Probabilistic Temporal Logic Falsification of Cyber-Physical Systems

HOUSSAM ABBAS, Arizona State University
GEORGIOS FAINEKOS, Arizona State University
SRIRAM SANKARANARAYANAN, University of Colorado, Boulder
FRANJO IVANČIĆ, NEC Laboratories America
AARTI GUPTA, NEC Laboratories America

We present a Monte-Carlo optimization technique for finding system behaviors that falsify a Metric Temporal Logic (MTL) property. Our approach performs a random walk over the space of system inputs guided by a robustness metric defined by the MTL property. Robustness is guiding the search for a falsifying behavior by exploring trajectories with smaller robustness values. The resulting testing framework can be applied to a wide class of Cyber-Physical Systems (CPS). We show through experiments on complex system models 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

Additional Key Words and Phrases: Hybrid Systems, Testing, Robustness, Metric Temporal Logic

ACM Reference Format:
Abbas, H., Fainekos, G., Sankaranarayanan, S., Ivančić, F., and Gupta, A. 2011. Probabilistic Temporal Logic Falsification of Cyber-Physical Systems. ACM Trans. Embedd. Comput. Syst. V, N, Article A (January YYYY), 25 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

Model Based Design (MBD) and automatic code generation are becoming the development methodologies of choice for safety critical applications. Most prominently, such design methodologies have been adopted by the automotive, medical and aerospace industries [Mathworks 2011; Esterel Technologies 2011] where correctness of the end product is of paramount importance. The types of systems in these industrial domains are particularly challenging because software is controlling the safe operation of a physical system. Such systems are also known as Cyber-Physical Systems (CPS). One of the pressing challenges in the MBD of CPS is how to verify the correctness of the developed model of the system as early as possible in the design cycle.

In answering such a problem, one must first specify what is an appropriate mathematical model that captures the behavior of the system and, second, what is an appropriate specification framework that has a nice mathematical structure that can help in analyzing the mathematical model of the system. One such popular mathematical framework for CPS modeling is hybrid automata [Henzinger 1996]. Unfortunately, in general, the verification problem for hybrid automata is undecidable even for simple safety requirements [Henzinger et al. 1998], i.e., there is no terminating algorithm that can answer whether a CPS ever enters a set of bad states. Thus, a lot of research has focused on

This work was partially supported by NSF grants CNS-1017074 and CNS-1016994.
Author's addresses: H. Abbas, School of Electrical, Computer, and Energy Eng., Arizona State University; G. Fainekos, School of Computing, Informatics and Decision Systems Eng., Arizona State University; S. Sankaranarayanan, Department of Computer Science, University of Colorado, Boulder; F. Ivančić and A. Gupta, NEC Laboratories America.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

discovering the classes of hybrid automata where the safety verification problem is decidable [Alur et al. 2000] and on reachability analysis and testing based techniques [Tripakis and Dang 2009].

However, in many cases, the system requirements extend well beyond simple safety properties. For example, we might be interested in conditional requirements such that "if the temperature increases above 10 degrees and remains above 10 degrees for 1 min, then it should be drop below 10 degrees within 2 min and remain below 10 degrees for 30min." Such specifications can be captured using Metric Temporal Logic (MTL) [Koymans 1990].

In this paper, we propose a technique for finding counterexamples to (MTL) properties for CPS 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 notion used in Logic. The sign of the metric for a given trajectory $\mathbf{y}$ and formula φ indicates whether $\mathbf{y}$ satisfies φ (written as $\mathbf{y} \models \varphi$). Furthermore, "nearby" trajectories, defined using a metric over trajectories, whose distances from $\mathbf{y}$ are smaller than its robustness also have the same outcome for the property φ as $\mathbf{y}$.

Given a robustness metric, finding a counterexample to a given property φ reduces to finding a trajectory $\mathbf{y}$ 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. 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 [Rubinstein and Kroese 2008]. 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 work 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 CPS using quasi-metrics to provide a notion of robustness for hybrid trajectories w.r.t properties that can involve discrete as well as 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 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*™ (S/S) models, since simulating the system is the only primitive needed. We have implemented our approach in the Matlab (TM) toolbox S-TALIRO [Annapureddy et al. 2011] 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 easily falsified using blind (uniform random) search.

Preliminary results of this work have appeared in [Nghiem et al. 2010], while the architecture of our toolbox S-TALIRO has appeared in [Annapureddy et al. 2011]. In this paper, we reformulate the problem and its solution into a more general framework, we present the proofs that were omitted from [Nghiem et al. 2010], we provide new hybrid metrics in Section 4 and we perform more thorough experimental analysis using our toolbox S-TALIRO.

