Vacuity Aware Falsification for MTL Request-Response Specifications

Adel Dokhanchi*, Shakiba Yaghoubi*, Bardh Hoxha†, and Georgios Fainekos*

Abstract— **We propose a method to improve the automated test case generation for Metric Temporal Logic (MTL) falsification for Cyber-Physical Systems (CPS). In this work, we focus on request-response MTL specifications. That is, specifications that consist of at least one antecedent and a corresponding consequent. Test case generation is particularly difficult for these specifications since the consequent is only considered if the antecedent is satisfied. Therefore, we propose a method that first targets the antecedent in the specification. We show that our framework can improve upon existing falsification methods on a number of benchmark problems.**

I. Introduction

Many Cyber-Physical Systems (CPS) are encountered in safety critical applications and have strict requirements on system behavior and functional safety. Hence, it is of paramount importance to guarantee that a CPS will satisfy these requirements. The process of checking the requirement on the system is usually referred to as the verification problem for CPS. Unfortunately, in general, the verification problem for CPS is an undecidable problem. Hence, a lot of effort has been invested on bounded-time model checking (reachability analysis) and falsification methods (for an overview see [17]).

Falsification methods try to find unsafe behaviors with respect to safety specifications [3]. These methods are used to debug the CPS design during model based development (through simulations), implementation (through software-in-the-loop testing), and prototyping (through hardware-in-the-loop testing). *Request-response* requirements are very important in safety critical systems where the CPS must react to a critical event. Request-response requirements specify that every request should be followed by some response usually within some bounded time. For example, one such specification is "Every time the engine shifts from 1st to 2nd gear, then it does not shift back to 1st gear within 2.5 sec" [15]. In this case, the request is the event of shifting from 1st to 2nd gear, while the response is that the engine should not shift back to 1st gear for a bounded amount of time.

Falsification of request-response specifications is particularly difficult since the falsification method must first satisfy the antecedent and, then, falsify the consequent. Hence, it can be the case that computational effort is wasted because the generated test cases do not satisfy the request part of the specification (see for example the discussion in [11]).

In this paper, we propose a method to improve automatic test case generation for falsification of CPS with respect to request-response requirements. We consider the application of utilizing vacuity detection in testing [7] to improve the counter-example generation process. Vacuity detection is the problem of determining whether a temporal logic specification is vacuously satisfied with respect to a signal or system. Vacuity depends on the structure of a Metric Temporal Logic (MTL) [18] formula. One of the main sources of vacuity in system testing and verification is the antecedent failure in request-response requirements [9]. Request-response requirements contain at least one implication operation ($\varphi \to \psi$) which consists of an antecedent (φ) and a consequent (ψ). The system trajectories that fail to satisfy the antecedent (φ) will trivially satisfy the implication ($\to$). We refer to these system trajectories (behaviors) that trivially (vacuously) satisfy the specifications as *vacuous signals*.

Our contribution in this paper is that we have developed a framework to discover and focus the falsification process on non-vacuous signals in order to improve the counter-example generation for CPS. We call the framework *Vacuity Aware Falsification* (VAF). We have implemented our results on top of S-TaLiRo [2]. Our experimental results demonstrate that VAF achieves better falsification outcomes.

Related Work: The most related work is by Akazaki [4]. Akazaki applied Gaussian Process Regression (GPR) [8] to improve the probability of antecedent satisfaction during the falsification process using the robust semantics of Signal Temporal Logic (STL) formulas [13], [14]. The work in [4] focuses the search on the antecedent satisfaction by applying GPR to estimate the input region that most likely leads the system to satisfy the antecedent.

Our work is based on the results of our earlier work [10], [11]. We generalize the concept of antecedent failure as a subset of the signal vacuity issue [11], and we utilize the signal vacuity detection to provide an alternative solution to this problem using a two stage falsification process. Hence, our solution can benefit from various stochastic optimization techniques as we report in the experiments. Furthermore, our framework can also be applied to the systems where the robust semantics do not provide any guidance to the falsification process. For instance, this can be the case when the request in the specification is over the Boolean values $\{T, F\}$. Finally, our approach can utilize the GPR method of [4] in order to improve the probability of antecedent satisfaction in our framework.

A thorough recent review on search based falsification

*The authors are with the School of Computing, Informatics and Decision Systems Engineering, Arizona State University, Tempe, AZ, U.S.A. Email: {adokhanc,syaghoub,fainekos}@asu.edu

†The author is with the Department of Computer Science, Southern Illinois University, Carbondale, IL, U.S.A. Email: bhoxha@cs.siu.edu

This research was partially funded by NSF awards CNS-1350420, CNS-1319560 and IIP-1361926, and the NSF I/UCRC Center for Embedded Systems.

methods can be found in [17]. In our prior work [10], [11], we studied the problems of vacuous requirements and the impact of vacuous signals to the efficiency of the falsification process. However in [11], we did not discuss how to improve the falsification process which is the focus of this paper.

