Monte-Carlo Techniques for Falsification of Temporal Properties of Non-Linear Hybrid Systems

Truong Nghiem[1], Sriram Sankaranarayanan[2], Georgios Fainekos[3], Franjo Ivančić[4], Aarti Gupta[4] and George J. Pappas[1].
1. Dept. of Electrical Eng., University of Pennsylvania, Philadelphia, PA.
2. Dept. of Computer Science, University of Colorado, Boulder, CO.
3. School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ.
4. NEC Laboratories America, Princeton, NJ.
{nghiem,pappas}@grasp.upenn.edu, srirams@colorado.edu, fainekos@asu.edu, {ivancic,agupta}@nec-labs.com

ABSTRACT

We present a Monte-Carlo optimization technique for finding inputs to a system that falsify a given Metric Temporal Logic (MTL) property. Our approach performs a random walk over the space of inputs guided by a robustness metric defined by the MTL property. Robustness can be used to guide our search for a falsifying trajectory by exploring trajectories with smaller robustness values. We show that the notion of robustness can be generalized to consider hybrid system trajectories. The resulting testing framework can be applied to non-linear hybrid systems with external inputs. We show through numerous experiments on complex systems that using our framework can help automatically falsify properties with more consistency as compared to other means such as uniform sampling.

Categories and Subject Descriptors

G.3 [**Mathematics of Computing**]: Probability and Statistics—*Probabilistic algorithms (including Monte Carlo)*

General Terms

Verification

Keywords

Hybrid Systems, Testing, Robustness, Metric Temporal Logic

1. INTRODUCTION

We propose a technique for finding counterexamples to *Metric Temporal Logic* (MTL) properties for non-linear hybrid systems through global minimization of a *robustness metric*. Global optimization is carried out using a Monte-Carlo technique that performs a random walk over the space of inputs consisting of initial states, controls and disturbances. The robustness metric defines the satisfaction of an MTL property over a given trajectory as a real number, as opposed to the Boolean $0-1$ notion used in Logic. The sign of the metric for a given trajectory $\mathbf{s}$ and formula φ indicates whether $\mathbf{s}$ satisfies φ (written as $\mathbf{s} \models \varphi$). Furthermore, "nearby" trajectories, defined using a metric over trajectories, whose distances from $\mathbf{s}$ are smaller than its robustness also have the same outcome for the property φ as $\mathbf{s}$. Robustness metrics have been previously studied by some of the authors for *robust testing* of hybrid systems [12, 11, 18]. However, for the most part, they have been described over continuous or switched systems trajectories and for properties over the continuous state variables. We provide a definition for hybrid trajectories in this work.

Given a robustness metric, finding a counterexample to a given property φ reduces to finding a trajectory $\mathbf{s}$ that minimizes the robustness score w.r.t φ. This can be viewed as an optimization problem over the space of inputs of the system. However, in practice, this optimization problem is not necessarily guaranteed to be tractable [1]. In almost all cases, the optimization problem (objective function and constraints) cannot be written down in a closed functional form. Nevertheless, such optimization problems can often be solved satisfactorily using Monte-Carlo techniques, that perform a random walk in order to sample from a probability distribution defined implicitly by the robustness metric [31]. Over the long run, the random walk converges to a stationary distribution over the input space such that the neighborhood of inputs with smaller values of robustness are sampled more frequently than inputs with larger values. Furthermore, Monte-Carlo techniques do not require the distribution itself to be known in a closed form. Instead, these techniques simply require the ability to compare the values (ratio) of the probability density function at two given points in the search space. In practice, this reduces to simulating the system using the sampled inputs. The contributions of this paper can be summarized as follows:

1. We show that metrics used for robust testing naturally define objective functions that enable us to cast the problem of falsifying MTL properties into a global optimization problem.
2. We demonstrate the use of hit-and-run Monte-Carlo samplers to carry out this optimization in the presence of (possibly non-convex) constraints over the inputs.
3. We extend our notions to hybrid systems, using quasi-metrics over discrete state-spaces to provide a notion of robustness for hybrid trajectories w.r.t properties that can involve discrete as well as the continuous state variables.

Our approach is applicable even if the property has been proven using a verification technique. In such cases, our technique obtains system trajectories that have low robustness values w.r.t the

HSCC'10, April 12–15, 2010, Stockholm, Sweden.

requirements. In practice, finding non-robust trajectories may imply designs with smaller safety margins. Traditional testing or verification techniques do not consider such trajectories using Boolean notions of temporal satisfaction. Our approach is readily applicable to *Simulink/Stateflow*$^{\text{TM}}$ (S/S) models, since simulating the system is the only primitive needed. We have implemented our approach inside Matlab (TM) and use it to discover counterexamples to MTL properties. We establish that random walks guided by robustness metrics can often falsify MTL properties that cannot be falsified using blind (uniform random) search.

2. PRELIMINARIES

2.1 Problem Definition

In this section, we briefly present the autonomous hybrid automaton model [1]. Non-autonomous systems are considered in Section 3.3. We then present metrics and use them to provide continuous semantics for Metric Temporal Logic (MTL) over continuous time trajectories. This is based on our previous work [12].

Def. 2.1 (Hybrid Automaton). *A* hybrid automaton Ψ *consists of components* $\langle V, \mathbf{L}, \mathcal{T}, \Theta, \mathbf{D}, \mathbf{I}, \ell_0 \rangle$*, wherein,* $V = \{x_1, \ldots, x_n\}$ *is the set of continuous variables;* $\mathbf{L}$ *is a finite set of locations (modes);* $\mathcal{T}$ *is a set of (discrete) transitions such that for each* $\tau : \langle \ell_1 \to \ell_2, g_\tau \rangle \in \mathcal{T}$*, we move from* $\ell_1 \in \mathbf{L}$ *to* $\ell_2 \in \mathbf{L}$ *and the relation* g_τ *over* $V \cup V'$ *is satisfied;* $H_0 = \{\ell_0\} \times \Theta$ *is the set of initial conditions with* $\ell_0 \in \mathbf{L}$ *and* $\Theta \subseteq \mathbb{R}^n$*;* $\mathbf{D}$ *is a mapping of each* $\ell \in \mathbf{L}$ *to a* vector field $\mathbf{D}(\ell)$*; and,* $\mathbf{I}$ *is a mapping of each* $\ell \in \mathbf{L}$ *to a* location invariant *set* $\mathbf{I}(\ell) \subseteq \mathbb{R}^n$.

The product of the locations $\mathbf{L}$ with the continuous state-space defines the hybrid state space $\mathbb{H} = \mathbf{L} \times \mathbb{R}^n$. A (timed) *trajectory* of a hybrid automaton is an infinite sequence of states $\langle l, \vec{x} \rangle \in \mathbf{L} \times \mathbb{R}^n$ of the form $\langle l_0, \vec{x}_0 \rangle$, $\langle l_1, \vec{x}_1 \rangle$, $\langle l_2, \vec{x}_2 \rangle$, ..., such that *initially* $l_0 = \ell_0$ and $\vec{x}_0 \in \Theta$, and for each consecutive state pair $\langle l_i, \vec{x}_i \rangle$, we either make discrete transition from l_i to l_{i+1} or we evolve under the continuous dynamics $\mathbf{D}(l_i)$ from $\vec{x}_i$ to $\vec{x}_{i+1}$. A hybrid automaton Ψ is *deterministic* iff starting from some initial state $\langle \ell_0, \vec{x}_0 \rangle$ there exists a unique trajectory $\mathbf{h} : \mathbb{R}_+ \to \mathbb{H}$ of the automaton ($\mathbb{R}_+$ is the set of non-negative reals). Unless otherwise stated, we consider deterministic hybrid systems throughout this paper. We will also be using the notation $\mathbf{l} : \mathbb{R}_+ \to \mathbf{L}$ to denote the location trajectory and $\mathbf{s} : \mathbb{R}_+ \to \mathbb{R}^n$ to denote the continuous trajectory of the system. In other words, for $t \in \mathbb{R}_+$, $\mathbf{h}(t) = \langle \mathbf{l}(t), \mathbf{s}(t) \rangle$.

MTL Falsification. It is well known that safety properties in themselves do not suffice to specify all system behaviors in practice. This is especially true for real-time embedded systems wherein richer properties such as timing requirements, stability and so on are equally important. Metric Temporal Logic (MTL) introduced by Koymans [20] is a popular formalism for expressing such properties. The problem of verifying a general MTL specification is undecidable for hybrid systems. Consequently, the *bounded-time* verification or falsification of such properties has been studied [27, 29, 11]. Our goal in this work is the efficient falsification of bounded time MTL properties for non-linear hybrid systems.