2. PRELIMINARIES

In this section, we provide a formal and concise definition of the problem that this work addresses. Then, we introduce metrics and we utilize them to provide continuous semantics for Metric Temporal Logic (MTL) specifications over continuous time trajectories. We will be using the following notation: $\mathbb{R}$ is the set of real numbers; $\overline{\mathbb{R}}$ is the closure of the reals, i.e., $[-\infty, +\infty]$; $\mathbb{R}^+$ is the set of positive real numbers and $\overline{\mathbb{R}}_+$ its closure, i.e., $\overline{\mathbb{R}}_+ = [0, +\infty]$; $\mathbb{N}$ is the set of natural numbers (including 0) and $\mathbb{N}_\infty = \mathbb{N} \cup \{+\infty\}$; $\mathbb{Z}$ is the set of integers and $\mathbb{Z}_\infty = \mathbb{Z} \cup \{\pm\infty\}$. Given sets A and B, B^A defines the set of all functions from A to B and $\mathcal{P}(A)$ denotes the powerset of A.

2.1. Problem Definition

In this work, we take a very general approach in modeling real-time embedded systems that interact with physical systems that have non-trivial dynamics. Such systems are also referred to as hybrid systems or Cyber-Physical Systems (CPS). In the following, we will be using the term hybrid systems since it is more concise. However, we would like to caution the reader against associating hybrid systems with hybrid automata [Alur et al. 1995] since the scope of our work is more general.

We view a system Σ as a mapping from a compact set of initial conditions X_0 and input signals $\mathbf{U} \subseteq U^R$ to output signals Y^R. Here, R is a bounded time domain equipped with a metric d_R, U is a compact set of possible input values at each point in time (input space) and Y is the set of output values (output space). This view of a system is standard in signals and systems [Lee and Varaiya 2003]. We impose four assumptions / restrictions on the systems that we consider:

(1) The input signals (if any) must be parameterizable using a finite number of parameters. That is, there exist two parameter vectors $\lambda = [\lambda_1 \ldots \lambda_m]^T \in \Lambda$, where Λ is a compact set, and $\tau = [\tau_1 \ldots \tau_m]^T \in R^m$ and a function $\mathfrak{U}$ such that for any $u \in \mathbf{U}$, there exist some λ and τ such that for all $t \in R$, $u(t) = \mathfrak{U}(\lambda, \tau)(t)$.
(2) The output space Y must be equipped with a generalized metric $\mathbf{d}$ which contains a subspace Z equipped with a metric d.
(3) For a specific initial condition x_0 and input signal u, there must exist a unique output signal $\mathbf{y}$ defined over the time domain R. That is, the system Σ is deterministic and we implicitly assume that the system does not exhibit Zeno behaviors [Lygeros et al. 2003].
(4) For considering the convergence of our sampling scheme, we assume that the space of inputs is bounded and discretized to a large but finite set. In practice, any representation of the input through a vector of floating point numbers inside the computer must be finite and, therefore, implicitly discretizes the space of inputs. Thus, this assumption does not pose a restriction.

Under Assumption 3, a system Σ can be viewed as a function $\Delta_\Sigma : X_0 \times \mathbf{U} \to Y^R$ which takes as an input an initial condition $x_0 \in X_0$ and an input signal $u \in \mathbf{U}$ and it produces as output a signal $\mathbf{y} : R \to Y$ (also referred to as *trajectory*). When the output signals are only a function of the initial condition, i.e., $\Delta_\Sigma : X_0 \to Y^R$, then the system Σ is called *autonomous*. In either case, the set of all output signals of Σ will be denoted by $\mathcal{L}(\Sigma)$. That is, $\mathcal{L}(\Sigma) = \{\mathbf{y} \mid \exists x_0 \in X_0 \,.\, \exists u \in \mathbf{U} \,.\, \mathbf{y} = \Delta_\Sigma(x_0, u)\}$ or in case of autonomous systems $\mathcal{L}(\Sigma) = \{\mathbf{y} \mid \exists x_0 \in X_0 \,.\, \mathbf{y} = \Delta_\Sigma(x_0)\}$.

Our high level goal is to infer the correctness of the system Σ by observing its response (output signals) to particular input signals and initial conditions. In particular, we are interested in finding witnesses, i.e., output signals, which prove that a requirement or specification is not satisfied by the system. The process of discovering such witnesses is usually referred to as *falsification*.