II. Preliminaries

We assume models of CPS are developed using Matlab Simulink/Stateflow. We intend to test models with respect to requirements (specifications) presented in Metric Temporal Logic (MTL) [18]. MTL is a well known formalism for stating real-time properties.

In this paper, we assume $\mathbb{R}$ be the set of real numbers, $\mathbb{R}_+$ is the set of non-negative real numbers, and $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$. Also, $\mathbb{N}$ is the set of natural numbers including 0. We define $\mathcal{P}(A)$ to be the power set of the set A. For testing, we set $T \in \mathbb{R}_+$ to be the maximum simulation time.

A. System Representation and Assumptions

Formally, we view a system Σ as a mapping from initial conditions $\mathcal{X}_0$, system parameters P and input signals U^R to output signals $\mathcal{Y}^R$. Here, R is defined as $R = [0, T]$, U is the set of input values (input space) and $\mathcal{Y}$ is the set of output values (output space). The following three restrictions on the system are critical in order to be algorithmically searchable over an infinite space:

1) The input signals $u \in U^R$ (if any) must be piecewise continuous defined over a finite number of intervals over $R = [0, T]$. This assumption is necessary in order to be able to parameterize the input signal space over a finite set of parameters. Thus, in the following we assume that any $u \in U^R$ of interest can be represented by a vector of parameter variables p taking values from a set P_U.
2) The output space $\mathcal{Y}$ must be equipped with a non-trivial metric. For example, the discrete metric does not provide any useful quantitative information.
3) The system Σ must be deterministic. That is, for a specific initial condition χ_0 and input signal u, there must exist a unique output signal η[1].

The previous restrictions render the system Σ to be a function $\Delta_\Sigma : \mathcal{X}_0 \times P \times P_U \rightarrow \mathcal{Y}^R$ which takes as input an initial condition vector $\chi_0 \in \mathcal{X}_0$ and two parameter vectors $p \in P$ and $p' \in P_U$, and produces as output a signal $\eta : [0, T] \rightarrow \mathcal{Y}$. Since we consider testing and/or simulation, we assume that there exists a sampling function $\tau : \mathbb{N} \rightarrow [0, T]$ that returns for each sample i its time stamp $\tau(i)$. In practice, τ is a partial function $\tau : N \rightarrow [0, T]$ with $N \subset \mathbb{N}$ and $|N| < \infty$. A *timed state sequence* or *trace* is the pair $\mu = (\eta \circ \tau, \tau)$. We will also denote $\eta \circ \tau$ by σ. The set of all timed state sequences of Σ that correspond to any sampling function τ will be denoted by $\mathcal{L}(\Sigma)$. That is, $\mathcal{L}(\Sigma) = \{(\eta \circ \tau, \tau) \mid \exists \tau \in [0, T]^N . \exists \chi_0 \in \mathcal{X}_0 . \exists p \in P . \exists p' \in P_U . \eta = \Delta_\Sigma(\chi_0, p, p')\}$. For the simplification of this paper, we define $\mu = \Sigma(\chi_0, p, u)$ to denote the trace $\mu = (\eta \circ \tau, \tau)$ as the outcome of system Σ simulation $\eta = \Delta_\Sigma(\chi_0, p, u)$ for a given sample function τ.

[1]Being deterministic is very crucial to benefit from vacuity aware falsification, since we expect the same behavior form Σ for the same input.

B. MTL Falsification Problem

Our goal is to test the system Σ with respect to an MTL formula. MTL specifications can capture system requirements by defining a set of atomic propositions AP which labels subsets of $\mathcal{Y}$ by an observation map $O : AP \rightarrow \mathcal{P}(\mathcal{Y})$ where each $\pi \in AP$ is mapped to a set $O(\pi) \subset \mathcal{Y}$[2].

Definition 1 (MTL Syntax): Assume AP is the set of atomic propositions and $\mathcal{I}$ is any non-empty interval of $\overline{\mathbb{R}}_+$. The set MTL of all well-formed MTL formulas is inductively defined as $\varphi ::= \top \mid \pi \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \bigcirc\phi \mid \phi_1 U_{\mathcal{I}} \phi_2$, where $\pi \in AP$, $\top$ is *true*, $\bigcirc$ is Next and $U_{\mathcal{I}}$ is Until operator.

For MTL formulas ψ, ϕ, we define $\psi \wedge \phi \equiv \neg(\neg\psi \vee \neg\phi)$, $\bot \equiv \neg\top$ (False), $\psi \rightarrow \phi \equiv \neg\psi \vee \phi$ (ψ Implies ϕ), $\Diamond_I \psi \equiv \top U_I \psi$ (Eventually ψ), $\Box_I \psi \equiv \neg\Diamond_I \neg\psi$ (Always ψ), and $\psi R_I \phi \equiv \neg(\neg\psi U_I \neg\phi)$ (ψ Releases ϕ) using syntactic manipulation.

Using a *metric* $\mathbf{d}$, we can define a distance function that captures how far away a point $y \in \mathcal{Y}$ is from a set $S \subseteq \mathcal{Y}$. Intuitively, the distance function assigns positive values when y is in the set S and negative values when y is outside the set S. The metric $\mathbf{d}$ must be at least a generalized quasi-metric as described in [3] which also includes the case where $\mathbf{d}$ is a metric as it was introduced in [14].