PROBLEM 2.1. *For an MTL specification* φ*, the MTL falsification problem consists of finding a trajectory of the system* Ψ *starting from some valid initial state* $\langle \ell_0, \vec{x}_0 \rangle$ *such that the resulting hybrid trajectory* $\mathbf{h}$ *or the corresponding continuous trajectory* $\mathbf{s}$ *falsifies specification* φ*, i.e.,* $\mathbf{h} \not\models \varphi$ *or* $\mathbf{s} \not\models \varphi$*, respectively.*

Our proposed solution for Problem 2.1 quantifies the *robustness of satisfaction* of an MTL formula over a system trajectory in order to guide the search for a falsifying trajectory [12]. We now present a brief discussion on metrics and the robustness of MTL formulas.

2.2 Metrics

We first consider state-spaces equipped with a metric function that provides a rigorous notion of "distance" between states. For continuous state spaces, a suitable norm such as L_p norm ($p \geq 1$) provides such a notion. One of our contributions here is an extension of these notions to *structural* (directed-) metrics for hybrid systems. This is described in Section 4.1.

Def. 2.2 (Metric). *A* metric function d*, defined over a state-space* X *is a function* $d : X \times X \mapsto \overline{\mathbb{R}}_+$*, where* $\overline{\mathbb{R}}_+ = [0, +\infty]$*. The metric* d *maps pairs of states to non-negative extended real numbers, satisfying the following properties:*

Identity: $d(x_1, x_2) = 0$ *iff* $x_1 = x_2$*,*

Symmetry: $d(x_1, x_2) = d(x_2, x_1)$*, and*

Triangle Inequality: $d(x_1, x_3) \leq d(x_1, x_2) + d(x_2, x_3)$.

If the Symmetry *condition is dropped from the definition, then* d *is termed a* quasi-metric.

Given a metric d, a radius $\varepsilon > 0$ and a point $x \in X$, the open ε-ball centered at x is defined as $B_d(x, \varepsilon) = \{y \in X \mid d(x, y) < \varepsilon\}$. Finally, if $\mathbf{s}$ and $\mathbf{s}'$ are two system trajectories that take values in a metric space with metric d, we will use ρ_d to denote the metric $\rho_d(\mathbf{s}, \mathbf{s}') = \sup_{t \in R}\{d(\mathbf{s}(t), \mathbf{s}'(t))\}$.

2.3 Robustness of Trajectories

We briefly present the robust interpretation (semantics) of MTL formulas. Details are available from our previous work [12]. In this section, we will refer to system trajectories as *signals*.

Def. 2.3 (MTL Syntax). *Let* AP *be the set of atomic propositions and* $\mathcal{I}$ *be any non-empty interval of* $\overline{\mathbb{R}}_+$*. The set* MTL *of all well-formed MTL formulas is inductively defined as* $\varphi ::= \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \mathcal{U}_{\mathcal{I}} \varphi$*, where* $p \in AP$ *and* $\top$ *is* true.

For real-time/hybrid systems, the atomic propositions label subsets of $\mathbb{R}^n$. An *observation map* $\mathcal{O} : AP \to 2^{\mathbb{R}^n}$ maps each proposition $p \in AP$ to a set $\mathcal{O}(p) \subseteq \mathbb{R}^n$. Without loss of generality, each $\mathcal{O}(p) \subseteq \mathbb{R}^n$ is assumed Lebesgue *measurable*. In Section 4.1, we will demonstrate how to formulate specifications over the hybrid state-space $\mathbb{H}$ instead of the continuous state-space $\mathbb{R}^n$.

We provide semantics that maps an MTL formula φ and a trajectory $\mathbf{s}(t)$ to a value drawn from the linearly ordered set $\overline{\mathbb{R}}$. The semantics for the atomic propositions evaluated for $\mathbf{s}(t)$ consists of the distance between $\mathbf{s}(t)$ and the set $\mathcal{O}(p)$ labeling atomic proposition p. Intuitively, this distance represents how robustly the point $\mathbf{s}(t)$ lies within (or outside) the set $\mathcal{O}(p)$.

Def. 2.4 (Signed Distance). *Let* $x \in X$ *be a point,* $S \subseteq X$ *be a set and* d *be a metric on* X*. We define the signed distance from* x *to* S *to be*

$$\mathbf{Dist}_d(x, S) := \begin{cases} -\inf\{d(x, y) \mid y \in S\} & \text{if } x \notin S \\ \inf\{d(x, y) \mid y \in X \backslash S\} & \text{if } x \in S \end{cases}$$

If this distance is zero, then the smallest perturbation of the point x can affect the outcome of $x \in \mathcal{O}(p)$. We denote the robust valuation of the formula φ over the signal $\mathbf{s}$ at time t by $[\![\varphi, \mathcal{O}]\!]_d(\mathbf{s}, t)$. Formally, $[\![\cdot, \cdot]\!]_d : (MTL \times \mathcal{P}(X)^{AP}) \to (X^{\mathbb{R}_+} \times \mathbb{R}_+ \to \overline{\mathbb{R}})$.

Def. 2.5 (Continuous-Time Robust Semantics). *Let* $\mathbf{s} \in X^{\mathbb{R}_+}$, $c \in \overline{\mathbb{R}}$ *and* $\mathcal{O} \in \mathcal{P}(X)^{AP}$, *then the continuous-time robust semantics of any formula* $\varphi \in MTL$ *with respect to* $\mathbf{s}$ *is recursively defined as follows*

$$\begin{aligned}
[\![\top, \mathcal{O}]\!]_d(\mathbf{s}, t) &:= +\infty \\
[\![p, \mathcal{O}]\!]_d(\mathbf{s}, t) &:= \mathbf{Dist}_d(\mathbf{s}(t), \mathcal{O}(p)) \\
[\![\neg\varphi_1, \mathcal{O}]\!]_d(\mathbf{s}, t) &:= -[\![\varphi_1, \mathcal{O}]\!]_d(\mathbf{s}, t) \\
[\![\varphi_1 \vee \varphi_2, \mathcal{O}]\!]_d(\mathbf{s}, t) &:= \max([\![\varphi_1, \mathcal{O}]\!]_d(\mathbf{s}, t), [\![\varphi_2, \mathcal{O}]\!]_d(\mathbf{s}, t)) \\
[\![\varphi_1 \,\mathcal{U}_{\mathcal{I}} \varphi_2, \mathcal{O}]\!]_d(\mathbf{s}, t) &:= \sup_{t' \in (t+\mathcal{I})} \min\big([\![\varphi_2, \mathcal{O}]\!]_d(\mathbf{s}, t'), \\
&\qquad \inf_{t < t'' < t'} [\![\varphi_1, \mathcal{O}]\!]_d(\mathbf{s}, t'')\big)
\end{aligned}$$

where $t \in \mathbb{R}_+$ *and* $t + \mathcal{I} = \{\tau \mid \exists \tau' \in \mathcal{I} \,.\, \tau = t + \tau'\}$.

For the purposes of the following discussion, let $(\mathbf{s}, t, \mathcal{O}) \models \varphi$ denote the standard Boolean MTL satisfiability. Note that Boolean MTL satisfiability reduces to an application of Def. 2.5 wherein the range of the valuation function is the Boolean set $\mathbb{B} = \{\mathbf{T}, \mathbf{F}\}$ instead of $\overline{\mathbb{R}}$. It is easy to show that if the trajectory satisfies the property, then its robustness is non-negative and, similarly, it the trajectory does not satisfy the property, then its robustness is non-positive. The following result holds [12].

Theorem 2.1. *Given a formula* $\varphi \in MTL$, *an observation map* $\mathcal{O} \in \mathcal{P}(X)^{AP}$ *and a continuous-time signal* $\mathbf{s} \in X^{\mathbb{R}_+}$, *the following hold: (1) If* $(\mathbf{s}, t, \mathcal{O}) \models \varphi$, *then* $[\![\varphi, \mathcal{O}]\!]_d(\mathbf{s}, t) \geq 0$. *In other words,* $\mathbf{s}$ *satisfies the formula* φ *at time instant* $t \geq 0$ *if its distance valuation is non-negative. Conversely, if* $[\![\varphi, \mathcal{O}]\!]_d(\mathbf{s}, t) > 0$, *then* $(\mathbf{s}, t, \mathcal{O}) \models \varphi$. *(2) If for some* $t \in \mathbb{R}^+$, $\varepsilon = [\![\varphi, \mathcal{O}]\!]_d(\mathbf{s}, t) \neq 0$, *then for all* $\mathbf{s}' \in B_{\rho_d}(\mathbf{s}, |\varepsilon|)$, *we have* $(\mathbf{s}, t, \mathcal{O}) \models \varphi$ *if and only if* $(\mathbf{s}', t, \mathcal{O}) \models \varphi$. *I.e,* ε *defines a* robustness tube *around the trajectory such that other "nearby" trajectories lying inside this tube also satisfy* φ.

Theorem 2.1 establishes the robust semantics of MTL as a natural measure of trajectory robustness. Namely, a trajectory is ε robust with respect to an MTL specification φ, if it can tolerate perturbations up to size ε and still maintain its current Boolean truth value. Alternatively, a trajectory with the opposite outcome for φ, if it exists, has a distance of at least ε away.

