Falsification of Temporal Logic Requirements Using Gradient Based Local Search in Space and Time

Shakiba Yaghoubi, and Georgios Fainekos

School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, AZ, USA
Email: {syaghoub, fainekos}@asu.edu

Abstract: We study the problem of computing input signals that produce system behaviors that falsify requirements written in temporal logic. We provide a method to automatically search for falsifying time varying uncertain inputs for nonlinear and possibly hybrid systems. The input to the system is parametrized using piecewise constant signals with varying switch times. By applying small perturbations to the system input in space and time, and by using gradient descent approach, we try to converge to the worst local system behavior. The experimental results on non-trivial benchmarks demonstrate that this local search can significantly improve the rate of finding falsifying counterexamples.

Keywords: Cyber-physical systems, Testing, Gradient Descent, Optimization, Temporal Logic.

1. INTRODUCTION

Autonomous automobile systems, industrial and medical robots, smart grids, and process control systems are all examples of Cyber-Physical Systems (CPS) that are widely used these days. All such systems have strict performance and safety requirements. Hence analysis of Cyber-Physical Systems' requirements is an important and challenging problem at the same time. Despite all the advances in exhaustive system verification methods, the resulting tools are not applicable in real-world-size problems: they either work on small problem instances or on restricted classes of systems (e.g, linear systems).

As a result testing methods are still primarily used in the industry. To improve the testing stage performance an automated framework to test the system by searching over the space of uncertain inputs becomes necessary. To decrease the necessary time and cost of the testing phase, we need to increase the efficiency of the testing process by optimizing the search. Since testing a system over continuous time input signals is a search over an infinite dimensional space, input parametrization has a significant importance.

Piecewise constant inputs are used for input parameterization in Yaghoubi and Fainekos (2017a) to falsify the specifications of smooth nonlinear gray-box systems and in Yaghoubi and Fainekos (2017b) to falsify the requirements of the whitebox hybrid systems. To simplify the analysis, in these works the input parameters are evenly spaced so that the time points at which the input signal value changes are constant. This input parameterization can assist in capturing the transient properties related to the input signal amplitude. However, since some of the system unsafe behaviors are related to temporal changes, temporal parameterization of the input signal seems to be necessary, as well.

Our main contribution in this paper is that we provide a space-time parameterization for the time varying inputs to the system, and use gradient descent (GD) to adaptively change the amplitude and the switch time of the signal to find optimal cases to be tested against system specifications. Because of the complexities that arise in the sensitivity calculation for hybrid systems (see Donzé and Maler (2007)), and to avoid unnecessary technicalities, we develop our method assuming that the system dynamics are smooth and nonlinear. However, the extension of the results in this paper to hybrid systems is easily achievable with the analysis in Yaghoubi and Fainekos (2017b), along these lines, even though, we do not present the theory here, one of the studied benchmarks in the paper which is studied in section 6.2 is also a hybrid system. We also show that combining stochastic global search methods (here we used Simulated Annealing) with the local search using GD directions can significantly increase the rate of finding unsafe behaviors. In one of the experimental studies while Simulated annealing was not able to find any unsafe behavior in 50 runs, its combination with GD was able to find an unsafe behavior in 11 runs.

2. PRELIMINARIES

We consider dynamical systems of the form

$$\Sigma : \quad \dot{x} = F(x, u) \tag{1}$$

where $x \in X \subseteq R^n$ is the system state, $u \in U^{[0,T]} \subseteq L_2^{[0,T]}$ is the input function that given a specific point in time $t \in [0, T]$ returns a value in the bounded set $U \subset R$ ($u : t \to U$), and $F : R^n \times U^{[0,T]} \to R^n$ is a C^1 flow. We assume that

⋆ This work was partially supported by the NSF awards CNS 1350420, IIP-1361926, and the NSF I/UCRC Center for Embedded Systems.

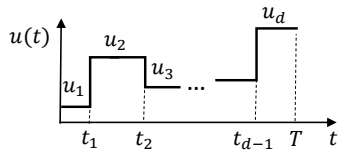

Fig. 1. Piecewise constant input parameterization

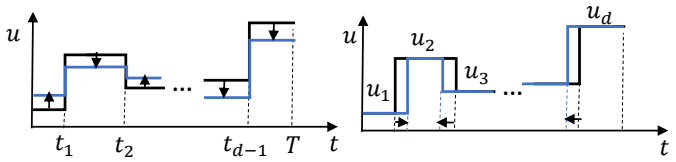

Fig. 2. Input perturbation: The left plot shows a perturbation in space (δu_u) and the right plot shows one in time (δu_t).

