a low value for 0.5 seconds, then a steep rise within 0.05 seconds to a high value which is held for 0.5 seconds. Hence the InSig class defines a particular type of signals that have a *rise* event. An input signal belonging to this class occurs as a subsignal from 10 to 11.05 seconds in Figure 1. The signal has a low value from 10 to 10.5 seconds, then it rises steeply to a high value in 0.05 seconds, and then stays at that high value for another 0.5 seconds. Now consider an STL

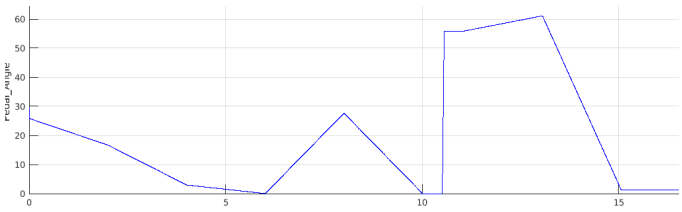

Fig. 1. Example: *rise* Input Signal

formula taken from [17]: $\varphi_{\mathsf{ip}} = (\theta < 8.8) \wedge \Diamond_{[0,0.05]}(\theta > 40)$. One can construct input signals very brief spikes such that the signal goes from a low value to a high value and then back again to a low value within 0.05 seconds. Such a signal would satisfy the formula φ_{ip}, but it does not capture the design intent. Consider a signal parameterization strategy which has a four control point segment, where the time placement of these points in $0, 0.5, 0.55, 1.05$, with the constraint that the signal values at the first two control points, and the last two control points is the same. Under this constraint, if the generated signal $\mathsf{SigGen}(x_1,\ldots,x_4)$ satisfies φ_{ip}, then it also is in the specified InSig class of our example. □

IV. Experiments

We implemented our approach in the Breach tool. Three benchmarks from the ARCH-COMP repository [17] were used for our experiments. These benchmarks are $Simulink^{®}$ based models of complex control systems, and have been used widely in CPS falsification community [16], [15].

Each benchmark model has various associated temporal logic specifications. Some involve only output variables, *i.e.*, these are InSig free, and some have input signal class constraints. We employed the CMA-ES black-box optimizer in Breach, and gave a fixed simulation budget, that is, the number of input signals over which the system was executed was fixed. Due to the stochastic nature of the optimizers, the resulting best robustness values (the smaller the value, the better for falsification) in repeated experiments were different. Bearing this is mind, we ran multiple experiments with the same parameters (but with different seeds).

InSig **free formulae.** For the baseline input signal exploration strategy, we generated signals using a uniform placement of control point over the simulation horizon, with a constant inter control point (CP) time distance. This inter CP distance was chosen based on the ARCH-Comp reports [16], [15], and was kept the same between the baseline, and our proposed approach to have a fair comparison. For our heuristic, we reduced the number of control points by reducing the interval in which we placed the CPs; from the entire simulation horizon of the baseline, to a time interval approximately corresponding to the settling time as argued in Subsection III-A. As our systems are MIMO systems, we looked at the step response of each input-output pair (keeping the other input constant, using the `stepinfo()` routine in Matlab® for a band of 5% of the final value, and took the largest settling time. We performed two sets of experiments for this reduced time interval; one where the reduced interval had floating placement (the placement of the floating interval being an optimization variable, and one where the reduced interval had a fixed placement from the start of the simulation.

Formulae with InSig**.** We considered two InSig classes in our experiments, those capturing *rise* and *fall* events, along the lines as in Example 2. If we use a uniform control point placement strategy for the baseline as for the InSig free formulae, this would result in all generated input signal not satisfying the triggering antecedent (the antecedent requires a steep rise within 0.05 seconds). The antecedent getting triggered requires control point placement at 0.05 second intervals, which in a uniform placement strategy results in too large a number of optimization variables (in addition to unrealistic input signals). As a result, for the baseline, we generated a *fixed* set of input signals with one rise event, where the rise event for the j-th signal occurred at j seconds; and we took the best robustness value corresponding to this set of fixed signals. Hence the baseline in this case did not involve an optimizer. For both the baseline and our heuristic, we chose the interval in which to vary the signal again to be of length the settling time. The non-rise and non-fall portions had uniformly placed control points of inter-CP distance 2. The rise/fall portion of the input took another 1.05 seconds.