The precise complexity of the computation of MTL robustness using formula rewriting procedures is still an open problem [12]. However, for LTL formulae φ, a dynamic programming style algorithm can compute the robustness value using $O(mn)$ comparisons, where $m = |\varphi|$ is the size of the formula and n is the length of the simulation trajectory. Similarly, if we assume that all the simulation trajectory is sampled at integer time instants and that the MTL formula involves integer time constants, then the robustness value of an MTL formula can be computed with $O(mnc)$ comparisons, where c is the largest time value that appears in φ. Note that in either case, we also need kn distance computations, where k is the number of atomic propositions that appear in φ. In turn, the computational complexity of the distance computations depends on the type of the sets used for modeling the regions of interest in the state-space. Details are presented elsewhere [12].

3. FALSIFYING CONTINUOUS SYSTEMS

In this section, we provide the basic formulation of falsification in terms of global optimization of a robustness metric defined in Section 2 and describe a Monte-Carlo technique to solve this global optimization.

Let Ψ be a given (deterministic) system whose initial states lie inside the set H_0. Let φ be a given MTL property that we wish to falsify. Given a trajectory $\mathbf{s}$, we have defined a robustness metric $[\![\varphi, \mathcal{O}]\!]_d(\mathbf{s}, t)$ that denotes how robustly $\mathbf{s}$ satisfies (or falsifies) φ at time t. For the following discussion, we assume a fixed label map $\mathcal{O}$ and always interpret the truth (and robustness) of MTL formulas evaluated at the starting time $t = 0$. Let $\mathcal{D}_\varphi(\mathbf{s}) = [\![\varphi, \mathcal{O}]\!]_d(\mathbf{s}, 0)$ denote the robustness metric for $\mathbf{s}$ under these assumptions.

Lifting $\mathcal{D}_\varphi$ *to Inputs:* The robustness metric $\mathcal{D}_\varphi$ maps each trajectory $\mathbf{s}$ to a real number r. The sign of r indicates whether $\mathbf{s} \models \varphi$ and its magnitude $|r|$ measures its robustness. Our goal is to find inputs $\vec{x}_0 \in \Theta$ such that the resulting trajectory $\mathbf{s} \not\models \varphi$, or equivalently $\mathcal{D}_\varphi(\mathbf{s}) \leq 0$. This can be expressed as the optimization of the objective $\mathcal{D}_\varphi$ over the space of all system trajectories:

$$\min_{\text{trajectory}\mathbf{s}} \mathcal{D}_\varphi(\mathbf{s}) \text{ s.t. initial state of } \mathbf{s} : \vec{x}_0 \in \Theta$$

However, the trajectories are not the true *decision variables* for this problem. For instance, it is hard to explore the space of trajectories directly while guaranteeing that each trajectory considered is valid. Fortunately, for deterministic systems, we may associate each input $\vec{x}_0 \in \Theta$ with a unique trajectory $\mathbf{s}$ and vice-versa. Let $\sigma(\vec{x}_0)$ denote the trajectory obtained starting from the initial state $\vec{x}_0$. Let $\mathcal{F}_\varphi(\vec{x}_0) = \mathcal{D}_\varphi(\sigma(\vec{x}_0))$ denote the robustness of the trajectory obtained corresponding to the initial state $\vec{x}_0 \in \Theta$. Therefore, the optimization can be expressed over the space of inputs as follows:

$$\mathsf{minimize}_{\vec{x}_0 \in \Theta}\, \mathcal{F}_\varphi(\vec{x}_0)\,.$$

The inputs $\vec{x}_0$ are the true decision variables of the problem and the optimization is carried out subject to the constraints in Θ.

Non-deterministic Systems: For non-deterministic and stochastic systems, a single input can be associated with multiple (possibly infinitely many) behaviors. For stochastic systems, we may evaluate $\mathcal{F}_\varphi(\vec{x})$ as an expectation obtained by sampling a large but finite set of trajectories. Non-deterministic systems can often be *determinized* by adding new input variables to represent the non-deterministic choice. For the most part, we consider deterministic systems in this paper. For instance, systems modeled in formalisms such as Simulink/Stateflow diagrams (TM) are deterministic, at least in theory.

The resulting optimization problem can be quite complex, unlikely to be convex for all but the simplest of cases. Furthermore, the objective function $\mathcal{F}$ though computable for any given input through simulation, is not expressible in a closed form. Directly obtaining gradients, Hessians and so on is infeasible for all but the simplest of cases. We now present Monte-Carlo techniques that can solve such global optimization problems through a randomized technique that mimics gradient descent in many cases.

3.1 Monte-Carlo Sampling

The Monte-Carlo techniques presented here are based on *acceptance-rejection* sampling [5, 2]. These techniques were first introduced in statistical physics wherein they were employed to simulate the behavior of particles in various potentials [15]. Variations of Monte-Carlo techniques are also widely used for solving global optimization problems [31].

We first present the basic sampling algorithm for drawing samples from a probability distribution and then the technique of *hit-and-run* sampling that respects the (convex) constraints on the input space due to Θ.

Let $f(\vec{x}) = \mathcal{F}_\varphi(\vec{x})$ be a computable robustness function, given a property φ . We seek to minimize f over the inputs in the set

Θ. We wish to sample Θ such that any two points $a, b \in \Theta$ with robustness values $f(a)$ and $f(b)$ are sampled with probability proportional to $\frac{e^{-\beta f_\varphi(a)}}{e^{-\beta f_\varphi(b)}}$, where $\beta > 0$ is a "temperature" parameter, whose significance will be made clear momentarily.

Algorithm 1: Monte-Carlo sampling algorithm.

```
   Input: Θ: Input Space, f(·): Robustness Function,
          ProposalScheme(·): Proposal Scheme
   Result: Samples ⊆ Θ
   begin
       Choose some initial input x⃗ ∈ Θ.
       while (¬Target) do
           /* Select x⃗′ using ProposalScheme      */
1          x⃗′ ← ProposalScheme(x⃗)
2          α ← exp(−β(f(x⃗′) − f(x⃗)))
3          r ← UniformRandomReal(0, 1)
4          if (r ≤ α) then   /* Accept proposal?  */
5              x⃗ ← x⃗′
6              if (f(x⃗) ≤ 0) then reachTarget := true
7          else
8              /* Reject & seek new proposal   */
   end
```

Algorithm 1 shows the schematic implementation of the algorithm. Each iteration of the sampler generates a new *proposal* $\vec{x}' \in \Theta$ from the current sample $\vec{x}$ using some *proposal scheme* defined by the user (Line 1). The objective $f(\vec{x}')$ is computed for this proposal. Subsequently, we compute the ratio $\alpha = e^{-\beta(f(\vec{x}')-f(\vec{x}))}$ (Line 2) and accept the proposal randomly, with probability α (Line 3). Note that if $\alpha \geq 1$ (i.e, $f(\vec{x}') \leq f(\vec{x})$), then the proposal is accepted with certainty. Even if $f(\vec{x}') > f(\vec{x})$ the proposal may still be accepted with some non-zero probability. If the proposal is accepted then $\vec{x}'$ becomes a new sample. Failing this, $\vec{x}$ remains the current sample.

A proposal scheme is generally defined by a probability distribution $P(\vec{x}'|\vec{x})$ that specifies the probability of proposing a new sample input $\vec{x}'$ given the current sample $\vec{x}$. For a technical reason (known as *detailed balance*, see [5]), our version of the algorithm requires that $P(\vec{x}'|\vec{x}) = P(\vec{x}|\vec{x}')$. Furthermore, given any two inputs $\vec{x}, \vec{x}' \in \Theta$, it should be possible with nonzero probability to generate a series of proposals $\vec{x}, \vec{x}_1, \ldots, \vec{x}'$ that takes us from input $\vec{x}$ to $\vec{x}'$. This is necessary in order to guarantee that the entire input state space is covered.

For simplicity, let Θ be bounded and discrete with a large but finite number of points (e.g., consider Θ as a set of finite precision floating point numbers). The robustness function $f(\vec{x})$ over Θ induces a probability distribution:

$$p(\vec{x}) = \frac{1}{M} e^{-\beta f(\vec{x})},$$

where M is the normalizing factor added to ensure that the probabilities add up to one. Suppose Algorithm 1 were run to generate a large number of samples N. Let η denote the frequency function mapping subsets of the input space to the number of times sample was drawn from the set. Let $P(S) = \sum_{\vec{x} \in S} p(\vec{x})$ denote the volume of the probability function for a set $S \subseteq \Theta$.

Theorem 3.1. *In the limit, the acceptance rejection sampling technique (almost surely) generates samples according to the distribution* p, $P(S) = \lim_{N \to \infty} \frac{\eta(S)}{N}$

As a direct consequence, one may conclude, for instance, that an input $\vec{x}_1$ with $f(\vec{x}_1) = -100$ is *more likely* to be sampled as compared to some other input $\vec{x}_2$ with $f(\vec{x}_2) = 100$ in the *long run*. A similar result holds for the continuous case assuming a suitable measure such that $\int_\Theta f(\vec{x}) d\Theta$ is well defined [31].