the system can be initiated from a set of initial conditions $X_0 \subseteq X$. Even though a closed form solution for (1) cannot be found in most of the cases, we can find a numerical solution to it assuming that the simulation time T is bounded. Given an initial condition $x_0 \in X_0$, and an input $u \in U^{[0,T]}$ we denote this solution at time t with $s(x_0, u, t)$. We indicate a piecewise constant input u consisting of d pieces using a sequence $(u_1, t_1, u_2, t_2, ..., t_{d-1}, u_d)$ where $0 < t_1 < ... < t_{d-1} < t_d = T$ and $u_1, ..., u_d \in U$. This input is described below and it is shown in Fig. 1.

$$u(t) = u_i, \text{ for } t \in [t_{i-1}, t_i], i = 1, ..., d \tag{2}$$

System (1) can be perturbed around its solution $s(x_0, u, t)$, if we make a small change δx_0 in its initial condition or δu in its input. We assume that $x_0 + \delta x_0 \in X_0$ and $u + \delta u \in U^{[0,T]}$. The change in the system solution is indicated as $\delta s(x_0, u, t)$ and it is equal to $s(x_0 + \delta x_0, u + \delta u, t) - s(x_0, u, t)$.

A perturbation $\delta u(t)$ in a piecewise constant input can be the result of small changes $(\delta u_1, ..., \delta u_d)$ in the values of $(u_1, ..., u_d)$ denoted by δu_u or small changes $(\delta t_1, ..., \delta t_{d-1})$ in the values of $(t_1, ..., t_{d-1})$ denoted by δu_t (see Fig. 2).

System requirements are usually expressed formally as Metric or Signal Temporal Logic (TL) formulas (abbreviated as MTL and STL, respectively, see the survey in Bartocci et al. (2018) for more information). TL formulas are built by combining *atomic propositions* or *predicates* using logical and temporal operators. The robustness of a system trajectory with respect to a TL formula ϕ shows how well it satisfies the specification: positive values indicate satisfaction and negative values indicate violation. The greater the robustness absolute value, the farther the system trajectory is from being satisfied/falsified (see Fainekos and Pappas (2009) for details on how the robustness is defined and calculated). As a result in a guided search for a system unsafe behavior, the algorithm should actively try to decrease the robustness value.

3. PROBLEM FORMULATION

Given a system as described in Eq. (1) and an initial condition and input to the system which determines a solution $s(x_0, u, t)$ with the robustness value r w.r.t. a system requirement ϕ, we would like to find δx_0 and δ_u such that $s(x_0 + \delta x_0, u + \delta u, t)$ has a robustness value $r_p < r$ w.r.t ϕ. Due to the definition of robustness of a trajectory $s(x_0, u, t)$ w.r.t a formula ϕ the absolute value of the robustness corresponds to the distance between a point $s(x_0, u, t^*_{x_0,u})$ on the trajectory and a point $z^*_{x_0,u}$ that belongs to one of the sets which corresponds to a predicate in ϕ. This set is called the critical set w.r.t $s(x_0, u, t)$ and $t^*_{x_0,u}$ is called the critical time. As a result, finding a neighboring trajectory of $s(x_0, u, t)$ with a robustness value smaller than r, can be reduced to minimizing the following local cost function w.r.t x_0 and u. This is shown in Yaghoubi and Fainekos (2017b).

$$J_{x_0,u}(x'_0, u') = ||s(x'_0, u', t^*_{x_0,u}) - z^*_{x_0,u}|| \tag{3}$$

Note that we are actually interested in minimizing the robustness function. However, this function is a complicated, non-smooth, and non-convex function which is hard to deal with specially since the search space is usually large dimensional. As a result, we try to minimize the cost in Eq. (3) which locally minimizes the robustness function. Focusing on the minimization of Eq. 3 has advantages over searching for the minimizer of the robustness function directly: $J_{x_0,u}$ is smooth and differentiable and using gradients we can find directions of improvement to guide the search in the large dimensional search space. The problem of our interest is stated below:

Problem 3.1. Given the system of Eq. (1), a compact time interval $[0, T]$, a set of initial conditions X_0, an input set U, a point $x_0 \in X_0$ and an input $u(t) \in U^{[0,T]}$ such that the system trajectory satisfies $0 < J_{x_0,u}(x_0, u)$, find vectors $\delta x_0, \delta u_u, \delta u_t$ that satisfy the following property:

$\exists \Delta_1, \Delta_2$ such that $\forall h_1 \in (0, \Delta_1), h_2 \in (0, \Delta_2)$: $J_{x_0,u}(x'_0, u') \leq J_{x_0,u}(x_0, u)$ where $x'_0 = x_0 + h_1 \delta x_0 \in X_0$ and $u'(t) = u(t) + h_2 \delta u(t) \in U^{[0,T]}$ and $\delta u(t)$ is the change in the input as a result of applying δu_u and/or δu_t to the input.