Optimizer/Experiment Parameters. CMA-ES allows setting both the maximum number of function evaluations (which is the same as the maximum number of simulations for the Simulink model), as well as the maximum time limit per experiment. Table I gives the number of experiments for each logical specification, and the maximum number of function evaluations allowed per experiment. The ballpark experiment times were 10 minutes for AT, 7 minutes for AFC, and 2 hours for WT. The experiment time was primarily determined by the simulation time for the corresponding Simulink model.

TABLE I
EXPERIMENTAL SETUP FOR EACH BENCHMARK

Benchmark	# Experiments per property	# Objective evaluations per experiment
AT	20	800
AFC	30	800
WT	20	200

A. Automatic Transmission (AT)

This is a model of an automatic transmission system derived from model from Mathworks. For our purposes, there are two input variables $throttle, brake$; and three outputs $RPM, gear$ and $speed$. The input ranges are $throttle \in [0, 100]$ and $brake \in [0, 325]$ (both can be active at the same time). A description of the model can be found in [25]. The main components of the model are three subunits for the engine, the transmission, and the vehicle. The closed-loop engine block computes the engine's RPM from the input signals *throttle* and impeller torque. Further, the transmission block computes the output torque and impeller torque from the engine's RPM (computed by the engine block), gear value and transmission RPM. The gear status and transmission RPM are computed by a closed-loop gear block and the vehicle block respectively. The vehicle block computes the output signal *speed* and the transmission RPM from the output torque (coming from the transmission block) and the input signal *brake*. Meanwhile, the gear block takes vehicle speed, and commands to up-shift or down-shift the gear from the provided *throttle* We used the following slight modifications of specifications from [33]. The minor modifications were made to accept more executions as good, by increasing the constants; for example, the specification φ^2_{AT} was changed from $\square\,(speed \leq 120)$ to $\square\,(speed \leq 160)$.

$$\varphi^1_{\text{AT}} = \square\,((gear = 3) \rightarrow speed \geq 20) \tag{7}$$

$$\varphi^2_{\text{AT}} = \square\,(speed \leq 160) \tag{8}$$

$$\varphi^3_{\text{AT}} = \square\,(RPM \leq 4800) \tag{9}$$

$$\varphi^4_{\text{AT}} = \square\,((gear = 4) \rightarrow speed \geq 35) \tag{10}$$

Experiment Parameters. We used a simulation horizon of 120 seconds and placed CPs at 2 second intervals for the baseline; thus the baseline had $120/2 + 1 + 1 = 62$ control points. For our heuristic, we identified 52 seconds as the settling time. This is a MIMO system, we looked at the step response of each input-output pair (keeping the other input constant), using the `stepinfo()` routine in Matlab® for a band of 5% of the final value, and took the largest settling time. We chose the same 2 second inter-CP distance as in the baseline. This resulted in $52/2 + 1 + 1 = 28$ control points. The placement of the this smaller time interval was itself an optimization variable. The full signal was constructed using linear interpolation. We ran 20 experiments in each case, each experiment evaluated the system over 800 inputs.

B. Air Fuel Control (AFC)

The model from [24] contains an air fuel controller for a powertrain model which aims to keep air-to-fuel ratio close to the stoichiometric ratio of 14.7. This is the ideal for the vehicle performance concerning the fuel consumption and is a key to maintaining vehicle response, carbon emission, and vehicle efficiency. The Simulink model is made up of numerous subsystems containing various types of continuous and discrete blocks, including look-up tables and a hybrid automaton. The Inputs of this system model are $PedalAngle$ and $EngineSpeed$, the output for our purpose is the air-to-fuel ration AF. We would like this ratio to be as close as possible to the reference air-to-fuel ratio $AFref$ which is a constant 14.7. The input $PedalAngle$ range is $[0, 61.2]$ in normal mode, and $[61, 2, 81.2]$ in power mode. The engine speed range is $[900, 1100]$.