Example 2.1. As a motivating example, we will consider the Automatic Transmission example which was also considered in [Zhao et al. 2003]. This is a slightly modified version of the Automatic Transmission model provided by Mathworks as a Simulink demo[1]. It is a model of an automatic transmission controller (See Fig. 1) with the following modifications. The only input to the system is the throttle schedule, while the break schedule is set simply to 0 for the duration of the simulation

[1] Available at: `http://www.mathworks.com/products/simulink/demos.html`

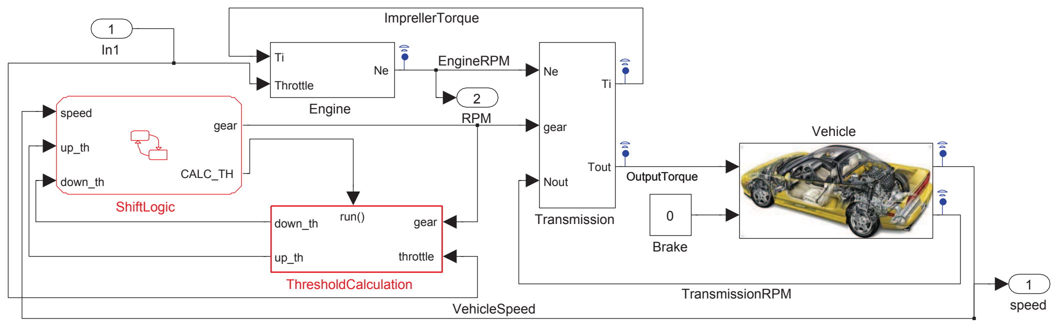

Fig. 1. A modified version of the Simulink (TM) Automatic Transmission Demo.

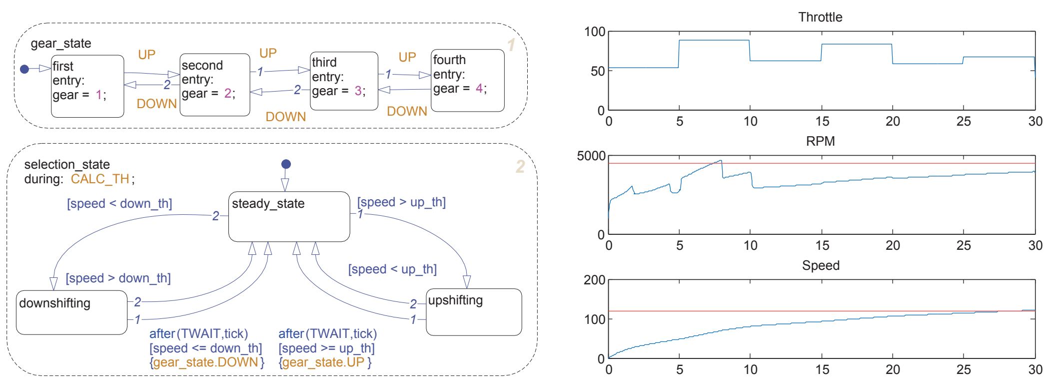

Fig. 2. Example 2.1. *Left:* The switching logic for the automatic drivetrain; *Right:* A input signal and the corresponding output signals that falsify the specification.

which is $30\,\text{sec}$, i.e., $R = [0, 30]$. Finally, the system has two outputs the speed of the engine ω (RPM) and the speed of the vehicle v, i.e., $Y = \mathbb{R}^2$ and $\mathbf{y}(t) = [\omega(t)\ v(t)]^T$ for all $t \in [0, 30]$.

Internally, the system has two 2 continuous-time state variables: the vehicle speed v and engine speed ω. That is, for this example, the output of the system is the same as the continuous state of the system. Initially, the vehicle is at rest at time 0, i.e., $X_0 = \{[0\ 0]^T\}$ and $x_0 = \mathbf{y}(0) = [0\ 0]^T$. Therefore, the output trajectories depend only on the input signal u which models the throttle, i.e., $\mathbf{y} = \Delta_\Sigma(u)$. The throttle at each point in time can take any value between 0 (fully closed) to 100 (fully open). Namely, $u(t) \in [0, 100]$ for each $t \in [0, 30]$. We remark that the system is deterministic, i.e., under the same input u, we will always observe the same output $\mathbf{y}$.