4. TIME-SPACE GRADIENT DESCENT

As mentioned in Yaghoubi and Fainekos (2017b), to find the solution to Problem 3.1, note that:

$$dJ_{x_0,u}(x_0, u) = \frac{\partial J_{x_0,u}}{\partial s} \delta s(x_0, u, t^*_{x_0,u}) = -n_s^T \delta s(t^*_{x_0,u}) \tag{4}$$

where $n_s = z^*_{x_0,u} - s(x_0, u, t^*_{x_0,u})$ and $\delta s(t) \triangleq \delta s(x_0, u, t)$. Using chain rule:

$$\delta s(t) = \frac{\partial s}{\partial x_0} \delta x_0 + \Sigma_{i=1}^{d} \frac{\partial s}{\partial u_i} \delta u_i + \Sigma_{i=1}^{d-1} \frac{\partial s}{\partial t_i} \delta t_i \tag{5}$$

where $\frac{\partial s}{\partial p}$ is the trajectory sensitivity to parameter p which will be calculated later. Observe that using $c_{x_0}, c_{u_i}, c_{t_i} > 0$ the following directions are descent w.r.t $J_{x_0,u}$:

$$\begin{aligned} \delta x_0 &= c_{x_0} \frac{\partial s}{\partial x_0}^T n_s \\ \delta u_i &= c_{u_i} \frac{\partial s}{\partial u_i}^T n_s, \quad i = 1, ..., d \\ \delta t_i &= c_{t_i} \frac{\partial s}{\partial t_i}^T n_s, \quad i = 1, ..., d-1 \end{aligned} \tag{6}$$

So we just need to calculate $\frac{\partial s}{\partial x_0}$ (sensitivity to initial condition), $\frac{\partial s}{\partial u_i}$ (sensitivity to the input values) and $\frac{\partial s}{\partial t_i}$ (sensitivity to the input switch times). Calculating $\frac{\partial s}{\partial x_0}$, $\frac{\partial s}{\partial u_i}$ has been shown before in Abbas et al. (2014) and Yaghoubi and Fainekos (2017b) and will be reviewed here, but we will show how to calculate $\frac{\partial s}{\partial t_i}$ here for the first time.

4.1 Sensitivity analysis

Sensitivity to initial condition: By perturbing the system around its initial conditions and using Newton's method one can easily see that the dynamics of $\frac{\partial s}{\partial x_0}$ follows

$$\frac{d}{dt}(\frac{\partial s}{\partial x_0}) = D_1F\big|_{s(x_0,u)}\frac{\partial s}{\partial x_0}, \quad \frac{\partial s}{\partial x_0}(0) = I_{n\times n} \tag{7}$$

where $I_{n\times n}$ is an identity matrix and D_i denotes the partial differentiation w.r.t the i-th element.

Sensitivity to input value: Since the input value $u_i, \forall i = 1,..,d$ is only activated in the interval $[t_{i-1}, t_i)$ with $t_0 = 0, t_d = T$, the dynamics of $\frac{\partial s}{\partial u_i}$ follows

$$\frac{\partial s}{\partial u_i}(0) = 0$$
$$\frac{d}{dt}(\frac{\partial s}{\partial u_i}) = \begin{cases} D_1F\big|_{s(x_0,u)}\frac{\partial s}{\partial u_i} + D_2F\big|_{s(x_0,u)}, & \text{if } t \in [t_{i-1}, t_i) \\ D_1F\big|_{s(x_0,u)}\frac{\partial s}{\partial u_i} & \text{otherwise} \end{cases} \tag{8}$$