We used the following two properties from [17], with larger constants to allow more relaxed requirements.

$$\varphi^1_{\text{AFC}} = \square_{[11,50]}\,(|AF - AFref| < 0.03 \cdot AFref) \quad \text{for input range (normal)} \quad 0 \leq PedalAngle < 61.2 \tag{11}$$

$$\varphi^2_{\text{AFC}} = \square_{[11,50]}\,(|AF - AFref| < 0.06 \cdot AFref) \quad \text{for input range (power)} \quad 61.2 \leq PedalAngle < 81.2 \tag{12}$$

We also modified two requirements from [17] to get two $\text{STL}_{\mathsf{InSig}}$ formulae with input triggering $rise$ and $fall$ events, and the input $PedalAngle$ being in normal mode $0 \leq PedalAngle < 61.2$.

$$\varphi^3_{\text{AFC}} = \square_{[11,50]}\left(rise_{1.05} \rightarrow \left(\square_{[1,5]}|AF - AFref| < 0.02 \cdot AFref\right)\right) \tag{13}$$

$$\varphi^4_{\text{AFC}} = \square_{[11,50]}\left(fall_{1.05} \rightarrow \left(\square_{[1,5]}|AF - AFref| < 0.02 \cdot AFref\right)\right) \tag{14}$$

The $rise$ signal class denotes signals that have a low constant value for 0.5 seconds, then a steep rise to a high value within 0.05 seconds, which is then held constant for another 0.5 seconds (see Example 2), and similarly for the $fall$ signal class. A low value is defined to be a value in the range $[0, 8.7]$, a high value is in the range $[40.1, 61.2]$. The formulae capture the temporary relaxed requirements on the air-to-fuel ratio following a rapid change in the input.

Experiment Parameters. The simulation horizon was 50 seconds. The inter-CP distance was kept at 2 seconds for the InSig free formulae for both the baseline and our heuristic. The baseline had $50/2+1+1 = 27$ control points. The settling time from the step response was computed to be 12 seconds for a 5% band. This resulted in $12/2 + 1 + 1 = 8$ control points.

For the formulae that had $rise$ and $fall$ triggering constraints, for our heuristic we used 6 CPs at 2 second intervals to generate the non-rise and non-fall portions, and we used 4 additional CPs placed at $t, t + 0.5, t + 0.55, t + 1.05$ (for varying t to generate the $rise$ and $fall$ sub-portions of the pedal angle input. An example of a generated input is the signal in Figure 1. For the rise and fall experiments, we used the earlier CP placement for the engine speed input as engine speed was not part of the triggering constraint. We solved the falsification optimization problem using the reduction from Proposition 3, hence each iteration of the optimization loop used a different temporal formula. For the baseline, we used a fixed pulse (low value 5, high value 50), and we constructed input signals with this single pulse starting at $11, 12, 13, 14, \ldots 44$ seconds (the consequent clause after the pulse, that took 1.05 seconds, was

for 5 additional seconds, and the simulation horizon was 50 seconds). For the rise *rise* case, the signal had a constant value of 5 before the pulse, and linearly dropped to 0 over the simulation horizon after the pulse, and similarly for the *fall* case. Thus, in the baseline, the only variable was the time placement of the pulse.

In all cases, the full signal was generated using linear interpolation. We ran 30 experiments in each case, each experiment evaluated the system over 800 inputs.

C. Wind Turbine (WT)

The model is a simplified wind turbine model proposed in [31] The input to the model is the wind speed v and outputs are the blade pitch angle θ, generator torque M_{gd}, the rotor speed Ω and demanded blade pitch angle θ_d [15]. The input wind speed is constrained by $8 \leq v \leq 16$. The model has two configuration, we used the configuration "Allruns" which allows variable wind speeds as inputs. We used specifications from [15], with minor modifications to the constants to relax the specifications.

$$\varphi^1_{\mathrm{WT}} = \square_{[30,200]}\ (\theta \leq 14.4) \tag{15}$$

$$\varphi^2_{\mathrm{WT}} = \square_{[30,200]}\ (\Omega \leq 14.5) \tag{16}$$

$$\varphi^3_{\mathrm{WT}} = \square_{[30,200]}\ \big(\Diamond_{[0,5]}\ (|\theta - \theta_d| \leq 1.6)\big) \tag{17}$$

$$\varphi^4_{\mathrm{WT}} = \Diamond_{[190,195]}\ (|\theta - \theta_d| \leq 1.6) \tag{18}$$

We constructed formula φ^4_{WT} as a derived implication of the formula φ^3_{WT}.

Experiment Parameters. We used a simulation horizon of 200 seconds, with an inter-CP distance of 5 seconds for both the baseline and our heuristic. The baseline had $200/5+1+1 = 42$ control points. The largest settling time from the step response data was 25 seconds. This resulted in $25/5+1+1 = 7$ control points. We ran 20 experiments in each case, each experiment evaluated the system over 800 inputs.