We will assume that a system specification requires that the vehicle speed v is always under 120km/h or that the engine speed ω is always below 4500RPM. Our goal is to falsify the above specification. In other words, we would like to generate tests such that the vehicle speed v and the engine speed ω exceed the values 120km/h and 4500RPM, respectively. Such a falsifying system trajectory appears in Fig. 2.

The model contains 69 blocks out of which there are 2 integrators (i.e., 2 continuous state variables), 3 look-up tables, 3 look-up 2D tables and a Stateflow chart. The Stateflow chart (see Fig. 2) contains two concurrently executing Finite State Machines (FSM) with 4 and 3 states, respectively. Even though this is a small size model and the specification is a simple bounded time reachability requirement, it already exhibits all the complexities that prevent formal modeling and analysis using the state of the art tools, e.g., SpaceEx [Frehse et al. 2011]. $\diamond$

Reachability requirements as described in Example 2.1 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, sequencing of events, conditional requirements, stability and so on are equally important. Metric Temporal Logic (MTL) introduced by Koymans [Koymans 1990] is a popular formalism that can express such properties. Our objective in this work is to provide efficient tools for the falsification of bounded time MTL properties for CPS.

PROBLEM 2.1 (MTL FALSIFICATION). *For an MTL specification φ, the MTL falsification problem consists of finding an output signal $\mathbf{y}$ of the system Σ starting from some valid initial state $x_0 \in X_0$ under an input signal $u \in \mathbf{U}$ such that $\mathbf{y}$ does not satisfy specification φ.*

An overview of our proposed solution to Problem 2.1 appears in Fig. 3. The sampler produces a point x_0 from the set of initial conditions and a vector of parameters λ that characterize the control input signal u. These are passed to the system simulator which returns an execution trace (output trajectory). The trace is then analyzed by the MTL robustness analyzer which returns a robustness value. In turn, the robustness score computed is used by the stochastic sampler to decide on a next input to analyze. If in this process, a falsifying trace is found, it is returned to the user, who can then proceed to examine it inside the system modeling environment.

In this paper, not only we provide an efficient solution to Problem 2.1, but we are also able to provide a measure of how robustly the system satisfies or not an MTL property. That is, our falsification framework does not have to return the first falsifying trajectory it detects, but it can continue searching for the least possible robust system behavior. Similarly, even if the system is not falsifiable, our tool returns the least robust correct behavior that was detected. Such information can be valuable to the system designer.

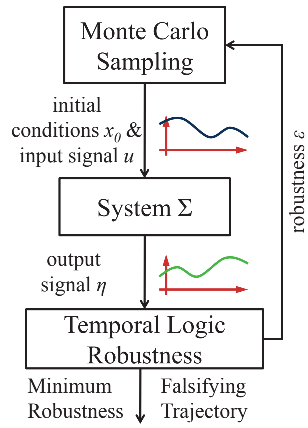

Fig. 3. Overview of the solution to the MTL Falsification of CPS.

2.2. Metrics and Distances

When given a collection of objects, it is frequently necessary to reason about how "close" these objects are to each other. In other words, we need a way to measure or compute the distance between any two objects in the collection. In mathematics, the distance between two objects that belong to a set Y can be quantified by a metric $\mathbf{d}$. The pair $(Y, \mathbf{d})$ is called a metric space.

Metrics arise very naturally in control and analysis of physical systems [Sontag 1998]. Interesting metrics can also be defined in computation theory with a number of diverse applications [Seda and Hitzler 2008]. In either case, the interest in defining metrics is usually to show that a function is contractive (and, thus, to prove some notion of stability [Sontag 1998] or utilize a fixed-point computation [Seda and Hitzler 2008]) or that we can define an interesting topology [Kopperman 1988]. Here, our interest in metrics is different. We are interested in quantifying set membership questions, i.e., how deep is the object within the set it belongs or how far away is from the set it should belong. At a high level, quantification of set membership questions is the subject of study in fuzzy mathematics [Bandemer and Gottwald 1995]. The fundamental difference between fuzzy set theory and our work is that fuzzy set theory abstracts away any topological information regarding the degree of membership. Such topological information is vital in our case as we will demonstrate in Section 2.3. Next, we briefly review the notion of generalized metrics and we refer the reader to [Seda and Hitzler 2008] and the references therein for a more detailed exposition.

Definition 2.2 (*Positively Ordered Commutative Monoid*).

— A *semigroup* $(V, +)$ is a set V together with a binary operation $+$ such that (i) the set is closed under $+$ and (ii) $+$ is associative.
— A *monoid* is a semigroup which has an identity element **0**, i.e., for any $v \in V$, $v + \mathbf{0} = \mathbf{0} + v = \mathbf{0}$.