Importance of β: The overall algorithm itself can be seen as a *randomized* gradient descent, wherein at each step a new point $\vec{x}'$ in the search space is compared against the current sample. The probability of moving the search to the new point follows an exponential distribution on the difference in their robustness values: $p \sim e^{-\beta(f(\vec{x}')-f(\vec{x}))}$. In particular, if $f_\varphi(\vec{x}') \leq f_\varphi(\vec{x})$, the new sample is accepted with certainty. Otherwise, it is accepted with probability $e^{-\beta(f(\vec{x}')-f(\vec{x}))}$. Informally, larger values of β ensure that only reductions to $f(\vec{x})$ are accepted whereas smaller values correspondingly increase the probability of accepting an increase in $f(\vec{x})$. As a result, points with lower values of f are sampled with an exponentially higher probability as compared to points with a higher value of the function f.

It is possible, in theory, to prove assertions about the number N of samples required for the sampled distribution to converge within some distance to the desired distribution governed by $e^{-\beta f_\varphi(\vec{x})}$. The acceptance rejection sampling method implicitly defines a (continuous time) Markov chain on Θ, whose invariant distribution is the distribution $p(\vec{x}) = \frac{f(\vec{x})}{\int_\Theta df(\vec{y})}$ we wish to sample from. For the case of discrete input spaces, the convergence is governed by the *mixing time* of the *Markov chain* defined by the proposal scheme. This time is invariably large (polynomial in the number of input points), and depends on the proposal scheme used.

Adapting β. One of the main drawbacks of Algorithm 1 is that, based on nature of the distribution, the sampling may get "trapped" in *local minima*. This typically results in numerous proposals getting rejected and few being accepted. Even though we are guaranteed eventual convergence, the presence of local minima slows down this process, in practice. We therefore periodically adjust the values of β (and also the proposal scheme) to ensure that the ratio of accepted samples vs. rejected samples remains close to a fixed value (1 in our experiments). This is achieved by monitoring the acceptance ratio during the sampling process and adjusting β based on the acceptance ratio. A high acceptance ratio indicates that β needs to be reduced, while a low acceptance rate indicates that β needs to be increased.

Proposal Schemes. It is relatively simple to arrive at viable schemes for generating new proposals. However, designing a scheme that works well for the underlying problem requires a process of experimentation. For instance, it suffices to simply choose an input $\vec{x}'$ uniformly at random from the inputs, regardless of the current sample. However, such a scheme does not provide many advantages over uniform random sampling. In principle, given a current sample $\vec{x}$, the choice of the next sample $\vec{x}'$ must depend upon $\vec{x}$.

A typical proposal scheme samples from a normal distribution centered at $\vec{x}$ with a suitably adjusted standard deviation (using some covariance matrix H). The covariance can be adjusted periodically based, once again, on the observed samples as well as the acceptance ratio. A smaller standard deviation around $\vec{x}$ yields samples whose robustness values differ very little from $f(\vec{x})$, thus increasing the acceptance ratio. However, it is hard to respect the constraint $\vec{x}' \in \Theta$ using such a proposal scheme.

Hit-and-run proposal scheme. Hit-and-run schemes are useful in the presence of input domains such as Θ. For simplicity, we assume that Θ is convex. Therefore, any line segment in some direction $\vec{v}$

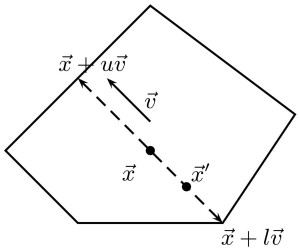


Figure 1: Hit-and-run proposal scheme.

starting from $\vec{x}$ has a maximum offset u such that the entire segment between $\vec{x}$ and $\vec{x} + u\vec{v}$ lies inside Θ.

At each step, we propose a new sample $\vec{x}'$ based on the current sample $\vec{x}$. This is done in two steps:

1. Choose a random unit vector $\vec{v}$ uniformly (or using a Gaussian distribution) (Cf. Fig. 1). In practice, one may choose a random vector $\vec{h}$ and generate a unit vector using $\vec{v} = \frac{\vec{h}}{|\vec{h}|_2}$.

2. Discover the interval $[l, u]$, such that

$$\forall \lambda \in [l, u],\ \vec{x} + \lambda\vec{v} \in \Theta\,.$$

   In other words, $\vec{v}$ yields a line segment containing the point x along the directions $\pm\vec{v}$ and $[l, u]$ represent the minimum and maximum offsets possible along the direction $\vec{v}$ starting from $\vec{x}$. If Θ is a polyhedron, bounds $[l, u]$ may be obtained efficiently by using a variant of the *minimum ratio test*. For a more complex convex set Θ, value of l (resp. u) may be obtained by solving the one dimensional optimization problem $\mathsf{min}(\mathsf{max})\ \lambda$ s.t. $\vec{x} + \lambda\vec{v} \in \Theta$, by using a *bisection* procedure given an initial guess on $[l, u]$.

3. Finally, we choose a value $\lambda \in [l, u]$ based on some probability distribution with a mean around 0. The variance of this distribution is an important parameter that can be used to control the acceptance ratio (along with β) to accelerate convergence.

Hit-and-run samplers can also be used for non-convex input domains such as unions of polytopes and so on. A detailed description of the theory behind such sampling techniques is available elsewhere [34, 31].

However, care must be taken to ensure that the input space Θ is not *skewed* along some direction $\vec{r}$. In the worst case, we may imagine Θ as a straight line segment. In such cases, the hit-and-run proposal scheme fails to generate new samples. This is remedied by adjusting the scheme for selecting unit directions to take the skew of Θ, embedding of Θ inside a subspace spanned by the independent variables and, finally, applying a suitable transformation to Θ that aids in sampling.

In practice, hit and run samplers can work over non-convex, disconnected domains. Theoretical results on these samplers are very promising. Smith [33] proves the asymptotic convergence of hit and run sampling over arbitrary open subsets of $\mathbb{R}^n$. Lovasz [22, 23] has further demonstrated convergence in time $O(n^3)$ for hit and run sampling of uniform distribution over a convex body in n dimensions. Algorithms for global optimization such as *hide-and-seek* [30] and *improving hit-and-run* [36] have combined hit-and-run sampling with Monte-Carlo techniques to generate useful global optimization techniques.

3.2 Computing Robustness

We briefly discuss the computation of the robustness metric for trajectories. Continuous trajectories are hard to compute precisely, even when the analytical form of the solution of the system is known. Thus, trajectories have to be *approximated* numerically.

An approximate *simulation function* G that supports *robust evaluation* of the given property φ should guarantee that for some bounded time horizon $[0, T)$, $\|G(\vec{x}_0, t) - \mathfrak{F}(\vec{x}_0, t)\| \leq \epsilon$, for all $t \in [0, T)$ and for a sufficiently small $\epsilon > 0$. Such a robust simulation function suffices, in practice, to resolve properties that may be of interest to the system designers. An appropriate simulation function can be obtained for a large class of ODEs using numerical simulation techniques of an appropriate order such as *Runge-Kutta* or Taylor-series methods with adaptive step sizes [28]. Numerical integration schemes can also be adapted to provide reliable bounds ϵ on the distance between the actual and the numerical solution. Given a simulation scheme G for a time interval of interest $[0, T)$, we obtain a trace $G_T(\vec{x}_0)$ as a finite set of *sample points* $\{G(\vec{x}_0, t) \mid t \in [0, T)\}$. The robustness $\mathcal{D}_\varphi$ can be approximated using this set of sample points obtained by a numerical integrator.

Unfortunately, for a trajectory σ obtained as the output of a numerical integrator with known error bounds, the trace distance function may no longer satisfy $\mathcal{D}_\varphi(\sigma) \geq 0$ whenever $\sigma \models \varphi$. Instead, we may conclude the existence of some interval $[-\epsilon_2, \epsilon_1]$ for some $\epsilon_1, \epsilon_2 \geq 0$, such that if $\mathcal{D}_\varphi(\sigma) \leq -\epsilon_2$, then $\sigma \not\models \varphi$ and if $\mathcal{D}_\varphi(\sigma) \geq \epsilon_1$ then $\sigma \models \varphi$. In general, we may not draw any conclusions if $-\epsilon_1 \leq \mathcal{D}_\varphi(\sigma) \leq \epsilon_2$. Furthermore, the bounds ϵ_1, ϵ_2 are often unknown for a given system. Nevertheless, the presence of such a bound implies that it still makes sense to perform the optimization using a numerically simulated trajectory function $G(\vec{x}_0, t)$.

$$\mathsf{min.}\ f_P(\vec{x}_0)\ \mathsf{s.t.}\ \vec{x}_0 \in \Theta\,.$$

In practice, such a minimally "robust" simulated trajectories will often be of great interest to system designers even if mathematically speaking they do not violate the property under consideration.