D. Interfaces of Benchmarks

A summary of the input and output signals of the benchmarks is given in Table II.

TABLE II
BENCHMARKS WITH INPUT AND OUTPUT SIGNALS

Model	Input Signals	Input Ranges	Output Signal
AFC	Pedal_Angle	[0,100]	AF (air-to-fuel ratio)
	Engine_Speed	[900,1100]	
AT	throttle	[0,100]	speed
	brake	[0,325]	RPM
			gear
Wind Turbine	wind	[8,16]	Theta (Θ)
			Theta_d (Θ_d)
			Omega (Ω)
			Mg (M_{gd})

E. Results: Falsification Rates

Table III presents the falsification results of our experiments. It gives the falsification rate, as well as the average number of test inputs required to falsify the given formulae. Falsification was deemed to have occurred when the robustness value turned negative (Proposition 2). Our proposed method varies the placement of the control point interval with the placement being governed by the optimizer. For the AT benchmarks, our proposed method performs significantly better than the baseline strategy. For φ^2_{AFC}, we also perform significantly better. For φ^1_{AFC} while we have a comparable falsification rate as the baseline, the simulation iteration at which we first falsify is somewhat higher. The other two AFC experiments, $\varphi^3_{\mathrm{AFC}}, \varphi^4_{\mathrm{AFC}}$. had input triggering constraints, the *rise* and *fall* triggering constraints, and our method with circumventing STL for the input constraints gets a perfect falsification rate compared to no falsifications in the baseline. For WT, we perform somewhat worse for φ^2_{WT}, significantly worse in φ^1_{WT}, and for the other two, both the baseline and our method were unable to falsify.

TABLE III
FALSIFICATION RATES. **FR**: % SUCCESSFULLY FALSIFIED; **AVG # OBJ EVAL**: AVERAGE NUMBER OF OBJECTIVE EVALUATIONS AT WHICH FIRST FALSIFIED

Model	Property	Baseline		Proposed Heuristic	
		FR %	Avg # Obj eval	FR %	Avg # Obj eval
AT	φ^1_{AT}	15	180	100	3
	φ^2_{AT}	0	–	100	53
	φ^3_{AT}	0	–	95	64
	φ^4_{AT}	50	364	100	4
AFC	φ^1_{AFC}	76.66	208	73.33	400
	φ^2_{AFC}	0	–	66.67	334
	φ^3_{AFC}	0	–	83.33	39
	φ^4_{AFC}	0	–	70	48
WT	φ^1_{WT}	80	79	5	163
	φ^2_{WT}	100	35	75	56
	φ^3_{WT}	0	–	0	–
	φ^4_{WT}	0	–	0	–

F. Results: Robustness Value Minimization

While the high level goal is falsification, engineers are concerned with how far away the system is from failure, or the degree of failure, that is they are concerned with the actual robustness values, not just whether the robustness value is negative or not. For example, a positive robustness value that is close to 0 may be viewed as indicating a problem in the system that needs to be investigated. Table IV presents the robustness values for the benchmarks (the objective was to get as low a robustness value as possible). "Floating interval placement" is for the heuristic where the placement of the control point interval was variable, and decided by the optimizer. "Fixed Interval Placement" is for the fixed placement of this interval, with the interval starting from time $t = 0$. We also normalized the robustness values. Normalization was done by multiplying the robustness value by $100/c$ where c was the constant the (error) signal (*e.g.*, the error signal $|AF - AFref|$ in φ^3_{AFC}) was compared to in the formula. We used two aggregate measures for comparing performance: avg – the average robustness over different experiments (with the same parameters); and avg_2 – the average robustness over the top 50% of the experiments. We used the second measure as in some benchmarks, the worst/best 30-40% of the robustness values were heavily skewing the results overall otherwise in the standard average.

Floating versus Fixed Interval: Some of the experiments did not see much improvement (lower values are better) from a floating CP interval compared to a fixed interval, however some saw significant improvement, for instance in φ^2_{AFC}.