Defining $\frac{\partial s}{\partial u} \triangleq [\frac{\partial s}{\partial u_1}, ..., \frac{\partial s}{\partial u_d}]$, the dynamics of $\frac{\partial s}{\partial u}$ can be written in closed form as

$$\frac{\partial s}{\partial u}(0) = 0$$
$$\frac{d}{dt}(\frac{\partial s}{\partial u}) = D_1F\big|_{s(x_0,u)}\frac{\partial s}{\partial u} + D_2F\big|_{s(x_0,u)}h(t), \text{ if } t \in [t_{i-1}, t_i) \tag{9}$$

where $h(t) = [h_1(t), ..., h_d(t)]$ and

$$h_i(t) = \begin{cases} 1 & \text{if } t \in [t_{i-1}, t_i) \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Sensitivity to input switch time: To calculate $\frac{\partial s}{\partial t_i}$, recall that this is the change in the solution as a result of a change in the switch time t_i denoted as δt_i. Assuming $\delta t_i > 0, \forall i = 1, ..., d-1$, for any $t < t_i$, $\frac{\partial s}{\partial t_i}(t) = 0$. Since δt_i is assumed to be small in value, for any $t \in [t_i, t_i + \delta t_i)$, $\delta s(t) = s(x_0, u', t) - s(x_0, u, t) = (F(x, u_i) - F(x, u_{i+1}))\delta t_i$, where u' is a result of changing t_i to $t_i + \delta t_i$ in u. So:

$$\frac{\partial s}{\partial t_i} \approx F(x, u_i) - F(x, u_{i+1}) \tag{11}$$

For any $t > t_i$, $\frac{\partial s}{\partial t_i}$ evolves as $\frac{d}{dt}(\frac{\partial s}{\partial t_i}) = D_1F\big|_{s(x_0,u)}\frac{\partial s}{\partial t_i}$. So the dynamics of $\frac{\partial s}{\partial t_i}$ can be approximated as:

$$\begin{cases} \dfrac{\partial s(t)}{\partial t_i} = 0 & \text{if } t < t_i \\ \dfrac{\partial s(t_i)}{\partial t_i} = F(x, u_i) - F(x, u_{i+1}) & \text{if } t = t_i \\ \dfrac{d}{dt}(\dfrac{\partial s}{\partial t_i}) = D_1F\big|_{s(x_0,u)}\dfrac{\partial s}{\partial t_i}, & \text{if } t > t_i \end{cases} \tag{12}$$

Since there is a reset in the value of the $\frac{\partial s(t)}{\partial t_i}$, we can use a hybrid automaton (HA) to evaluate it. Considering $S_t = [S_{t_1}, ...S_{t_{d-1}}] = [\frac{\partial s}{\partial t_1}, ...\frac{\partial s}{\partial t_{d-1}}]$, this is shown in Fig. 3.

Note that the time-space gradient descent calculation can be readily combined with the results in our paper Yaghoubi and Fainekos (2017b) to calculate descent directions for hybrid systems.

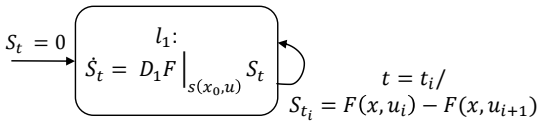

Fig. 3. HA for calculating the sensitivity to the input switch times

The space-time gradient descent framework is described in Algorithm 1. The function SIMUL&SENS simulates the system Σ and stores the numerical solution in $s_{x_0,u}$ and calculates the sensitivity functions using Eq. (7), (9) and (12) and stores them in $\partial s/\partial p$. The function GETROB returns the robustness value of the trajectory with respect to the specification along with the critical time and approach vector information. The function INBOX shrinks the new initial conditions and input to fit in X_0, U. And finally the function GD calculates the descent directions w.r.t Eq. 6.

5. A DISCUSSION ON THE CHOICE OF INPUT PARAMETERIZATION

Despite their simplicity, we found piecewise constant inputs appealing for falsification of system requirements, due to different reasons:

(1) We can easily ensure that $u \in U^{[0,T]}$. It is harder to ensure this property using other parameterization methods like spline or polynomial interpolation or Fourier series without putting further constraints-like bounding the sum of Fourier coefficients- on the input. These additional constraints can reduce the input efficiency in practice since for example if we bound the sum of Fourier coefficients by u_{max} the maximum input value can reach u_{max} only if the phase shifts are zero.
(2) Using time and space parameterization, we can test a wide variety of input properties which can reveal different (possibly unsafe) system behaviors: using the time instances we can test time/frequency related features and using the value instances we can test transient properties. Also, the input can change its value u_1 to $u_2 \in U$ at any time (this is important since for example the optimal input for linear systems is a bang-bang control in many cases (see Kirk (2012)).)
(3) While calculating the time and space gradients is straightforward for piecewise constant inputs, it is a bottleneck for other interpolation methods.