Example 3.1. *Consider the time variant system*

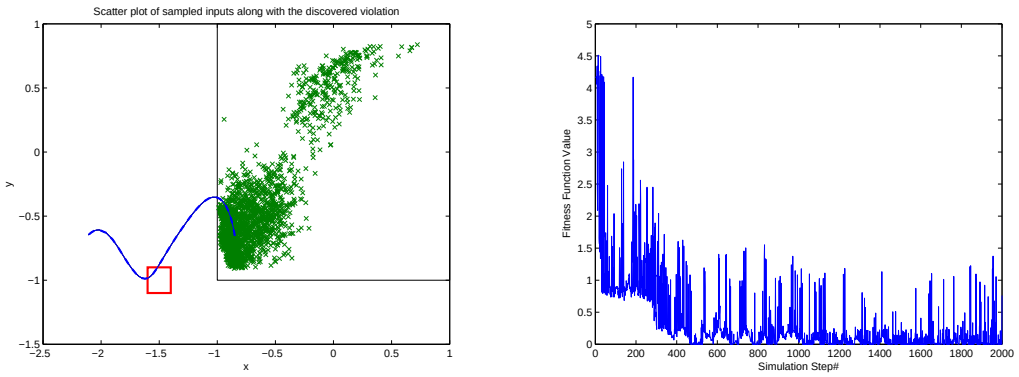
$$\begin{array}{rcl} \frac{dx}{dt} & = & x - y + 0.1t \\ \frac{dy}{dt} & = & y\cos(2\pi y) - x\sin(2\pi x) + 0.1t \end{array}$$

with the initial condition $(x, y) \in [-1, 1] \times [-1, 1]$. *We wish to falsify the property* $\Box_{[0,2]}a$, *wherein* $\mathcal{O}(a) = [-1.6, -1.4] \times [-.9, -1.1]$. *Our simulation uses a numerical ODE solver with a fixed time step over the time interval* $t \in [0, 2]$. *Figure 2(A) shows the trajectory the falsifies our safety property using the hit-and-run sampler and the scatter plot consisting of the samples generated by the Monte-Carlo sampler. Figure 2(B) plots the robustness of the sampled trajectory at each simulation step.*

Remark 3.1. *If the user is willing to tolerate additional computational cost, then it is possible to bound the inaccuracies of the numerical simulation even under the presence of floating-point errors [13]. Then, these bounds can be used to provide bounds on the robustness of the actual continuous-time trajectory [12].*

3.3 Non-autonomous Systems

We now consider extensions to non-autonomous control systems of the form $\dot{\vec{x}} = f(\vec{x}, \vec{u})$, with time-varying inputs $\vec{u} : [0, T] \mapsto \mathbb{R}^m$ constrained to belong to a (measurable) set of functions $\mathcal{U}$. Our goal is to recast the search for control inputs $\vec{u} \in \mathcal{U}$ in terms of a search for a set of parameters $\vec{\lambda}_u \in \mathbb{R}^k$ that lie in some domain Γ. Valuations to the parameters $\vec{\lambda} \in \Gamma$, result in a set of appropriate control inputs $\vec{u}(\vec{\lambda}) \in \mathcal{U}$. Such a parameterization helps us reduce



(A) **(B)**

Figure 2: (A) Time trajectory violating the property $\square_{[0,2]}([-1.6,-1.4]\times[-.9,-1.1])$ along with the scatter plot of sampled inputs and (B) robustness shown in the y-axis as a function of the simulation step number.

the search in terms of a standard global optimization over the space of real-valued decision variables as opposed to functional valued variables. However, such a reduction may not necessarily capture all the control inputs possible in the set $\mathcal{U}$, and therefore restrict our search to a (proper) subset of inputs.

Given a space of inputs $\vec{u} \in \mathcal{U}$, there are numerous ways to parameterize the space of control inputs. We discuss a few such parameterizations below:

Piece-wise Constant Control. We partition the overall time interval $[0,T]$ into a set of intervals $\bigcup_{i=1}^{m}[t_{i-1},t_i)$, wherein $t_0 = 0$ and $t_m = T$. For each interval $[t_{i-1},t_i)$, $i \geq 1$, the control $u(t)$ is restricted to be a constant value λ_{i-1}. The control is therefore parameterized by the tuple $\langle\lambda_0,\ldots,\lambda_m\rangle$, chosen so that the constraints in $\mathcal{U}$ are respected.

Piece-wise Linear Control. Piece-wise constant control may be extended to piecewise linear controls and beyond. Once again, we partition $[0,T]$ into m disjoint intervals. For each interval $[t_{i-1},t_i]$, we restrict the form of each control input to be piece-wise linear: $u(t_{i-1}+\delta) = u(t_{i-1}) + \lambda_{i-1}\delta$. This effectively yields the parameters $\langle u(0),\lambda_0,\ldots,\lambda_m\rangle$ that define the control inputs. These parameters are chosen so that that the resulting controls respect the constraints in $\mathcal{U}$. Extensions to this scheme can introduce higher-order parameters of the form $u(t_{i-1}+\delta) = u(t_{i-1}) + \delta\lambda^1_{i-1} + \delta^2\lambda^2_{i-1}$ and so on.

Spline Functions. We choose a family of splines functions $\upsilon(\vec{x},\vec{\lambda})$ over a set of parameters $\vec{\lambda}$ and seek to express each control input $u(\vec{x}) = \upsilon(\vec{x},\vec{\lambda}_0)$ for some instantiation of the parameters $\vec{\lambda} = \vec{\lambda}_0$. The splines could be varied at definite time instances and continuity/differentiability conditions imposed at such instances.

Example 3.2 (Aircraft Model). *We consider a simple aircraft model describing the movement of an aircraft as a particle in the vertical plane, taken directly from previous work of Lygeros [24]. The equations of motion can be described by the ODE:*

$$\dot{\vec{x}} = \begin{bmatrix} -\frac{S\rho B_0}{2m}\vec{x}_1^2 - g\sin(\vec{x}_2) \\ \frac{S\rho C_0}{2m}\vec{x}_1 - g\frac{\cos(\vec{x}_2)}{\vec{x}_1} \\ \vec{x}_1\sin(\vec{x}_2) \end{bmatrix} + \begin{bmatrix} \frac{u_1}{m} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{S\rho}{2m}\vec{x}_1^2(B_1u_2 + B_2u_2^2) \\ \frac{S\rho C_1}{2m}\vec{x}_1u_2 \\ 0 \end{bmatrix}$$

with state-space variables x_1 describing the aircraft speed, x_2 describing its flight path angle *and x_3 describing its altitude. The control input u_1 represents the thrust and u_2 the* angle of attack. *Numerical values for the parameters are as described by Lygeros [24].*

The initial value $\vec{x}_0$ belongs to the box $[200,260]\times[-10,10]\times[120,150]$. The control inputs are within the range $(u_1(t),u_2(t)) \in [34386,53973]\times[0,16]$. We wish to find initial values and control inputs that falsify the MTL formula

$$\varphi = \square_{[1,1.5]}(x_1 \in [250,260]) \Rightarrow \square_{[3,4]}(x_1 \notin [230,240])\,,$$

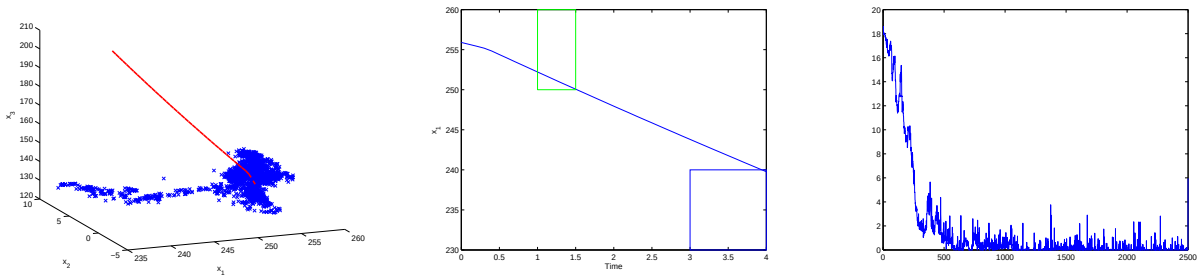
which claims that if the aircraft speed lies in range $[250,260]$ during the time interval $t \in [1,1.5]$, then it cannot be within the range $[230,240]$ within the time interval $t \in [3,4]$. The time interval of interest is taken to be $[0,4.0]$. This is divided into 10 sub-intervals and the control inputs were parameterized to be piece-wise constant within each sub-interval, yielding 20 parameters. Overall, the search space has 24 parameters. We wish to find a control input so that the resulting trajectory falsifies φ.

A hit and run sampler was used for 2500 steps. The samples along a falsifying trajectory found are shown in Fig. 3(a,b). Figure 3(c) shows how the robustness score varies with the number of simulation steps. We note that our implementation assigns a score zero to all falsifying traces. This is performed in order to explore the space of falsifying traces uniformly.