Definition 2 (Signed Distance): Assume $\sigma(i) \in \mathcal{Y}$ is a point at sample index $i \in N$, $S \subseteq \mathcal{Y}$ is a set and $\mathbf{d}$ is a metric. The Signed Distance from $\sigma(i)$ to S is defined as:

$$\mathbf{Dist_d}(\sigma(i), S) := \begin{cases} -\mathbf{dist_d}(\sigma(i), S) & \text{if } \sigma(i) \notin S \\ \mathbf{dist_d}(\sigma(i), \mathcal{Y} \backslash S)\} & \text{if } \sigma(i) \in S \end{cases}$$

where $\mathbf{dist_d}(y, S) := \inf\{\mathbf{d}(y, y') \mid y' \in S\}$ and inf is the infimum function.

To simplify the presentation, we use predicates over the system outputs instead of atomic propositions and observation maps as in STL [19]. For example, we write $\Box_{[0,10]}(speed \geq 80)$ instead of $\Box_{[0,10]}\pi$ where $O(\pi) = [80, +\infty)$. That is, $\pi \equiv (speed \geq 80)$. We can use these notations interchangeably because MTL robustness semantics [14] is equivalent to STL robustness semantics [13].

Definition 3 (MTL Robust Semantics): Consider a metric $\mathbf{d}$, trace μ and $O : AP \rightarrow \mathcal{P}(\mathcal{Y})$, then the robust semantics of any formula $\phi \in MTL$ with respect to μ at sample time $i \in N$ is recursively defined as:

$$\begin{aligned}
[\![\top]\!]_{\mathbf{d}}(\mu, i) &:= +\infty \\
[\![\pi]\!]_{\mathbf{d}}(\mu, i) &:= \mathbf{Dist_d}(\sigma(i), O(\pi)) \\
[\![\neg\phi]\!]_{\mathbf{d}}(\mu, i) &:= -[\![\phi]\!]_{\mathbf{d}}(\mu, i) \\
[\![\phi_1 \vee \phi_2]\!]_{\mathbf{d}}(\mu, i) &:= \max\left([\![\phi_1]\!]_{\mathbf{d}}(\mu, i), [\![\phi_2]\!]_{\mathbf{d}}(\mu, i)\right) \\
[\![\bigcirc\phi]\!]_{\mathbf{d}}(\mu, i) &:= \begin{cases} [\![\phi]\!]_{\mathbf{d}}(\mu, i+1) & \text{if } i+1 \in N \\ -\infty & \text{otherwise} \end{cases}
\end{aligned}$$

[2]Alternatively, instead of using symbol π from AP, we could explicitly write in the formula the predicate which defines $O(\pi)$ as it is the case in Signal Temporal Logic (STL) [19].

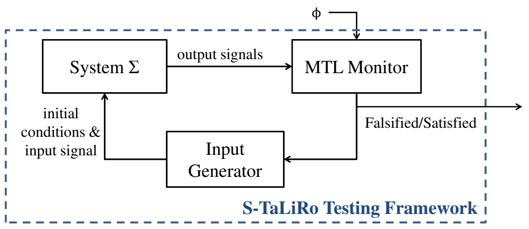

Fig. 1. Overview of S-TaLiRo testing framework for the Metric Temporal Logic (MTL) falsification problem [3], [6].

$$[\![\phi_1 U_{\mathcal{I}} \phi_2]\!]_{\mathbf{d}}(\mu, i) := \max_{j \in \tau^{-1}(\tau(i) +_R \mathcal{I})} \Big(\min\Big([\![\phi_2]\!]_{\mathbf{d}}(\mu, j), \min_{i \le k < j} [\![\phi_1]\!]_{\mathbf{d}}(\mu, k) \Big) \Big)$$

where $t +_R \mathcal{I} = \{t'' \in R \mid \exists t' \in \mathcal{I} \,.\, t'' = t + t'\}$, τ^{-1} is the inverse function of τ to extract the sample index $i \in N$, and $-$ is a unary operator defining the "negative" values of the range of $\mathbf{d}$. A trace μ satisfies an MTL formula ϕ (denoted by $\mu \models \phi$), if $[\![\phi]\!]_{\mathbf{d}}(\mu, 0) > 0$. On the other hand, a trace μ' falsifies an MTL formula ϕ (denoted by $\mu' \not\models \phi$), if $[\![\phi]\!]_{\mathbf{d}}(\mu', 0) < 0$.

Now, we introduce the falsification framework to provide the infrastructure for our proposed method.

Problem 1 (MTL Falsification): Given a system Σ and an MTL specification ϕ, the falsification problem consists of finding a trace μ of the system Σ starting from an initial state χ_0, parameter p, and an input signal u such that $\mu = \Sigma(\chi_0, p, u)$ and $\mu \not\models \phi$.