6. CASE STUDIES

In this section two case studies are used to evaluate the performance of the time-space gradient descent in testing system requirements. We used MATLAB 2017a on an Intel(R) Core(TM) i7-4790 CPU @3.6 GHZ with 16 GB memory processor with Windows 10 Enterprise for all the case studies and experimental results in this paper.

6.1 Powertrain system

This benchmark is introduced as the third model in Jin et al. (2014). Some preliminary results on the original model are presented in Dokhanchi et al. (2017). These

Algorithm 1 Space-Time GD algorithm for robustness

Require: System model Σ, initial condition and parameterized input x_0 and u, sets of possible initial conditions and input values X_0 and U, system specification φ, final time T, step size h, number of descent iterations k_1, maximum number of the step size decrease k_2 and the multiplier of the step size $p < 1$.

Ensure: Local optimal initial condition x_0^*, local optimal input u^* and the related optimal robustness value r^*

$(x_0', u', r^*) \leftarrow (x_0, u_0, \infty)$
for $i = 1$ *to* k_1 **do**
  $(s_{x_0,u}, \partial s/\partial p) \leftarrow$ SIMUL&SENS(Σ, x_0', u', T)
  $(r, t^*, n_s) \leftarrow$ GETROB$(s_{x_0,u}, \varphi)$
  if $r \leq r^*$ **then**
    $(x_0, u) \leftarrow (x_0', u'), \quad (x_0^*, u^*, r^*) \leftarrow (x_0', u', r)$
  else
    $h' \leftarrow h$
    for $j = 1$ *to* k_2 **do**
      $h' \leftarrow h'.p$
      $(x_0', u') \leftarrow$ INBOX$(x_0, u, X_0, U, h', \delta_x, \delta_u, \delta t)$
      $(s_{x_0,u}, \partial s/\partial p) \leftarrow$ SIMUL&SENS(Σ, x_0', u', T)
      $(r, t^*, n_s) \leftarrow$ GETROB$(s_{x_0,u}, \varphi)$
      if $r \leq r^*$ **then**
        $(x_0^*, u^*, r^*) \leftarrow (x_0', u', r)$
        Break
      end if
    end for
    if $r > r^*$ **then**
      Break
    end if
  end if
  $(dx, du) \leftarrow \mathrm{GD}(t^*, n_s, \partial s/\partial p)$.
  $(x_0', u') \leftarrow$ INBOX$(x_0, u, X_0, U_0, h, \delta_x, \delta_u, \delta t)$
end for

results reveal some of the challenges about this model. The model we study in this paper is the stiff polynomial approximation of the original model. It is a closed loop model of an engine under an air/fuel controller. The closed loop system consists of 5 states and takes two exogenous inputs: the throttle angle θ_{in} and, the engine speed ω. We test the system in the "Normal" operation mode w.r.t the following requirement: "The air/fuel (A/F) ratio remain in the invariant set $[14.56, 14.84]$ from $t = 3$ to the end of the simulation time T=50."

During the test, the engine speed is supposed to remain constant but it can attain different values in $[900\pi/30, 1100\pi/30]$. The throttle input however is a time varying input as a result of possibly different behaviors by the driver, but it is assumed to be bounded in the set $[0, 81.2]$. We used 1 parameter to describe ω and 21 parameters (11 for signal values and 10 for switch times) to describe θ_{in}. Starting from a trajectory with robustness 0.129, we find a falsifying trajectory with robustness -0.003 using GD in 4 iterations. The initial and final trajectories and the inputs are shown in Fig. 4 and 5, respectively.

Note that some properties for this model have been verified/falsified in Fan et al. (2015), but they only test the system under time invariant uncertainties and assume that the driver behaviors (that affect θ_{in}) are limited to two specific behaviors. However, here, we test the system over different driver behaviors.

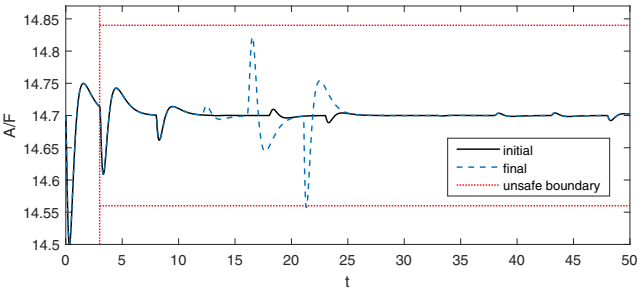

Fig. 4. GD increases the air/flow ratio over and undershoot causing it to find a falsifying trajectory.