TABLE IV
RESULTS COMPARISON FOR BASELINE, FLOATING INTERVAL PLACEMENT, AND FIXED INTERVAL PLACEMENT; avg/std: NORMALIZED AVERAGE/STD ROBUSTNESS; avg_2/std_2: NORMALIZED AVERAGE/STD ROBUSTNESS OF TOP 50%

Model	Property	Baseline				Floating Interval Placement				Fixed Interval Placement			
		avg	std	avg_2	std_2	avg	std	avg_2	std_2	avg	std	avg_2	std_2
AT (62;28 CPs)	φ^1_{AT}	3.804	2.2	2.608	2.65	-0.877	1.16e-3	-0.878	2.63e-4	-0.877	1.54e-3	-0.877	2.32e-4
	φ^2_{AT}	1.570	0.732	1.018	0.388	-0.694	0.120	-0.780	0.063	-0.752	0.134	-0.854	0.045
	φ^3_{AT}	0.634	0.339	0.514	4.89e-3	-4.023	1.09	-4.362	0.028	-4.280	0.077	-4.332	0.048
	φ^4_{AT}	1.230	1.57	-0.276	0.106	-0.524	1.02e-3	-0.525	0.28e-3	-0.523	1.55e-3	-0.524	0.377e-3
AFC (27;8 CPs)	φ^1_{AFC}	-8.004	9.06	-15.102	6.83	-5.646	13.1	-13.991	11.5	14.892	24.1	-4.443	16.1
	φ^2_{AFC}	32.154	7.24	27.029	4.67	2.800	21.7	-12.925	3.95	32.019	9.99	24.024	4.23
(10 CPs)	φ^3_{AFC}	97.279	2.41	94.728	3.08	-125.78	60.5	-170.92	51.7	-17.586	34.593	-44.753	25.295
	φ^4_{AFC}	95.068	4.44	90.136	5.69	-83.74	45.2	-122.28	30.3	-5.406	29.422	-26.340	28.367
WT (42;7 CPs)	φ^1_{WT}	-0.007	0.103	-0.066	0.0294	0.319	0.2	0.176	0.139	0.529	0.272	0.309	0.114
	φ^2_{WT}	-0.367	0.107	-0.450	0.0374	-0.0848	0.341	-0.313	0.0862	13.113	0.417	12.787	0.0883
	φ^3_{WT}	5.475	2.56	3.469	1.09	14.060	4.74	10.331	2.69	83.027	0.587	82.794	0.756
	φ^4_{WT}	43.306	8.58	36.200	3.44	59.252	17.66	46.540	11.65	100	0	100	0

Floating point was superior in all, except in φ^2_{AT}, where the miniscule difference (less than 0.1% in the robustness value average) can be chalked to stochasticity of the optimizer.

Floating versus Baseline: Our floating heuristic performed better than the baseline (lower values are better), except in three cases: φ^1_{AFC} (1.1% higher robustness value average), φ^3_{WT} (6.86% higher robustness value average), and φ^4_{WT} (10.3% higher robustness value average)), where the comparison was between the normalized avg_2 values.

V. CONCLUSION

In this work we presented a formal treatment of augmenting STL with general input signal class primitives, obtaining the logic $\mathrm{STL}_{\mathsf{InSig}}$ for use in a black-box falsification framework. We presented a careful analysis which showed how to incorporate these more general signals, that are difficult to capture in an STL framework alone, in the black-box optimization loop by changing the temporal logic formula of the robustness function in each iteration of the loop (Proposition 3). We also presented and explored a heuristic to reduce the interval size over which input signals need to be varied in order to falsify requirements. Note that we do not reduce the simulation horizon. Additionally, while we reduce the interval duration over which to vary the input signals, *where* this interval lies is itself variable, and this position is searched for by the optimizer. Our interval reduction heuristic is based on the settling time concept from Control Systems; and the smaller interval is especially useful for debugging purposes as it results in simpler error demonstrating inputs that have smaller signal variation. For most of the benchmarks, our heuristic resulted in better performance evidenced by lower (*i.e.*, more error indicating) robustness values. For the few benchmarks where our heuristic resulted in higher robustness values, the robustness value increase was small. Thus, the positives of the obtained simpler erroneous input signals makes this heuristic promising for debugging purposes.

As future work, the reduced smaller interval can be combined with other heuristics to improve the performance further; such as non-uniform placement of control points. Other directions are to explore other control systems concepts to obtain more targeted intervals over which to vary the input signals in order to reduce the dimension of the input search space further, and to explore which input signal classes are good for falsification.

REFERENCES

[1] Houssam Abbas and Georgios Fainekos. Convergence proofs for simulated annealing falsification of safety properties. In *50th Annual Allerton Conference on Communication, Control, and Computing*, pages 1594–1601. IEEE, 2012.