The robust semantics [13], [14] can help us to guide the search for MTL falsification [3]. In order to falsify the specification, we use the temporal logic robustness as a cost function which we attempt to minimize. Therefore, we converted the falsification problem into an optimization problem. The high level overview of the solution of the Robustness Guided Temporal Logic Falsification (TLF) problem appears in Fig. 1. The optimization algorithm generates initial conditions, and input signals. Then, the system Σ produces the output signal for which the specification robustness is evaluated by an MTL monitor [15]. The process is repeated until a maximum number of tests is reached or a falsifying behavior is detected. The framework of Fig. 1 can be implemented as a MATLAB toolbox, i.e., S-TaLiRo [6] or Breach [12].

III. Vacuity Aware Falsification Framework

To simplify the presentation, we assume that the MTL specification has only one implication operation. In order to falsify the implication operation $\phi = \psi \rightarrow \varphi \equiv \neg\psi \vee \varphi$, we need to satisfy the antecedent ψ first.

Problem 2 (Vacuity Aware Falsification): Given a system Σ and a request-response MTL specification ϕ with an implication $\psi \rightarrow \varphi$ subformula in a positive form[3], find a trace μ of the system Σ starting from an initial state χ_0, a fixed parameter p and an input signal $u = \overline{u}\underline{u}$ such that the prefix of the trace $\overline{\mu} = \Sigma(\chi_0, p, \overline{u})$ satisfies the antecedent $\overline{\mu} \models \psi$[4] and the whole trace $\mu = \Sigma(\chi_0, p, \overline{u}\underline{u})$ falsifies the main MTL formula $\mu \not\models \phi$.

[3]Without any negation in the parent nodes of $\psi \rightarrow \varphi$ in ϕ's parse tree.
[4]Traces that satisfy the antecedent are called non-vacuous traces [11].

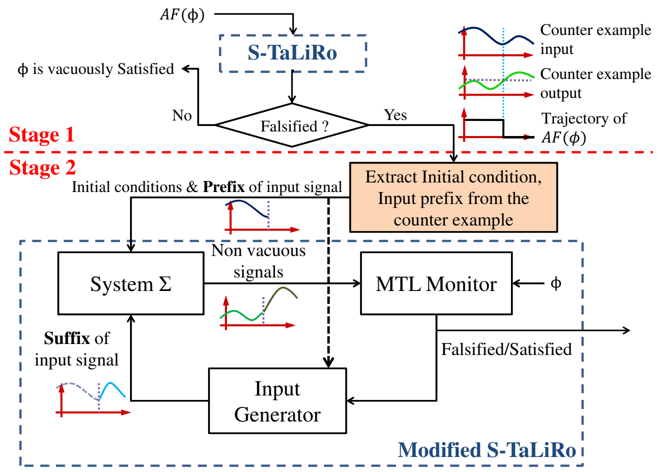

Fig. 2. Proposed flow for Vacuity Aware Falsification.

A. Proposed Solution

Our strategy for Vacuity Aware Falsification is a two stage solution: 1) The falsification process should first satisfy the antecedent. 2) For the traces that satisfy the antecedent, the process will guide the system toward falsifying the consequent. The proposed flow is provided in Fig. 2. To address the first stage, we create the Antecedent Failure (AF) formula $AF(\phi)$ where its falsification is interpreted as satisfaction of the antecedent. The AF in $AF(\phi)$ is a function that when given the formula $\phi = \Box_I(\psi \rightarrow \varphi)$, it returns $\Box_I(\neg\psi)$. The algorithm that extracts $AF(\phi)$ from a more complex MTL formula ϕ is provided in [11]. The $AF(\phi) = \Box_I(\neg\psi)$ formula asserts that the antecedent ψ would never happen in the time window of I, see [11] for more details. If S-TaLiRo falsifies $AF(\phi)$, it means that the antecedent ψ has eventually been satisfied. According to the architecture in Fig. 2, our proposed flow runs the testing framework in two stages:

Stage 1: We try to falsify $AF(\phi)$ using the falsification framework. If the $AF(\phi)$ is falsified during Stage 1, it means that the antecedent has eventually been satisfied by $\overline{\mu}$. Thus, we can proceed to Stage 2 to falsify the main formula ϕ. Otherwise, ϕ is vacuously satisfied in this run.

Stage 2: Since $AF(\phi)$ is falsified (in Stage 1), the counter example is the system input u that leads the system to create trajectories that satisfy the antecedent of the specification. Now, we should extract the shortest prefix of the input (denoted as $\overline{u}$) that leads the system to just falsify $AF(\phi)$ and immediately stop the simulation at the falsification point. The input prefix $\overline{u}$ leads the system to create the $\overline{\mu}$ trace. We should choose the input prefix $\overline{u}$ as short as possible to increase the search space to help the input generator to find the best suffix (input $\underline{u}$) that may lead the system to falsify ϕ. Now, we can explain why the system Σ should be deterministic. This is because in Stage 2 we expect to create the same satisfying output $\overline{\mu}$ for the same input prefix $\overline{u}$ that is extracted from Stage 1.