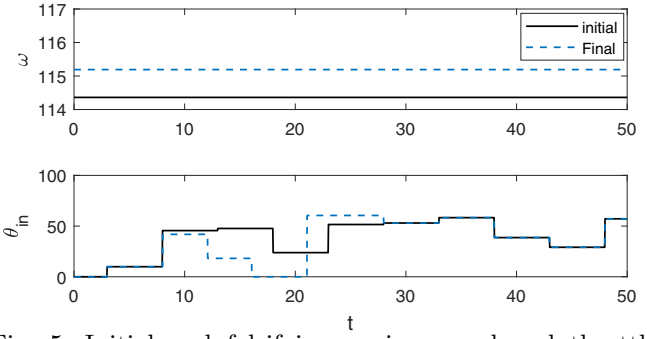

Fig. 5. Initial and falsifying engine speed and throttle input.

6.2 Maneuvering object

This benchmark is a hybrid model also used in Yaghoubi and Fainekos (2017b). The maneuvering object has a pair of off-centered thrusters as the control input. The location-based (hybrid) dynamics of the vehicle is as follows:

$$\begin{bmatrix} \dot{x}_i \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} x_{i+3} \\ 0.1x_4 + \Sigma_{i=1,2} s_i(l)(x_1 - \alpha_i) + F_1 cos(x_5) - F_2 sin(x_5) \\ 0.1x_5 + \Sigma_{i=1,2} s_i(l)(x_2 - \beta_i) + F_1 sin(x_5) - F_2 cos(x_5) \\ -bF_1/I + aF_2/I \end{bmatrix} \tag{13}$$

where $i = 1, 2, 3$ and (x_1, x_2) is the positions along the x and y axis. The hybrid model consists of 3 locations: the first to third locations are specified using the sets $\{x \,|\, x_1 < 4\}$, $\{x \,|\, 4 \leq x_1 \leq 8\}$, and $\{x \,|\, x_1 > 8\}$, respectively. At the first, second and third locations, we have $s_1 = s_2 = 0, s_1 = -1, s_2 = 0$, and $s_1 = 0, s_2 = -2$, respectively. Also, (α_1, β_1) and (α_2, β_2) are the centers of the unsafe sets $\mathcal{U}_1$ and $\mathcal{U}_2$, respectively. We consider $a = b = I = 1$.

The system is required to satisfy this requirement: "Always avoid $\mathcal{U}_1 = [5.5, 6.5] \times [2.5, 3.5]$ and $\mathcal{U}_2 = [9.5, 10.5] \times [1.5, 3.5]$, and eventually reach the goal set $A = [11, 13] \times [3, 5]$ within T=10 seconds."

The search is over the time and the amplitude of the piecewise constant inputs $F_1 \in [0, 0.3]^{[0,T]}, F_2 \in [-.2.2]^{[0,T]}$ and also initial conditions for $x_1 \in [0, 1]$ and $x_2 \in [0.4, 0.8]$. We parametrize F_1 and F_2 using 19 parameters each (10 parameters for signal values and 9 for switch times), so the search is over a 40 dimensional space. Using gradient descent in Algorithm 1 on the negation of the requirement formula, we find inputs F_1 and F_2 that satisfy the requirement. The initial trajectory has robustness 3.3483 w.r.t the negation of the requirement (so it does not satisfy the requirement) and gradient descent decreases the robust-

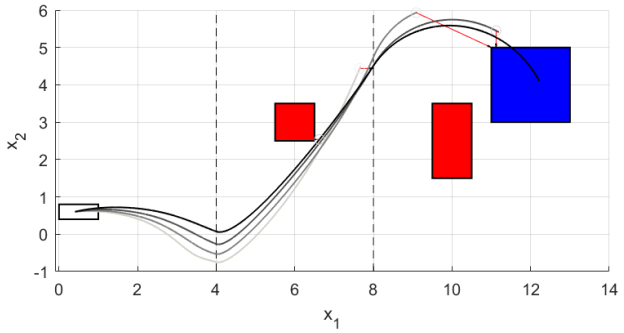

Fig. 6. Maneuvering object trajectories: GD gradually improves the trajectories and finally finds a satisfying trajectory

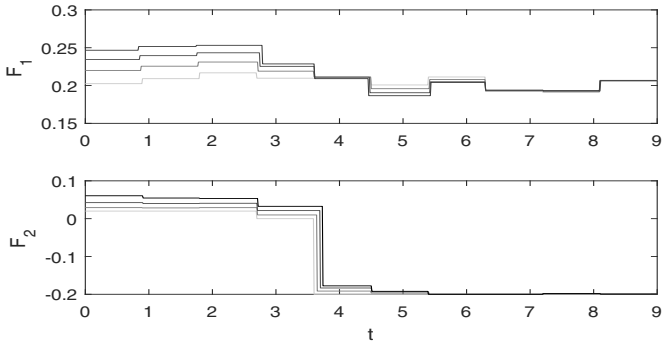

Fig. 7. Maneuvering object control inputs: the shifts along the t-axis are as a result of time descent and the shift along the F-axis are as a result of space descent.

ness to -0.12462 in 4 iterations and finds a trajectory that satisfies the requirement. The system trajectories and their corresponding inputs in these 4 iterations are shown in Fig. 6 and Fig. 7. We show the signals with darker marker as the iteration number increases.