— A *commutative monoid* is a monoid whose binary operation is commutative.
— An *ordered monoid* $(V, +, \preceq)$ is a monoid with an (partial) order relation $\preceq$ which is compatible with $+$, i.e., $v_1 \preceq v_2$ implies $v_1 + v_3 \preceq v_2 + v_3$ and $v_3 + v_1 \preceq v_3 + v_2$ for all $v_1, v_2, v_3 \in V$.
— A *positively ordered monoid* is an ordered monoid such that for all $v \in V$, $\mathbf{0} \preceq v$.

Definition 2.3 (*Generalized Metric*). Let $(V, +, \preceq)$ be a positively ordered commutative monoid and Y be an arbitrary set. A *generalized metric* $\mathbf{d}$ is a function $\mathbf{d} : Y \times Y \mapsto V$ which satisfies the following properties for $y_1, y_2, y_3 \in Y$: **Identity**: $\mathbf{d}(y_1, y_2) = 0$ iff $y_1 = y_2$, **Symmetry**: $\mathbf{d}(y_1, y_2) = \mathbf{d}(y_2, y_1)$, and **Triangle Inequality**: $\mathbf{d}(y_1, y_3) \preceq \mathbf{d}(y_1, y_2) + \mathbf{d}(y_2, y_3)$.

If V also has an absorbing element ∞, i.e., for any $v \in V$, $v + \infty = \infty + v = \infty$, then $\mathbf{d}$ is called an extended generalized metric. If the *Symmetry* condition is dropped from the definition, then $\mathbf{d}$ is termed a *generalized quasi-metric*. If $(V, +, \preceq)$ is $(\mathbb{R}_+, +, \leq)$ with the usual addition $+$ and total order $\leq$, then we drop the term "generalized" from the terminology and denote the metric by d.

Using a generalized metric $\mathbf{d}$, we can define the distance of a point $y \in Y$ from a set $S \subseteq Y$. Intuitively, this distance is the shortest distance from y to all the points in S. In a similar way, the depth of a point y in a set S is defined to be the shortest distance of y from the boundary of S.

Definition 2.4 (*Distance, Depth, Signed Distance [Boyd and Vandenberghe 2004]* §8). Let $y \in Y$ be a point, $S \subseteq Y$ be a set and $\mathbf{d}$ be a generalized metric on Y. Then, we define the

— Distance from y to S to be $\mathbf{dist}_{\mathbf{d}}(y, S) := \inf\{\mathbf{d}(y, y') \mid y' \in S\}$, and
— Signed Distance $\mathbf{Dist}_{\mathbf{d}}(y, S)$ to be $-\mathbf{dist}_{\mathbf{d}}(y, S)$ if $y \notin S$ and $\mathbf{dist}_{\mathbf{d}}(y, Y \backslash S)$ if $y \in S$.

We should point out that we use the extended definition of supremum and infimum. In other words, the supremum of the empty set is defined to be bottom element of the domain, while the infimum of the empty set is defined to be the top element of the domain. For example, when we reason over $\overline{\mathbb{R}}$, then $\sup \emptyset := -\infty$ and $\inf \emptyset := +\infty$.

Also of importance is the notion of an open ball of radius ε centered at a point $y \in Y$. Given a generalized metric $\mathbf{d}$, a radius $\varepsilon \in V$ and a point $y \in Y$, the open ε-ball (or neighborhood) centered at y is defined as $\mathcal{B}_{\mathbf{d}}(y, \varepsilon) = \{y' \in Y \mid \mathbf{d}(y, y') \prec \varepsilon\}$. The previous definition of a neighborhood includes all points y' which have distance from y less than ε. Since in this work we also use quasi-metrics, we also need the notion of *neighborhood-to*. The neighborhood-to includes all points y' which have distance to y less than ε. Similar to $\mathcal{B}_{\mathbf{d}}$, we define $\mathcal{N}_{\mathbf{d}}(y, \varepsilon) = \{y' \in Y \mid \mathbf{d}(y', y) \prec \varepsilon\}$.

Finally, in order to reason in time about the system behavior, we need to define metrics over signal spaces. If $\mathbf{y}$ and $\mathbf{y}'$ are two system output signals $\mathbf{y}, \mathbf{y}' : R \to Y$ that take values in a generalized metric space $(Y, \mathbf{d})$, we will use $\rho_{\mathbf{d}}$ to denote the metric $\rho_{\mathbf{d}}(\mathbf{y}, \mathbf{y}') = \sup_{t \in R}\{\mathbf{d}(\mathbf{y}(t), \mathbf{y}'(t))\}$.