4. FALSIFYING HYBRID SYSTEMS

In this section, we consider the case of falsifying MTL properties for (potentially non-autonomous) hybrid systems. In general, the testing framework, which was presented in Section 3.3, can be applied to switched systems with discrete modes and transitions between them as long as the property φ does not involve the system's discrete modes.

Given a deterministic hybrid system Ψ and an initial condition h_0, let $G(h_0,t)$ represent a simulation function for a set of time instances T such that G approximates the trajectories of the hybrid system. We assume that G approximates the time trajectories with some given tolerance bound ϵ by adjusting the integration method. In practice, this may be harder to achieve for hybrid systems than for purely continuous systems due to the problem of robust event detection [10]. However, assuming that such a simulator is available, we may translate the trace fitness function defined for continuous simulations to hybrid simulations with discrete transitions.



(a) **(b)** **(c)**

Figure 3: (a) Scatter plot of the samples, (b) the trajectory falsifying MTL formula ψ : $\Box_{[1,1.5]}(x_1 \in [250, 260]) \Rightarrow \Box_{[3,4]}(x_1 \notin [230, 240])$ and (c) the variation of robustness of samples for the aircraft model over the simulation steps.

Specifications for hybrid automata involve a sequence of locations of the discrete subsystem. The simplest such property being the (un)reachability of a given "error" location. As a result, continuous state distance based on a norm (or a metric distance) does not, in general, provide a true notion of distance between the specification and the trace. This is especially true in the presence of discrete transitions with reset maps.

4.1 Robustness of Hybrid Trajectories

For the case of hybrid systems with reset maps, the robustness metric used thus far cannot be used to compare the hybrid states $(\ell, \vec{x})$ and $(m, \vec{y})$ in terms of some norm distance between $\vec{x}$ and $\vec{y}$. Therefore, structural considerations based on the graph that connects the different modes of the hybrid automata have to be considered while designing fitness functions. We now consider metrics for hybrid systems.

First, we have to define what is the distance between two modes of the hybrid automaton. We claim that a reasonable metric is the *shortest path distance* between two locations. Intuitively, the shortest path distance provides us with a measure of how close we are to a desirable or undesirable operating mode of the automaton. Such information is especially useful in the class of falsification algorithms that we consider in this paper.

In the following, given hybrid automaton Ψ, we let $\Gamma(\Psi) = (\mathbf{L}, \rightarrow)$ represent the directed graph formed by its discrete modes and transitions. The shortest path distance from node u to node v in the graph $\Gamma(\Psi)$ will be denoted by $\pi(u, v)$. Note that $\pi(u, v) = \infty$ iff there is no path from u to v in the graph $\Gamma(\Psi)$. It is easy to verify that the shortest path distance satisfies all the criteria for a quasi-metric. The shortest path metric can be computed by running a Breadth First Search (BFS) algorithm on the graph. It is well known that BFS runs in linear time on the size of the input graph.

In order to reason about trajectories $\mathbf{h}$ in the hybrid state space $\mathbb{H}$, we introduce a generalized distance $\delta : \mathbb{H} \times \mathbb{H} \rightarrow \mathbb{B}^+$, where $\mathbb{B}^+ = (\{0\} \times \overline{\mathbb{R}}_+) \cup (\mathbb{N}_\infty \times \{+\infty\})$ and $\mathbb{N}_\infty = \mathbb{N} \cup \{\infty\}$, with definition for $h = \langle l, x \rangle \in \mathbb{H}$ and $h' = \langle l', x' \rangle \in \mathbb{H}$,

$$\delta(h, h') = \begin{cases} \langle 0, d(x, x') \rangle & \text{if } l = l' \\ \langle \pi(l, l'), +\infty \rangle & \text{otherwise} \end{cases}$$

where π is the shortest path metric and d is a metric on $\mathbb{R}^n$. In order for our generalized distance to behave like a metric, the range $\mathbb{B}^+$ must be an additive Abelian semigroup with identity and absorbing elements and, also, it must be linearly ordered. We order the set using the dictionary order. Given $\langle k, r \rangle$, $\langle k', r' \rangle \in \mathbb{Z}_\infty \times \overline{\mathbb{R}}_+$, where $\mathbb{Z}$ is the set of integers and $\mathbb{Z}_\infty = \mathbb{Z} \cup \{\pm\infty\}$, we define the order relation $\prec$ as

$$\langle k, r \rangle \prec \langle k', r' \rangle \text{ iff } \begin{cases} k < k' & \text{if } k \neq k' \\ r < r' & \text{if } k = k' \end{cases}$$

Hence, $\mathbb{B}^+$ has a smallest element, namely $\langle 0, 0 \rangle$, and an absorbing element, namely $\langle +\infty, +\infty \rangle$, which is also the least upper bound. Finally, the addition (and the negation in the case of robust semantics) is defined component-wise. It is easy to verify that the generalized distance δ satisfies the identity and triangle inequality properties. In other words, δ is a generalized quasi-metric on $\mathbb{H}$.

The only requirement in the definition of the robust semantics of MTL formulas (Section 2.3) is that both the trajectory under study and the specifications take values from the same space. A straightforward induction on the structure of formula φ will indicate that Theorem 2.1 also holds in the case where the metric d is replaced by a quasi-metric, e.g., δ, in the signed distance function and the metric ρ_d. In the following, we will be using the notation $[\![\cdot, \cdot]\!]_\delta^{\Gamma(\Psi)}$ to indicate that now the graph of the hybrid automaton is required in the robustness computation. Formally, we have that $[\![\cdot, \cdot]\!]_\delta^{\Gamma(\Psi)} : (MTL \times \mathcal{P}(\mathbb{H})^{AP}) \rightarrow (\mathbb{H}^{\mathbb{R}_+} \times \mathbb{R}_+ \rightarrow \mathbb{B})$, where $\mathbb{B} = ((\mathbb{Z}^{<0} \cup \{-\infty\}) \times \{-\infty\}) \cup (\{0\} \times \overline{\mathbb{R}}_-) \cup \mathbb{B}^+$.

Theorem 4.1. *Given a formula $\varphi \in MTL$, an observation map $\mathcal{O} \in \mathcal{P}(\mathbb{H})^{AP}$, a hybrid automaton graph $\Gamma(\Psi)$ and a hybrid trajectory $\mathbf{h} \in \mathbb{H}^{\mathbb{R}_+}$, the following holds. If for some $t \in \mathbb{R}_+$, we have $\langle \varepsilon_1, \varepsilon_2 \rangle = [\![\varphi, \mathcal{O}]\!]_\delta^{\Gamma(\Psi)}(\mathbf{h}, t) \neq \langle 0, 0 \rangle$, then for all $\mathbf{h}' \in B_{\rho_\delta}(\mathbf{h}, \langle |\varepsilon_1|, |\varepsilon_2| \rangle)$, we have $(\mathbf{h}, t, \mathcal{O}, \Gamma(\Psi)) \models \varphi$ if and only if $(\mathbf{h}', t, \mathcal{O}, \Gamma(\Psi)) \models \varphi$.*

Therefore, we are in position to reason about hybrid trajectories without changing our definition of MTL robustness. Now the atomic propositions can map to subsets of $\mathbb{H}$ placing, thus, requirements not only on the continuous state-space, but also on the mode of the hybrid system. Informally, a robustness value of $\langle k, r \rangle$ will mean the following:

- If $k = 0$ and $r \neq 0$, then we can place a tube of radius $|r|$ around the continuous part of the trajectory which will guarantee equivalence under the MTL formula. Moreover, it is required that at each point in time t, the locations are the same for all such trajectories.
- If $k > 0$, then the specification is satisfied and, moreover, the trajectory is k discrete transitions away from being falsified.
- If $k < 0$, then the specification is falsified and, moreover, the trajectory is k discrete transitions away from being satisfied.

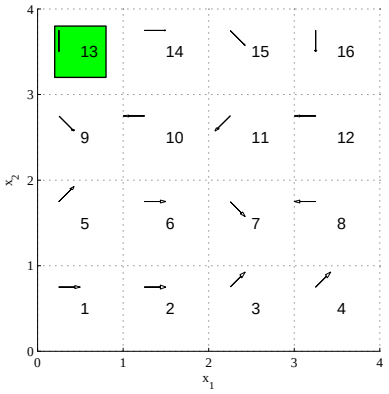


Figure 4: The environment of the vehicle benchmark example (Example 4.1). The arrows indicate the direction of the vector field in each location and the numbers the id of each location. The green box indicates the set of initial conditions projected on the position plane.