In Stage 2, we fix the initial condition χ_0, parameter p and input prefix $\overline{u}$ which forces all the new testing trajectories to become *non-vacuous signals*. Recall that non-vacuous signals are the signals that satisfy the antecedent. As a result, the input generator will search over the suffix of the input $\underline{u}$ for the system to find a non-vacuous signal $\mu = \Sigma(\chi_0, p, \overline{u}\underline{u})$ that will eventually falsify the main formula.

The high-level pseudo code of the algorithm that corresponds to Fig. 2 is provided in Algorithm 1, where opt and opt' are the optimizers of choice, and N_{MAX} is the upper-limit for the number of optimizer's iterations. In Line 2, we run S-TaLiRo to falsify $\phi_{AF} = AF(\phi)$ (Stage 1). S-TaLiRo returns χ_0, p, u correspond to the minimum robustness. If the search is successful, we move to Stage 2, unless we report that ϕ is vacuously satisfied (Line 15). In Stage 2, we extract the input prefix $\overline{u}$ in Line 6 and run S-TaLiRo to find the falsifying suffix (Line 7). In Line 7, $(\overline{u}, U)$ is the input space with fixed prefix $\overline{u}$. S-TaLiRo in Stage 2 searches over the suffixes of the input signal to find the trajectory μ' that falsifies the specification until the number of tests of opt' reaches to N_ϕ.

Finally, we report the falsification results in Lines 10 and 12. Here, we need to remark that $\overline{\mu} \not\models \phi_{AF}$ does not guarantee that there exists a μ' such that $\mu' \not\models \phi$.

Algorithm 1 Vacuity Aware Falsification

```
Input: $\Sigma, P, \mathcal{X}_0, U, \phi, opt, opt', N_{MAX}$;
Output: Message about Falsification Report;
Procedure VAF($\Sigma, P, \mathcal{X}_0, U, \phi, opt, N_{MAX}$)
 1: $\phi_{AF} \leftarrow AF(\phi)$
 2: $[\chi_0, p, u, N_{AF}] \leftarrow$ S-TaLiRo($\Sigma, P, \mathcal{X}_0, U, \phi_{AF}, opt, N_{MAX}$)
 3: $\mu \leftarrow \Sigma(\chi_0, p, u)$ ; $N_\phi \leftarrow N_{MAX} - N_{AF}$
 4: if $\mu \not\models \phi_{AF}$ then
 5:     Extract $\overline{\mu} \subset \mu$ such that $\overline{\mu} \not\models \phi_{AF}$
 6:     Extract $\overline{u} \subset u$ such that $\overline{\mu} = \Sigma(\chi_0, p, \overline{u})$
 7:     $[\chi_0, p, u', N_f] \leftarrow$ S-TaLiRo($\Sigma, p, \chi_0, (\overline{u}, U), \phi, opt', N_\phi$)
 8:     $\mu' \leftarrow \Sigma(\chi_0, p, u')$
 9:     if $\mu' \not\models \phi$ then
10:         return "$\phi$ is falsified"
11:     else
12:         return "$\phi$ is NOT falsified"
13:     end if
14: else
15:     return "$\phi$ is vacuously satisfied!"
16: end if
```

B. Input Prefix-Suffix Example

An example for extracting the input prefix $\overline{u}$ from input u is depicted in Fig. 3. Consider the following specification $\phi = \Box_{[0,t_1]}(a \rightarrow \Diamond_{[0,t_2]} b)$ where $a \equiv v > 80$ and $b \equiv v < 60$, which formalizes the following natural language requirement:

"Always during the simulation time up to t_1 seconds, if the speed (v) goes above 80, then it must eventually drop below 60 in t_2 seconds"

Figure 3 represents the system input and trajectory corresponding to the formula ϕ. In Fig. 3, the system input u (Throttle) and the system output v (Speed) are presented. Any system trace μ that falsifies ϕ must first satisfy the precondition of ϕ. In other words, its prefix $\overline{\mu}$ must falsify the antecedent failure, namely $AF(\phi) = \Box_{[0,t_1]}\neg(a) = \Box_{[0,t_1]}(v \leq 80)$. The system trajectory in Fig. 3 is a falsifying signal for

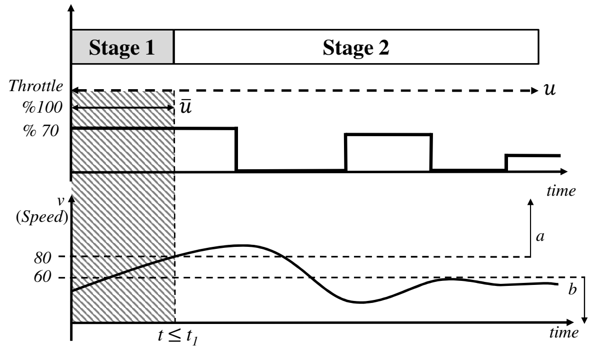

Fig. 3. Stage 1 (Gray) and Stage 2 (White) of the Vacuity Aware Falsification.

the antecedent failure $\Box_{[0,t_1]}(v \leq 80)$. Therefore, the trajectory in Fig. 3 is a non-vacuous signal since $v > 80$. The entire duration of input signal u is represented by a dashed line which contains the whole throttle schedule. The shortest prefix of the input signal $\overline{u}$ that leads the system to $v > 80$ is represented with a hashed box.