7. EXPERIMENTAL RESULTS

To globally search for the minimizer of the robustness function, we combined Simulated Annealing(SA) with GD. We performed two statistical studies on the "Maneuvering object" and the "Powertrain" benchmarks to determine the effectiveness of applying GD.

In both experiments, we ran SA and SA+GD 50 times starting from same initial conditions and with equal total number of samples N=100. In the "Maneuvering object" case study, we searched for the falsifying behaviors to the requirement mentioned in Section 6.2 and GD was applied 15 times to any SA sample with a robustness value less than 5.5 in SA+GD [1] . The results are presented in Table 1. In the second study on the "Powertrain" benchmark, we tested the system under the requirement in Section 6.1. In SA+GD, we apply GD 15 times to all of the SA samples and we take the next SA sample based on the last GD sample. The results are presented in Table 2.

In both cases, the falsification rate has improved significantly with the help of GD. In the case of the "Maneuvering object" benchmark it achieves one order of magnitude improvement. In the case of the Powertrain benchmark,

[1] Since trajectories with greater robustness values are too far away from falsifying the requirements to be found using local search.

Table 1.

Maneuvering object		
Optim. method	SA	SA+GD
Num. of falsification	6/40	24/40
Avg. min robustness value	4.4435	3.2116
max min robustness value	10.1442	10.1301
min min robustness value	-0.5313	-0.6104

Table 2.

Power-train		
Optim. method	SA	SA+GD
Num. of falsification	0/50	11/50
Avg. min robustness value	0.0725	0.0573
max min robustness value	0.1376	0.1382
min min robustness value	0.0162	-0.0436

while SA+GD falsifies the requirement in 11 out of 50 runs, SA has not been able to find any falsifying input.

8. RELATED WORK

8.1 Simulation-guided falsification

For a discussion on simulation guided verification approaches see Kapinski et al. (2015).

Simulation-guided strategies for testing system properties are devised based on the notion of falsification. Guided by optimization methods, they try to intelligently create test instances that reveal undesirable system behaviors w.r.t a system requirement. Some examples are:

S-TaLiRo and Breach: S-TaliRo (Annpureddy et al. (2011)) and Breach (Donzé (2010)) are Matlab toolboxes that use global optimization solvers to search for counterexamples to system requirements expressed as TL formulas. The tools search the space of the uncertain parameters to find trajectories with minimal robustness values w.r.t the given requirement. They can analyze arbitrary Simulink models or user defined functions that model a system. To search over space of the varying time inputs to the system, the tools require a parameterization representation of the input signal.

S-TaliRo uses stochastic optimization methods like Simulated Annealing, and Ant Colony optimization method, while Breach uses Nelder-Mead simplex-based methods with multiple restarts as the optimization algorithm.

Our method is built on top of S-TaliRo and uses the input parameterization mentioned in Section 2 and looks for the values of the parameters that locally minimize the robustness cost function.

Multiple Shooting: In Zutshi et al. (2014), an approach called multiple shooting is used for falsifying safety properties. They use trajectory segments (disconnected simulation traces) starting from initial values sampled from the whole state space to find a falsifying trajectory. Using system gradient information and NLP solvers they reduce the gaps between segments to create approximate trajectories from promising segmented trajectories. Like our approach, this tool requires gradient information for minimizing segments' endpoint distances, and as a result it is not applicable to general black-box models with arbitrary complexity.

RRT-REX: RRT-REX (Dreossi et al. (2015)) is a toolbox based on rapidly growing random tree (RRT) algorithms which are used for path planning. It utilizes Star-discrepancy coverage measure to guide the exploration of the search space. This measure shows how well a continuous state space is covered. The tool does not require the input signal to be parametrized and the input is allowed to change values based on the robustness values of a partial system trajectory. While the tool looks promising for some models and specifications, it shows poor performance on specifications defined with precise temporal instants. This is not an issue for our method as this precise time instant becomes the critical time $t^*_{x_0,u}$ mentioned in Section 3 with respect to which the GD directions are calculated.