[2] Takumi Akazaki. Falsification of conditional safety properties for cyber-physical systems with gaussian process regression. In *Runtime Verification - 16th International Conference, RV 2016*, volume 10012 of *Lecture Notes in Computer Science*, pages 439–446. Springer, 2016.

[3] Takumi Akazaki and Ichiro Hasuo. Time robustness in MTL and expressivity in hybrid system falsification. In *Computer Aided Verification - 27th International Conference, CAV 2015, Proceedings, Part II*, volume 9207 of *Lecture Notes in Computer Science*, pages 356–374. Springer, 2015.

[4] Rajeev Alur, Tomás Feder, and Thomas A. Henzinger. The benefits of relaxing punctuality. *J. ACM*, 43(1):116–146, 1996.

[5] Yashwanth Annpureddy, Che Liu, Georgios E. Fainekos, and Sriram Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *Tools and Algorithms for the Construction and Analysis of Systems - 17th International Conference, TACAS 2011*, volume 6605 of *Lecture Notes in Computer Science*, pages 254–257. Springer, 2011.

[6] Benoît Barbot, Nicolas Basset, and Thao Dang. Generation of signals under temporal constraints for CPS testing. In Julia M. Badger and Kristin Yvonne Rozier, editors, *NASA Formal Methods - 11th International Symposium, NFM 2019,*, volume 11460 of *Lecture Notes in Computer Science*, pages 54–70. Springer, 2019.

[7] Benoît Barbot, Nicolas Basset, Thao Dang, Alexandre Donzé, James Kapinski, and Tomoya Yamaguchi. Falsification of cyber-physical systems with constrained signal spaces. In *NASA Formal Methods - 12th International Symposium, NFM 2020,*, volume 12229 of *Lecture Notes in Computer Science*, pages 420–439. Springer, 2020.

[8] Krishnendu Chatterjee and Vinayak S. Prabhu. Quantitative temporal simulation and refinement distances for timed systems. *IEEE Transactions on Automatic Control*, 60(9):2291–2306, 2015.

[9] Jyotirmoy Deshmukh, Xiaoqing Jin, Rupak Majumdar, and Vinayak Prabhu. Parameter optimization in control software using statistical fault localization techniques. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 220–231, 2018.

[10] Jyotirmoy V. Deshmukh, Xiaoqing Jin, James Kapinski, and Oded Maler. Stochastic local search for falsification of hybrid systems. In *Automated Technology for Verification and Analysis - 13th International Symposium, ATVA 2015*, volume 9364 of *Lecture Notes in Computer Science*, pages 500–517. Springer, 2015.

[11] Adel Dokhanchi, Shakiba Yaghoubi, Bardh Hoxha, and Georgios Fainekos. Vacuity aware falsification for MTL request-response specifications. In *13th IEEE Conference on Automation Science and Engineering, CASE 2017*, pages 1332–1337. IEEE, 2017.

[12] Alexandre Donzé. Breach, A toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification, 22nd International Conference, CAV 2010*, volume 6174 of *Lecture Notes in Computer Science*, pages 167–170. Springer, 2010.

[13] Alexandre Donzé, Thomas Ferrère, and Oded Maler. Efficient robust monitoring for STL. In *Computer Aided Verification - 25th International Conference, CAV 2013*, volume 8044 of *Lecture Notes in Computer Science*, pages 264–279. Springer, 2013.

[14] Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *Formal Modeling and Analysis of Timed Systems - 8th International Conference, FORMATS 2010*, volume 6246 of *Lecture Notes in Computer Science*, pages 92–106. Springer, 2010.

[15] Gidon Ernst, Paolo Arcaini, Ismail Bennani, Aniruddh Chandratre, Alexandre Donzé, Georgios Fainekos, Goran Frehse, Khouloud Gaaloul, Jun Inoue, Tanmay Khandait, Logan Mathesen, Claudio Menghi, Giulia Pedrielli, Marc Pouzet, Masaki Waga, Shakiba Yaghoubi, Yoriyuki Yamagata, and Zhenya Zhang. ARCH-COMP 2021 category report: Falsification with validation of results. In *8th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21)*, volume 80 of *EPiC Series in Computing*, pages 133–152. EasyChair, 2021.