IV. Experiments

In this section, we consider the application of our proposed method to improve the performance of the falsification method. Our experiments were conducted on a 64-bit Intel Xeon CPU (2.5GHz) with 64-GB RAM and Windows Server 2012. We used MATLAB 2015a to run the falsification toolbox S-TaLiRo [2] and to implement our method (Fig. 2 and Algorithm 1). For our experiments, we used the following stochastic optimization methods: Simulated Annealing (SA) [3], Cross-Entropy (CE) optimization [20], and Uniform Random (UR) sampling. We remark that all the experiments were performed with the default parameters for each optimization method. All the benchmark problems are available with the S-TaLiRo distribution [2] or from the ARCH workshop repository [1].

A. Navigation Benchmark with Inputs

We consider a version of the Navigation Benchmark proposed by Fehnker and Ivancic [16] with a few modifications. The Navigation Benchmark is a four continuous-state autonomous affine hybrid automaton. We refer the reader to [5] for an introduction to hybrid automata. The primary modification is that now we allow for external inputs to the system (beyond the constant affine term in the original model). Even though affine hybrid systems can now be efficiently solved using reachability tools for hybrid systems, it still remains a challenge to verify request-response requirements as expressed in MTL. In addition, we remark that for this benchmark, the affine dynamics in each mode could be changed to complex smooth non-linear dynamics without any impact to the applicability of the proposed methodology.

The benchmark studies a hybrid automaton $\mathcal{H}$ with a variable number of discrete locations and 4 continuous state variables x_1, x_2, x_3 and x_4 that form the state vector $x = [x_1\ x_2\ x_3\ x_4]^T$. The structure of the hybrid automaton can be better visualized in Fig. 4. The hybrid automaton has a

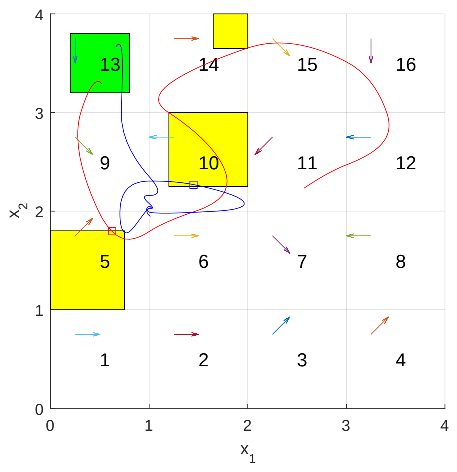

Fig. 4. Modified Navigation benchmark with 16 locations (modes): Two trajectories falsifying the requirements ϕ_{NB_1}, and ϕ_{NB_2}.

number of modes (16 in the example of Fig. 4) where in each mode, the dynamics of the system are different.

In detail, in each location i of the hybrid automaton, the system evolves under the differential equation

$$\dot{x} = Ax - Bv(i) + Cu \tag{1}$$

where u is a 2 dimensional external continuous input to the system (in this benchmark for all time t, $u(t) \in [-5, 5]^2$), the matrices A, B and C are defined as

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1.2 & 0.1 \\ 0 & 0 & 0.1 & -1.2 \end{bmatrix}, \ B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1.2 & 0.1 \\ 0.1 & -1.2 \end{bmatrix} \ \text{and} \ C = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0.5 \\ 0 & 1 \end{bmatrix}$$

and the constant vector term in each location is

$$v(i) = [\sin(\pi D(i)/4) \ \cos(\pi D(i)/4)]^T.$$

The array D is one of the parameters of the hybrid automaton that the user can control in order to define different benchmarks. It defines the input vector in each discrete location (see arrows in Fig. 4).

The invariant set of every location (mode) is a 1×1 box that constraints the "position" of the system (x_1, x_2), while the velocity (x_3, x_4) can flow unconstrained. The guards in each location are the edges and the vertices that are common among the neighboring locations. When a guard is reached, the system switches between system dynamics. The set of initial conditions is the set to $H_0 = \{(m, x) \mid m = 13, x \in [0.2\ 0.8] \times [3.2\ 3.8] \times [-0.4\ 0.4]^2\}$ (green box in Fig. 4).

Sample trajectories (under some input signal) of the system appear in Fig. 4 for initial conditions $(0.6821, 3.6558, 0.0685, -0.1790)$ for the blue trajectory and $(0.4136, 3.2076, -0.3705, 0.3474)$ for the red trajectory.