2.3. Robustness of Trajectories

With the help of metrics we can now provide a robust interpretation (semantics) to MTL formulas. Details are available in our previous work [Fainekos and Pappas 2009]. In this section, we refer to output signals simply as *signals*.

Definition 2.5 (*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 the output space Y. An *observation map* $\mathcal{O} : AP \to \mathcal{P}(Y)$ maps each proposition $p \in AP$ to a set $\mathcal{O}(p)$ to a subset of Y. We require that for all $p \in AP$, $\emptyset \subset \mathcal{O}(p) \subset Y$. We emphasize here that the results in [Fainekos and Pappas 2009] require that the output space Y is equipped with an extended metric d. In Section 4, we relax this requirement and we demonstrate how these results are extended to output spaces which are equipped with a generalized quasi-metric.

We provide semantics that maps an MTL formula φ and a signal $\mathbf{y}(t)$ to a value drawn from the linearly ordered set $\overline{\mathbb{R}}$. The semantics for the atomic propositions evaluated for $\mathbf{y}(t)$ consists of the distance between $\mathbf{y}(t)$ and the set $\mathcal{O}(p)$ labeling atomic proposition p. Intuitively, this distance represents how robustly the point $\mathbf{y}(t)$ lies within (or is outside) the set $\mathcal{O}(p)$. If this distance is zero, then the smallest perturbation of the point y can affect the outcome of $y \in \mathcal{O}(p)$. We denote the robust valuation of the formula φ over the signal $\mathbf{y}$ at time t by $[\![\varphi, \mathcal{O}]\!]_d(\mathbf{y}, t)$. Formally, $[\![\cdot, \cdot]\!]_d : (MTL \times \mathcal{P}(Y)^{AP}) \to (Y^R \times R \to \overline{\mathbb{R}})$.

Definition 2.6 (*Robust Semantics*). Consider a metric space (Y, d), where d is an extended metric. Let $\mathbf{y} \in Y^R$, $c \in \overline{\mathbb{R}}$ and $\mathcal{O} \in \mathcal{P}(Y)^{AP}$, then the robust semantics of any formula $\varphi \in MTL$ with respect to $\mathbf{y}$ is recursively defined as follows

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

where $t \in R$ and $t +_R \mathcal{I} = \{\tau \mid \exists \tau' \in \mathcal{I} \,.\, \tau = t + \tau'\} \cap R$.

Example 2.7. The requirement expressed in natural language in Example 2.1 can be formally written as: $\phi_0^{AT} = \Box p_1^{AT} \vee \Box p_2^{AT}$, where each atomic proposition p_i^0 is mapped to: $\mathcal{O}(p_1^{AT}) = [120, +\infty) \times \mathbb{R}$ and $\mathcal{O}(p_2^{AT}) = \mathbb{R} \times [4500, +\infty)$, respectively. From the designer perspective, it might be easier to conceptualize the falsification problem as a test generation problem and, therefore, pose the formal requirement as the negation of the behavior that she/he would like to observe, i.e., $\phi_1^{AT} = \neg(\Diamond p_1^{AT} \wedge \Diamond p_2^{AT})$. Under the semantics of Def. 2.6, the two formulas are equivalent. $\diamond$

For the purposes of the following discussion, let $(\mathbf{y}, t, \mathcal{O}) \models \varphi$ denote the standard Boolean MTL satisfiability. For clarity in the presentation, we define the satisfiability relation for the base case, i.e., for atomic propositions: $p \in AP$, $(\mathbf{y}, t, \mathcal{O}) \models \varphi$ if $\mathbf{y}(t) \in \mathcal{O}(p)$. Note that Boolean MTL satisfiability reduces to an application of Def. 2.6 wherein the negation is defined to be the Boolean negation and the metric d is the discrete metric: for $y_1, y_2 \in Y$, $d(y_1, y_2) = 0$ if $y_1 = y_2$ and $d(y_1, y_2) = 1$ if $y_1 \neq y_2$. It is easy to show that if the signal satisfies the property, then its robustness is non-negative and, similarly, it the signal does not satisfy the property, then its robustness is non-positive. The following result holds [Fainekos and Pappas 2009].