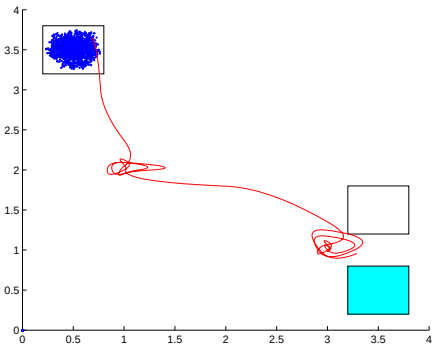


Figure 5: The scatter plot of the sampled initial positions projected on the position plane, along with the least robust trajectory of the vehicle benchmark example (Example 4.1).

This new hybrid notion of MTL robustness is useful in the context of testing for hybrid systems. Namely, our original definition of MTL robustness places requirements only on the observable continuous-time trajectories of the system while ignoring the underlying discrete dynamics. The new robustness notion can structurally distinguish system trajectories that might have similar robustness values otherwise. Thus, it can be used to guide our search algorithms towards less robust system modes. Moreover, we can now impose different requirements at different operating modes of the system. This was not possible before.

Example 4.1. *Consider a complex instance of the vehicle benchmark [14] shown in Fig. 4. The benchmark studies a hybrid automaton Ψ with 4×4 discrete locations and 4 continuous variables x_1, x_2, x_3, x_4 that form the state vector $\vec{x} = [x_1\ x_2\ x_3\ x_4]^T$. We refer to the vectors $[x_1\ x_2]^T$ and $[x_3\ x_4]^T$ as the position and the velocity of the system, respectively. The structure of the hybrid automaton can be better visualized in Fig. 4. The invariant set of every (i, j) location is an 1×1 box that constraints the position of the system, while the velocity can flow unconstrained. The guards in each location are the edges and the vertices that are common among the neighboring locations.*

Each location has affine constant dynamics with drift. In detail, in each location (i, j) of the hybrid automaton, the system evolves under the differential equation $\dot{\vec{x}} = A\vec{x} - Bu(i, j)$ where

$$u(i,j) = [\sin(\pi C(i,j)/4)\ \cos(\pi C(i,j)/4)]^T \text{ and}$$

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1.2 & 0.1 \\ 0 & 0 & 0.1 & -1.2 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1.2 & 0.1 \\ 0.1 & -1.2 \end{bmatrix} \quad C = \begin{bmatrix} 4 & 2 & 3 & 4 \\ 3 & 6 & 5 & 6 \\ 1 & 2 & 3 & 6 \\ 2 & 2 & 1 & 1 \end{bmatrix}$$

Our goal is to find an initial state in the set $H_0 = \{13\} \times [0.2, 0.8] \times [3.2, 3.8] \times [-0.4, 0.4] \times [-0.4, 0.4]$ that will falsify the formula $\varphi = (\neg\ b)\ \mathcal{U}_{[0,25.0]}\ c$, wherein atomic proposition b refers to the shaded rectangle (bottom right) in Fig. 5 and c to the unshaded rectangle above in Fig. 5. In detail, $\mathcal{O}(b) = \{4\} \times [3.2, 3.8] \times [0.2, 0.8] \times \mathbb{R}^2$ and $\mathcal{O}(c) = \{8\} \times [3.2, 3.8] \times [1.2, 1.8] \times \mathbb{R}^2$. Informally, φ says that the system should reach c within 25 time units without passing through b. The least robust (≈ 0) trajectory $\mathbf{h}_l$ found by our algorithm is shown in Fig. 5 along with the scatter plot for the samples. Note that it could be the case that the system is correct with the respect to the specification, but it is definitely not robustly correct.

5. EXPERIMENTS

We have implemented our techniques inside the Matlab(TM) environment. Our implementation is general enough to interact with various means for describing hybrid systems including Simulink / Stateflow models. We currently support full time bounded MTL for continuous as well as hybrid time trajectories.

We perform a comparison of our implementation (MC) against a simple uniform random (UR) exploration of the state-space. Both MC and UR are each run for a set maximum number of iterations, terminating early if a falsifying trajectory is found. Since these techniques are randomized, each experiment was repeated numerous times under different seeds in order to obtain statistically significant results. Uniform random exploration provides an ideal measure of the difficulty of falsifying a property over a given input. Its rate of success empirically quantifies the difficulty of falsifying a given property. Finally, we have already argued about the importance of obtaining the least robust trajectory where falsification cannot be achieved. To this end, we compare the set of minima found using MC as well as that using UR. Table 1 reports on the results of our comparison using different MTL properties for the benchmark systems considered thus far. We find that the performance varies depending on the ease with which the property can be violated by means of uniformly sampling the input space. In almost all the cases, MC technique performs better than uniform random sampling, especially when a falsification is hard to find through UR sampling. We also note, that while MC technique finds the least robust trajectory in almost all cases, the least robust trajectory suffers from large outlying values in some cases. We believe that these values represent local minima that the MC sampling is unable to get away from within the limited number of iterations. In practice, we may periodically reset the MC simulation using random restarts. However, such restarts were not used in our experimental comparison.

6. RELATED WORK

Due to the known undecidability results in the analysis of hybrid systems [1] and the state explosion problem of the reachability computation algorithms (see [18] for some related references), a lot of recent research activity has concentrated on testing approaches to the verification of continuous and hybrid systems [19].

Table 1: Experimental Comparison of Monte-Carlo vs. Uniform Random falsification. Legend: ψ: MTL formula number, #Fals.: number of instances falsified, #MinRob.: best robustness scores, Time: Avg. time in seconds.

ψ	#Run P(MC $\leq$ UR)	#Iter. per run	#Fals. MC	#Fals. UR	MinRob. MC $\langle min, avg, max \rangle$	MinRob. UR $\langle min, avg, max \rangle$	Time MC avg	Time UR avg
Aircraft Example 3.2								
$\Box_{[.5,1.5]} a \wedge \Diamond_{[3,4]} b$	100	500	88	100	$\langle 0, 1.2, 18.6 \rangle$	$\langle 0, 0, 0 \rangle$	5.6	.8
$\Box_{[0,4]} c \wedge \Diamond_{[3.5,4]} d$	100	1000	100	66	$\langle 0, 0, 0 \rangle$	$\langle 0, .02, .2 \rangle$	10.2	28
$\Diamond_{[1,3]} e$	100	2000	81	16	$\langle 0, .9, 40 \rangle$	$\langle 0, .5, 1.3 \rangle$	20.3	34.0
$\Diamond_{[.5,1]} f \wedge \Box_{[3,4]} g$	100	2500	0	0	$\langle 9.5, 9.7, 10.1 \rangle$	$\langle 9.7, 10.7, 12.4 \rangle$	55.5	55.4
$\Box_{[0,.5]} h$	100	2500	100	100	$\langle 0, 0, 0 \rangle$	$\langle 0, 0, 0 \rangle$	3	.5
$\Box_{[2,2.5]} i$	100	2500	99	51	$\langle 0, 0, 1.0 \rangle$	$\langle 0, .2, 1.4 \rangle$	11.9	26.9
Vehicle Benchmark 4.1								
$(\neg\, b)\, \mathcal{U}_{[0,25.0]}\, c$	35	1000	11	8	$\langle 0, .02, .04 \rangle$	$\langle 0, .02, .04 \rangle$	747	804

The use of Monte Carlo techniques for model checking has been considered previously by Grosu and Smolka [17]. Whereas Grosu and Smolka consider random walks over the automaton defined by the system itself, our technique defines random walks over the input state space. These are, in general, distinct approaches to the problem. In practice, our approach does not have the limitation of being restricted by the topology of the system's state transition graph. Depending on this topology, the probability of visiting states deeper in the graph can sometimes be quite small in pathological cases. On the other hand, Grosu et al.'s technique can be extended readily to the case of systems with control inputs without requiring a finite parameterization of the control. We are currently investigating the possibility of combining both types of random walks in a single framework. Previous work by some of the authors in this work considered Monte-Carlo techniques for finding bugs in programs [32]. However, our previous efforts were restricted to safety properties and did not have a systematic definition of robustness that we employ here.

There exist two main approaches to the testing problem of hybrid systems. The first approach is focused on choosing inputs and/or parameters in a systematic fashion so as to cover the state-space of the system [9, 3, 4, 25, 26]. These approaches are mainly based on the theory of rapidly exploring random trees (RRTs). The other approach is based on the notion of robust simulation trajectory [8, 16, 18, 21]. In robust testing, a simulation trajectory can represent a neighborhood of trajectories achieving, thus, better coverage guarantees. Recently, the authors in [7] have made the first steps in bridging these two aforementioned approaches.

On the research front of falsification/verification of temporal logic properties through testing, the results are limited [27, 29, 11]. The work that is the closest to ours appears in [29]. The authors of that work develop a different notion of robustness for temporal logic specifications, which is also used as a fitness function for optimization problems. Besides the differences in the application domain, i.e., [29] focuses on parameter estimation for biological systems, whereas our paper deals with the falsification of hybrid systems, the two works have also several differences at the theoretical and computational levels. At the theoretical level, we have introduced a new metric for hybrid spaces which enables reasoning over hybrid trajectories, while at the computational level our approach avoids set operations, e.g., union, complementation etc, which, in general, increase the computational load.