We evaluated the following request-response requirements on the system (the sets which correspond to each predicate in the formulas are highlighted as yellow boxes in Fig. 4):

$$\begin{aligned}\phi_{NB_1} =& \Box((i = 10 \wedge x_1 \geq 1.2 \wedge x_2 \geq 2.25) \\ & \rightarrow \Box\neg(i = 5 \wedge x_1 \leq 0.75 \wedge x_2 \leq 1.8)) \\ \phi_{NB_2} =& \Box((i = 5 \wedge x_1 \leq 0.75 \wedge x_2 \leq 1.8) \\ & \rightarrow \Box\neg(i = 14 \wedge x_1 \geq 1.65 \wedge x_2 \geq 3.65))\end{aligned}$$

Both specifications state "if a set X is visited, then from that point on a set Y should not be visited". Variations of these requirements with timing constraints can be easily constructed. Since the predicates in ϕ_{NB_i} represent hybrid space (discrete locations with continuous state variables) we need to use hybrid distance semantics for the robustness semantics (see the generalized distance function $\mathbf{d_h}$ in [3]). Finally, we set $N_{MAX} = 200$ for Algorithm 1.

The results for both formulas are presented in Table I (ϕ_{NB} rows). All the experiments are conducted with the same number of optimization's tests (N_{MAX}) for both VAF and S-TaLiRo. The following observations can be made. First and foremost, using VAF uniformly improves the falsification outcomes independently of what the underlying method is. In all cases, by utilizing VAF, the rate of detecting falsifying behaviors is at least doubled. Second, on harder problem instances, i.e., for specification ϕ_{NB_2}, the VAF method outperforms the methods without VAF by an order of magnitude. In general, the difficulty of a benchmark can be assessed by how easily it is falsified using uniform random sampling.

B. *Automatic Transmission*

The Automatic Transmission (AT) model is provided by Mathworks as a Simulink demo[5]. The AT has two inputs: Throttle and Brake. The throttle and break can take any value between 0% to 100%, at each point in time. The outputs contain two real-valued traces: the speed of the engine ω and the speed of the vehicle v. In addition, the outputs contain one discrete-valued trace *gear* with four possible values. Thus, AT is a Simulink model that exhibits both continuous and discrete behavior. In order to evaluate the improvement of S-TaLiRo framework by using VAF, we considered the following safety request-response requirements:

1) "After shifting down into gear one, there should be no shift from gear one to any other gear within 2.5 sec."
2) "After shifting down into gear one, the engine speed ω should always stay below 3000 RPM within 2.5 sec."

The simulation time for the system is set to 30 seconds. Therefore, we can use bounded MTL formulas for the requirement such that the horizon of MTL formula equals to the simulation time (30 seconds). We formalize the above requirements as the follows:

$$\phi_{AT_1} = \Box_{[0,27.5]}((\neg g_1 \wedge \bigcirc g_1) \rightarrow \Box_{(0,2.5]} g_1)$$

$$\phi_{AT_2} = \Box_{[0,27.5]}((\neg g_1 \wedge \bigcirc g_1) \rightarrow \Box_{(0,2.5]} r_1)$$

where $g_1 \equiv \{gear = 1\}$ and $r_1 \equiv \{\omega \leq 3000\}$.

For the AT experiments, we set the number of optimization's tests to be 1000 ($N_{MAX} = 1000$). In addition, for VAF, we create the antecedent failure of ϕ_{AT_1} and ϕ_{AT_2} as follows:

$$AF(\phi_{AT}) = \Box_{[0,27.5]}(\neg(\neg g_1 \wedge \bigcirc g_1))$$

For evaluating VAF, we first setup the S-TaLiRo to falsify the $AF(\phi_{AT})$ which is the execution of Stage 1 in Fig. 2. Then,

[5]Modeling an Automatic Transmission Controller, Available at: `http://www.mathworks.com/help/simulink/examples/modeling-an-automatic-transmission-controller.html`

TABLE I

Comparing Vacuity Aware Falsification (VAF) with Temporal Logic Falsification (TLF) for the falsification of ϕ_{NB}, ϕ_{AT}.

	Vacuity Aware Falsification (VAF)			S-TaLiRo	
Spec.	Opt.	AF(Spec.) is falsified (Stage 1)	Spec. is falsified (Stage 2)	Opt.	falsified
ϕ_{NB_1}	UR	100/100	88/100	UR	32/100
ϕ_{NB_1}	SA	91/100	59/91	SA	21/100
ϕ_{NB_1}	CE	100/100	67/100	CE	26/100
ϕ_{NB_2}	UR	100/100	10/100	UR	1/100
ϕ_{NB_2}	SA	92/100	23/92	SA	1/100
ϕ_{NB_2}	CE	100/100	24/100	CE	2/100
ϕ_{AT_1}	UR	97/100	97/97	UR	20/100
ϕ_{AT_2}	UR+SA	95/100 (UR)	95/95 (SA)	SA	17/100
ϕ_{AT_2}	UR+CE	91/100 (UR)	91/91 (CE)	CE	8/100

we run the second stage of VAF if Stage 1 was successful (see Fig. 2 Stage 2).

The falsification results of our proposed method are provided in Table I (ϕ_{AT_i} rows). It can be observed that for ϕ_{AT_1} the original UR method can successfully falsify only 20 out of 100 runs. However, our method successfully falsified the antecedent failure in 97 out of 100 runs in Stage 1, and among the runs that successfully falsify $AF(\phi_{AT})$, all of them would ultimately falsify the original specification in Stage 2.