THEOREM 2.8. *Given an output space (Y, d), where d is an extended metric, a formula $\varphi \in MTL$, an observation map $\mathcal{O} \in \mathcal{P}(Y)^{AP}$ and an output signal $\mathbf{y} \in Y^R$, the following hold:*

(1) *If $(\mathbf{y}, t, \mathcal{O}) \models \varphi$, then $[\![\varphi, \mathcal{O}]\!]_d(\mathbf{y}, t) \geq 0$. Conversely, if $[\![\varphi, \mathcal{O}]\!]_d(\mathbf{y}, t) > 0$, then $(\mathbf{y}, t, \mathcal{O}) \models \varphi$.*
(2) *If $(\mathbf{y}, t, \mathcal{O}) \not\models \varphi$, then $[\![\varphi, \mathcal{O}]\!]_d(\mathbf{y}, t) \leq 0$. Conversely, if $[\![\varphi, \mathcal{O}]\!]_d(\mathbf{y}, t) < 0$, then $(\mathbf{y}, t, \mathcal{O}) \not\models \varphi$.*
(3) *If for some $t \in \mathbb{R}^+$, $\varepsilon = [\![\varphi, \mathcal{O}]\!]_d(\mathbf{y}, t) \neq 0$, then for all $\mathbf{y}' \in B_{\rho_d}(\mathbf{y}, |\varepsilon|)$, we have $(\mathbf{y}, t, \mathcal{O}) \models \varphi$ if and only if $(\mathbf{y}', t, \mathcal{O}) \models \varphi$. I.e, ε defines a* robustness tube *around the trajectory such that other "nearby" trajectories lying inside this tube also satisfy φ.*

Theorem 2.8 establishes the robust semantics of MTL as a natural measure of signal robustness. Namely, a signal 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 signal with the opposite outcome for φ, if it exists, has a distance of at least ε away.

This is the main differentiating property from other works that also consider quantitative semantics for temporal logics such as [de Alfaro et al. 2004; Lamine and Kabanza 2000]. Namely, our semantics maintain the topological information which can be used to define neighborhoods for signals, while in quantitative or fuzzy semantics such information is lost. A more thorough comparison with other quantitative logics is provided in [Fainekos and Pappas 2009].

3. FALSIFYING SYSTEMS WITH METRIC OUTPUT SPACES

In this section, we provide the basic formulation of MTL falsification as a global minimization of the robustness metric defined in Section 2 when the output space (Y, d) is a metric space, i.e., when $(Y, \mathbf{d}) = (Z, d)$, and describe a Monte-Carlo technique to solve this global optimization problem.

Let Σ be a system as defined in Section 2.1. Let φ be a given MTL property that we wish to falsify. Given a signal $\mathbf{y}$, we have defined a robustness metric $[\![\varphi, \mathcal{O}]\!]_d(\mathbf{y}, t)$ that denotes how robustly $\mathbf{y}$ 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 $\mathcal{D}_\varphi(\mathbf{y}) = [\![\varphi, \mathcal{O}]\!]_d(\mathbf{y}, 0)$ denote the robustness metric for $\mathbf{y}$ under these assumptions.

The robustness metric $\mathcal{D}_\varphi$ maps each output signal $\mathbf{y}$ to a real number r. The sign of r indicates whether $\mathbf{y} \models \varphi$ and its magnitude $|r|$ measures its robustness. Ideally, for the MTL verification problem, we would like to prove that $\inf_{\mathbf{y}\in\mathcal{L}(\Sigma)} \mathcal{D}_\varphi(\mathbf{y}) > \varepsilon > 0$ where ε is a desired robustness threshold. For the MTL falsification problem (Problem 2.1), we attempt to solve the problem:

$$\text{Find } \mathbf{y} \in \mathcal{L}(\Sigma) \text{ s.t. } \mathcal{D}_\varphi(\mathbf{y}) < 0 \tag{1}$$

More generally, given a robustness threshold $\varepsilon \geq 0$, we would like to solve the problem:

$$\text{Find } \mathbf{y} \in \mathcal{L}(\Sigma) \text{ s.t. } \mathcal{D}_\varphi(\mathbf{y}) < \varepsilon \tag{2}$$

In this work, we provide a solution to either problem through the optimization problem:

$$\mathbf{y}^\star = \arg \min_{\mathbf{y}\in\mathcal{L}(\Sigma)} \mathcal{D}_\varphi(\mathbf{y}) \tag{3}$$

If $\mathcal{D}_\varphi(\mathbf{y}^\star) < \varepsilon$, then we have produced a counterexample that can be used for debugging.

In the following, we provide parameterizations of the search space and a Monte-Carlo sampling method that will help us solve (3).