Younes and Simmons, and more recently, Clarke et al. have proposed the technique of *Statistical Model Checking* (SMC) [35, 6], which generates uniform random inputs to a system subject to some constraints, thus converting a given system into a stochastic system. A probabilistic model checker can be used to prove assertions on the probability that the system satisfies a given temporal property φ. This probability can be safely approximated using Wald's probabilistic ratio test. Statistical model checking, like our technique, requires a simulator to be available for the system but not a transition relation representation. In contrast to SMC, our approach is guided by a robustness metric towards less robust trajectories. On the other hand, the complex nature of the system and the robustness metrics imply that we cannot yet provide guarantees on the probability of satisfaction of the formula.

7. CONCLUSIONS

Embedded systems require the verification of elaborate specifications such as those that can be expressed in MTL. The undecidability of the MTL verification problem over such complex continuous systems mandates the use of lightweight formal methods that usually involve testing. In this paper, we have presented a testing framework for the Metric Temporal Logic (MTL) falsification of non-linear hybrid systems using Monte-Carlo optimization techniques. The use of hit-and-run Monte-Carlo optimization is required in order to overcome the difficulties in handling the complex system dynamics as well as the nonlinearities in the objective function. Moreover, in order to enable more efficient search in hybrid state-spaces, a generalized distance function was introduced.

Experimental results indicate the superiority of our testing framework over random search in most of the benchmark examples. The advantages of our approach are not limited only to the fact that we can falsify arbitrary systems, but also that we can provide robustness guarantees even to systems that have been proven correct. Even though our results are preliminary, the experiments are promising enough to indicate that this might be a practical alternative to hybrid system verification methods.

8. ACKNOWLEDGEMENTS

This research was partially supported by NSF CSR-EHS 0720518.

9. REFERENCES

[1] ALUR, R., COURCOUBETIS, C., HALBWACHS, N., HENZINGER, T. A., HO, P.-H., NICOLLIN, X., OLIVERO,

A., SIFAKIS, J., AND YOVINE, S. The algorithmic analysis of hybrid systems. *Theoretical Computer Science 138*, 1 (1995), 3–34.

[2] ANDRIEU, C., FREITAS, N. D., DOUCET, A., AND JORDAN, M. I. An introduction to MCMC for machine learning. *Machine Learning 50* (2003), 5–43.

[3] BHATIA, A., AND FRAZZOLI, E. Incremental search methods for reachability analysis of continuous and hybrid systems. In *HSCC* (2004), vol. 2993 of *LNCS*, Springer, pp. 142–156.

[4] BRANICKY, M., CURTISS, M., LEVINE, J., AND MORGAN, S. Sampling-based planning, control and verification of hybrid systems. *IEE Proc.-Control Theory Appl. 153*, 5 (2006), 575–590.

[5] CHIB, S., AND GREENBERG, E. Understanding the Metropolis-Hastings algorithm. *The American Statistician 49*, 4 (Nov 1995), 327–335.

[6] CLARKE, E., DONZE, A., AND LEGAY, A. Statistical model checking of analog mixed-signal circuits with an application to a third order $\delta - \sigma$ modulator. In *Hardware and Software: Verification and Testing* (2009), vol. 5394/2009 of *Lecture Notes in Computer Science*, pp. 149–163.

[7] DANG, T., DONZE, A., MALER, O., AND SHALEV, N. Sensitive state-space exploration. In *Proc. of the 47th IEEE CDC* (Dec. 2008), pp. 4049–4054.

[8] DONZE, A., AND MALER, O. Systematic simulation using sensitivity analysis. In *HSCC* (2007), vol. 4416 of *LNCS*, Springer, pp. 174–189.

[9] ESPOSITO, J. M., KIM, J., AND KUMAR, V. Adaptive RRTs for validating hybrid robotic control systems. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics* (2004).

[10] ESPOSITO, J. M., AND KUMAR, V. An asynchronous integration and event detection algorithm for simulating multi-agent hybrid systems. *ACM Trans. Model. Comput. Simul. 14*, 4 (2004), 363–388.

[11] FAINEKOS, G. E., GIRARD, A., AND PAPPAS, G. J. Temporal logic verification using simulation. In *FORMATS* (2006), vol. 4202 of *LNCS*, Springer, pp. 171–186.

[12] FAINEKOS, G. E., AND PAPPAS, G. J. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science 410*, 42 (2009), 4262–4291.

[13] FAINEKOS, G. E., SANKARANARAYANAN, S., IVANČIĆ, F., AND GUPTA, A. Robustness of model-based simulations. In *IEEE Real-Time Systems Symposium* (2009).

[14] FEHNKER, A., AND IVANČIĆ, F. Benchmarks for hybrid systems verification. In *HSCC* (2004), vol. 2993 of *LNCS*, springer, pp. 326–341.

[15] FRENKEL, D., AND SMIT, B. *Understanding Molecular Simulation: From Algorithms to Applications*. Academic Press, 1996.

[16] GIRARD, A., AND PAPPAS, G. J. Verification using simulation. In *HSCC* (2006), vol. 3927 of *LNCS*, Springer, pp. 272 – 286.

[17] GROSU, R., AND SMOLKA, S. Monte Carlo model checking. In *TACAS* (2005), vol. 3440 of *Lecture Notes in Computer Science*, pp. 271–286.

[18] JULIUS, A. A., FAINEKOS, G. E., ANAND, M., LEE, I., AND PAPPAS, G. J. Robust test generation and coverage for hybrid systems. In *HSCC* (2007), no. 4416 in LNCS, Springer, pp. 329–342.

[19] KAPINSKI, J., KROGH, B. H., MALER, O., AND STURSBERG, O. On systematic simulation of open continuous systems. In *HSCC* (2003), vol. 2623 of *LNCS*, Springer, pp. 283–297.

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

[21] LERDA, F., KAPINSKI, J., CLARKE, E. M., AND KROGH, B. H. Verification of supervisory control software using state proximity and merging. In *HSCC* (2008), vol. 4981 of *LNCS*, Springer, pp. 344–357.

[22] LOVASZ, L. Hit-and-run mixes fast. *Mathematical Programming 86* (1999), 443–461.

[23] LOVASZ, L., AND VEMPALA, S. S. Hit-and-run from a corner. *SIAM Journal on Computing 35*, 4 (2006), 985–1005.

[24] LYGEROS, J. On reachability and minimum cost optimal control. *Automatica 40* (2004), 917–927.

[25] NAHHAL, T., AND DANG, T. Test coverage for continuous and hybrid systems. In *CAV* (2007), vol. 4590 of *LNCS*, Springer, pp. 449–462.

[26] PLAKU, E., KAVRAKI, L. E., AND VARDI, M. Y. Hybrid systems: From verification to falsification. In *CAV* (2007), vol. 4590 of *LNCS*, Springer, pp. 463–476.

[27] PLAKU, E., KAVRAKI, L. E., AND VARDI, M. Y. Falsification of LTL safety properties in hybrid systems. In *TACAS* (2009), vol. 5505 of *LNCS*, pp. 368 – 382.

[28] PRESS, W. H., FLANNERY, B. P., TEUKOLSKY, S. A., AND VETTERLING, W. T. *Numerical Recipes: The Art of Scientific Computing*, 2nd ed. Cambridge University Press, Cambridge (UK) and New York, 1992.

[29] RIZK, A., BATT, G., FAGES, F., AND SOLIMAN, S. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In *6th International Conference on Computational Methods in Systems Biology* (2008), no. 5307 in LNCS, Springer, pp. 251–268.

[30] ROMEIGN, H., AND SMITH, R. Simulated annealing for constrained global optimization. *Journal of Global Optimization 5* (1994), 101–126.

[31] RUBINSTEIN, R. Y., AND KROESE, D. P. *Simulation and the Monte Carlo Method*. Wiley Series in Probability and Mathematical Statistics, 2008.

[32] SANKARANARAYANAN, S., CHANG, R. M., JIANG, G., AND IVANČIĆ, F. State space exploration using feedback constraint generation and Monte Carlo sampling. In *ESEC/SIGSOFT FSE* (2007), ACM, pp. 321–330.

[33] SMITH, R. Monte Carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research 38*, 3 (1984), 1296–1308.

[34] SMITH, R. L. The hit-and-run sampler: a globally reaching Markov chain sampler for generating arbitrary multivariate distributions. In *Proceedings of the 28th conference on Winter simulation* (1996), IEEE Computer Society, pp. 260–264.

[35] YOUNES, H. L. S., AND SIMMONS, R. G. Statistical probabilistic model checking with a focus on time-bounded properties. *Information & Computation 204*, 9 (2006), 1368–1409.

[36] ZABINSKY, A., SMITH, R., MACDONALD, J., ROMEIJN, H., AND KAUFMAN, D. Improving hit-and-run for global optimization. *Journal of Global Optimization 3* (1993), 171–192.