For the rows corresponding to ϕ_{AT_2}, we choose UR for the falsification at Stage 1. This is due to the fact that the hybrid robustness value of g_1 is equivalent to $\top$ when $gear = 1$, and $\perp$ when $gear \neq 1$ with no intermediate values between them (see the generalized distance function $\mathbf{d_h^0}$ in [3]). Therefore, the cost function of the stochastic optimizer does not decrease towards the falsification. In this case, since g_1 behaves like a Boolean event, UR is the preferred optimization algorithm in Stage 1. This demonstrates the flexibility of our method in that we can choose different optimizations for different Stages of VAF.

For the second stage of ϕ_{AT_2}, we used SA and CE. The VAF method with using UR+SA improves the falsification for SA-TaLiRo as follows: The original SA-TaLiRo successfully falsifies 17 out of 100 runs. We used UR-TaLiRo in Stage 1 to falsify antecedent failure in 95 out of 100 runs and SA-TaLiRo used those signal prefixes to falsify all of the runs in Stage 2. CE-TaLiRo improves the results in a similar way. Our experiments on AT show that VAF with UR-TaLiRo in Stage 1, can drastically improve the falsification process.

V. Conclusions

We have introduced a new framework for Vacuity Aware Falsification (VAF) for Cyber-Physical Systems (CPS). Our experimental results demonstrate improvements for different S-TaLiRo optimization methods when we apply our new VAF framework. In the future, this method will be applied to more complex request-response requirements with more than one implication operations.

References

[1] Applied Verification for Continuous and Hybrid Systems (ARCH) `http://cps-vo.org/group/ARCH`.

[2] S-TaLiRo : `https://sites.google.com/a/asu.edu/s-taliro/`.

[3] H. Abbas, G. Fainekos, S. Sankaranarayanan, F. Ivančić, and A. Gupta. Probabilistic temporal logic falsification of cyber-physical systems. *ACM Trans. Embed. Comput. Syst.*, 12(2s):95:1–95:30, May 2013.

[4] T. Akazaki. Falsification of conditional safety properties for cyber-physical systems with gaussian process regression. In *Runtime Verification - 16th International Conference, RV 2016, Madrid, Spain, September 23-30, 2016, Proceedings*, pages 439–446, 2016.

[5] R. Alur. *Principles of Cyber-Physical Systems*. MIT Press, 2015.

[6] Y. S. R. Annapureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan. S-TaLiRo: A tool for temporal logic falsification for hybrid systems. In *Tools and algorithms for the construction and analysis of systems*, volume 6605 of *LNCS*, pages 254–257. Springer, 2011.

[7] T. Ball and O. Kupferman. Vacuity in testing. In *Tests and Proofs, Second International Conference, TAP 2008, Prato, Italy, April 9-11, 2008. Proceedings*, pages 4–17, 2008.

[8] E. Bartocci, L. Bortolussi, L. Nenzi, and G. Sanguinetti. System design of stochastic models using robustness of temporal properties. *Theor. Comput. Sci.*, 587:3–25, 2015.

[9] S. Ben-David, D. Fisman, and S. Ruah. Temporal antecedent failure: Refining vacuity. In *CONCUR 2007 - Concurrency Theory, 18th International Conference, CONCUR 2007, Lisbon, Portugal, September 3-8, 2007, Proceedings*, pages 492–506, 2007.

[10] A. Dokhanchi, B. Hoxha, and G. Fainekos. Metric interval temporal logic specification elicitation and debugging. In *13. ACM/IEEE International Conference on Formal Methods and Models for Codesign, MEMOCODE 2015, Austin, TX, USA, September 21-23, 2015*, pages 70–79, 2015.

[11] A. Dokhanchi, B. Hoxha, and G. Fainekos. Formal requirement elicitation and debugging for testing and verification of cyber-physical systems. *CoRR*, abs/1607.02549, 2016.

[12] A. Donze. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification*, volume 6174 of *LNCS*, pages 167–170. Springer, 2010.

[13] A. Donze and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In *Formal Modelling and Analysis of Timed Systems*, volume 6246 of *LNCS*. Springer, 2010.

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

[15] G. Fainekos, S. Sankaranarayanan, K. Ueda, and H. Yazarel. Verification of automotive control applications using S-TaLiRo. In *Proceedings of the American Control Conference*, 2012.

[16] A. Fehnker and F. Ivancic. Benchmarks for hybrid systems verification. In *Hybrid Systems: Computation and Control*, volume 2993 of *LNCS*, pages 326–341. Springer, 2004.

[17] J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito, and K. Butts. Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques. *IEEE Control Systems Magazine*, 36(6):45–64, 2016.

[18] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[19] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Proceedings of FORMATS-FTRTFT*, volume 3253 of *LNCS*, pages 152–166, 2004.

[20] S. Sankaranarayanan and G. Fainekos. Falsification of temporal properties of hybrid systems using the cross-entropy method. In *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '12, pages 125–134, New York, NY, USA, 2012. ACM.