[16] Gidon Ernst, Paolo Arcaini, Ismail Bennani, Alexandre Donzé, Georgios Fainekos, Goran Frehse, Logan Mathesen, Claudio Menghi, Giulia Pedrielli, Marc Pouzet, Shakiba Yaghoubi, Yoriyuki Yamagata, and Zhenya Zhang. ARCH-COMP 2020 category report: Falsification. In *ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)*, volume 74 of *EPiC Series in Computing*, pages 140–152. EasyChair, 2020.

[17] Gidon Ernst, Paolo Arcaini, Alexandre Donzé, Georgios Fainekos, Logan Mathesen, Giulia Pedrielli, Shakiba Yaghoubi, Yoriyuki Yamagata, and Zhenya Zhang. ARCH-COMP 2019 category report: Falsification. In *ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systemsi, part of CPS-IoT Week 2019*, volume 61 of *EPiC Series in Computing*, pages 129–140. EasyChair, 2019.

[18] Gidon Ernst, Sean Sedwards, Zhenya Zhang, and Ichiro Hasuo. Falsification of hybrid systems using adaptive probabilistic search. *ACM Trans. Model. Comput. Simul.*, 31(3):18:1–18:22, 2021.

[19] Georgios E. Fainekos and George J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theor. Comput. Sci.*, 410(42):4262–4291, 2009.

[20] Georgios E. Fainekos, Sriram Sankaranarayanan, Koichi Ueda, and Hakan Yazarel. Verification of automotive control applications using s-taliro. In *American Control Conference, ACC, 2012*, pages 3567–3572. IEEE, 2012.

[21] Thomas Ferrère, Dejan Nickovic, Alexandre Donzé, Hisahiro Ito, and James Kapinski. Interface-aware signal temporal logic. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019*, pages 57–66. ACM, 2019.

[22] Iman Haghighi, Noushin Mehdipour, Ezio Bartocci, and Calin Belta. Control from signal temporal logic specifications with smooth cumulative quantitative semantics. In *58th IEEE Conference on Decision and Control, CDC 2019*, pages 4361–4366. IEEE, 2019.

[23] Nikolaus Hansen. The CMA evolution strategy: A tutorial. *CoRR*, abs/1604.00772, 2016.

[24] Xiaoqing Jin, Jyotirmoy V. Deshmukh, James Kapinski, Koichi Ueda, and Kenneth R. Butts. Powertrain control verification benchmark. In *17th International Conference on Hybrid Systems: Computation and Control (part of CPS Week), HSCC 2014*, pages 253–262. ACM, 2014.

[25] Xiaoqing Jin, Alexandre Donzé, Jyotirmoy V. Deshmukh, and Sanjit A. Seshia. Mining requirements from closed-loop control models. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, 34(11):1704–1717, 2015.

[26] Kang-Zhi Liu and Yu Yao. *Robust Control: Theory and Applications*. Wiley Publishing, 1st edition, 2016.

[27] Rupak Majumdar and Vinayak S. Prabhu. Computing distances between reach flowpipes. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, HSCC '16, page 267276. Association for Computing Machinery, 2016.

[28] Oded Maler and Dejan Nickovic. Monitoring properties of analog and mixed-signal circuits. *STTT*, 15(3):247–268, 2013.

[29] Noushin Mehdipour, Cristian Ioan Vasile, and Calin Belta. Average-based robustness for continuous-time signal temporal logic. In *58th IEEE Conference on Decision and Control, CDC 2019*, pages 5312–5317. IEEE, 2019.

[30] Zahra Ramezani, Alexandre Donzé, Martin Fabian, and Knut Åkesson. Temporal logic falsification of cyber-physical systems using input pulse generators. In *8th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21), 2021*, volume 80 of *EPiC Series in Computing*, pages 195–202. EasyChair, 2021.

[31] Simone Schuler, Fabiano Daher Adegas, and Adolfo Anta. Hybrid modelling of a wind turbine. In *ARCH@CPSWeek 2016, 3rd International Workshop on Applied Verification for Continuous and Hybrid Systems*, volume 43 of *EPiC Series in Computing*, pages 18–26. EasyChair, 2016.

[32] Masaki Waga. Falsification of cyber-physical systems with robustness-guided black-box checking. In *HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, 2020*, pages 11:1–11:13. ACM, 2020.

[33] Zhenya Zhang, Paolo Arcaini, and Ichiro Hasuo. Constraining counterexamples in hybrid system falsification: Penalty-based approaches. In *NASA Formal Methods - 12th International Symposium, NFM 2020*, volume 12229 of *Lecture Notes in Computer Science*, pages 401–419. Springer, 2020.