8.2 Optimal control of switched systems

The problem of finding optimal piecewise constant inputs with varying switch times is similar to the problem of finding optimal control policy in switched systems. In Axelsson et al. (2005) an algorithm to find the local optimal policy for mode switches in hybrid dynamical systems is provided. The design variable consists of the switching times and their numbers, and the cost criterion is a functional of the states. In Gonzalez et al. (2010) a hierarchical algorithm is provided that converges to an optimal policy for mode scheduling that consists of a discrete component, the sequence of modes, and two continuous components, the duration and the continuous input of each mode.

9. CONCLUSION

In this paper, we used piecewise constant signals with varying switch times to parameterize the system input space. With this parameterization scheme we can test the system against general temporal logic requirements. The search for unsafe behaviors is optimized locally by finding descent directions for the parameters (signal value and switch time). Using non-trivial benchmarks, we show that combining global stochastic search with local search significantly improves the rate of finding system bugs.

REFERENCES

Abbas, H., Winn, A., Fainekos, G., and Julius, A.A. (2014). Functional gradient descent method for metric temporal logic specifications. In *American Control Conference (ACC), 2014*, 2312–2317. IEEE.

Annpureddy, Y., Liu, C., Fainekos, G.E., and Sankaranarayanan, S. (2011). S-taliro: A tool for temporal logic falsification for hybrid systems. In *TACAS*, volume 6605, 254–257. Springer.

Axelsson, H., Wardi, Y., and Egerstedt, M. (2005). Transition-time optimization for switched systems. *IFAC Proceedings volumes*, 38(1), 453–458.

Bartocci, E., Deshmukh, J., Donzé, A., Fainekos, G., Maler, O., Nickovic, D., and Sankaranarayanan, S. (2018). Specification-based monitoring of cyber-physical systems: A survey on theory, tools and applications. In *Lectures on Runtime Verification - Introductory and Advanced Topics*, volume 10457 of *LNCS*, 128–168. Springer.

Dokhanchi, A., Yaghoubi, S., Hoxha, B., and Fainekos, G. (2017). Arch-comp17 category report: Preliminary results on the falsification benchmarks. In G. Frehse and M. Althoff (eds.), *ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 48 of *EPiC Series in Computing*, 170–174. EasyChair.

Donzé, A. (2010). Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *International Conference on Computer Aided Verification*, 167–170. Springer.

Donzé, A. and Maler, O. (2007). Systematic simulation using sensitivity analysis. *Hybrid Systems: Computation and Control*, 174–189.

Dreossi, T., Dang, T., Donzé, A., Kapinski, J., Jin, X., and Deshmukh, J.V. (2015). Efficient guiding strategies for testing of temporal properties of hybrid systems. In *NASA Formal Methods Symposium*, 127–142. Springer.

Fainekos, G. and Pappas, G. (2009). Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42), 4262–4291.

Fan, C., Duggirala, P.S., Mitra, S., and Viswanathan, M. (2015). Progress on powertrain verification challenge with c2e2. In *ARCH@ CPSWeek*, 207–212.

Gonzalez, H., Vasudevan, R., Kamgarpour, M., Sastry, S.S., Bajcsy, R., and Tomlin, C.J. (2010). A descent algorithm for the optimal control of constrained nonlinear switched dynamical systems. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, 51–60. ACM.

Jin, X., Deshmukh, J.V., Kapinski, J., Ueda, K., and Butts, K. (2014). Powertrain control verification benchmark. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, 253–262. ACM.

Kapinski, J., Deshmukh, J., Jin, X., Ito, H., and Butts, K. (2015). Simulation-guided approaches for verification of automotive powertrain control systems. In *American Control Conference (ACC), 2015*, 4086–4095. IEEE.

Kirk, D.E. (2012). *Optimal control theory: an introduction*. Courier Corporation.

Yaghoubi, S. and Fainekos, G. (2017a). Hybrid approximate gradient and stochastic descent for falsification of nonlinear systems.

Yaghoubi, S. and Fainekos, G. (2017b). Local descent for temporal logic falsification of cyber-physical systems. In *Seventh Workshop on Design, Modeling and Evaluation of Cyber Physical Systems*.

Zutshi, A., Deshmukh, J.V., Sankaranarayanan, S., and Kapinski, J. (2014). Multiple shooting, cegar-based falsification for hybrid systems. In *Proceedings of the 14th International Conference on Embedded Software*, 5. ACM.