3.1. Autonomous Systems

In case of autonomous systems, the space of output signals is not the true search space 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 $x_0 \in X_0$ with a unique trajectory $\mathbf{y}$ and vice-versa. Let $\mathcal{F}_\varphi(x_0) = \mathcal{D}_\varphi(\Delta_\Sigma(x_0))$ denote the robustness of the trajectory obtained corresponding to the initial state $x_0 \in X_0$. Therefore, the optimization can be expressed over the space of inputs as follows:

$$\min_{x_0\in X_0} \mathcal{F}_\varphi(x_0) \tag{4}$$

The components of the vector x_0 are the search variables of the problem and the optimization is carried out subject to the constraints in X_0.

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* $\tilde{\Delta}_\Sigma$ that supports *robust evaluation* of the given property φ should guarantee that for some finite sampling $\tilde{R}$ of the bounded time domain R, for $\tilde{\mathbf{y}} = \tilde{\Delta}_\Sigma(x_0)$ and for $\mathbf{y} = \Delta_\Sigma(x_0)$, $|[\![\phi, \mathcal{O}]\!]_d(\mathbf{y}, t) - [\![\phi, \mathcal{O}]\!]_d(\tilde{\mathbf{y}}, t)| \leq \epsilon$, for all $t \in \tilde{R}$, for a sufficiently small positive ϵ. 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 [Press et al. 1992]. Numerical integration schemes can also be adapted to provide reliable bounds ϵ on the distance between the actual and the numerical solution. Thus, the robustness value $\mathcal{D}_\varphi(\mathbf{y})$ can be approximated by a value $\tilde{\mathcal{D}}_\varphi(\tilde{\mathbf{y}})$ using the set of sample points $\tilde{\mathbf{y}}$ obtained by a numerical integrator. Details on how $\tilde{\mathcal{D}}_\varphi(\tilde{\mathbf{y}})$ can be computed can be found in [Fainekos and Pappas 2009].

Unfortunately, for a trajectory $\tilde{\mathbf{y}}$ obtained as the output of a numerical integrator with known error bounds, the trace distance function may no longer satisfy $\tilde{\mathcal{D}}_\varphi(\tilde{\mathbf{y}}) \geq 0$ whenever $\mathbf{y} \models \varphi$. Instead, we may conclude the existence of some interval $[-\epsilon_2, \epsilon_1]$ for some $\epsilon_1, \epsilon_2 \geq 0$, such that if $\tilde{\mathcal{D}}_\varphi(\tilde{\mathbf{y}}) \leq -\epsilon_2$, then $\mathbf{y} \not\models \varphi$ and if $\tilde{\mathcal{D}}_\varphi(\tilde{\mathbf{y}}) \geq \epsilon_1$ then $\mathbf{y} \models \varphi$. In general, we may not draw any conclusions if $-\epsilon_1 \leq \mathcal{D}_\varphi(\tilde{\mathbf{y}}) \leq \epsilon_2$. 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 $\tilde{\mathbf{y}}$. Thus, our optimization problem becomes:

$$\min_{x_0 \in X_0} \mathcal{F}_\phi(x_0) = \min_{x_0 \in X_0} \tilde{\mathcal{D}}_\varphi(\tilde{\Delta}_\Sigma(x_0)). \tag{5}$$

In practice, even 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.

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 [Fainekos et al. 2009]. Then, these bounds can be used to provide bounds on the robustness of the actual continuous-time trajectory [Fainekos and Pappas 2009].

The resulting optimization problem (5) 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.

3.2. Monte-Carlo Sampling

The Monte-Carlo techniques presented here are based on *acceptance-rejection* sampling [Chib and Greenberg 1995; Andrieu et al. 2003]. These techniques were first introduced in statistical physics, wherein, they were employed to simulate the behavior of particles in various potentials [Frenkel and Smit 1996]. Variations of Monte-Carlo techniques are also widely used for solving global optimization problems [Rubinstein and Kroese 2008]. In this paper, we focus on a class of Monte-Carlo sampling techniques known as Markov-Chain Monte-Carlo (MCMC) techniques. These techniques are based on random walks over a Markov chain that is defined over the space of inputs.

ALGORITHM 1: Monte-Carlo sampling algorithm.

Input: X_0: Input Space, $f(\cdot)$: Robustness Function, ε: Robustness threshold, $\mathsf{PS}(\cdot)$: Proposal Scheme
Output: $x \in X_0$
begin
1 Choose some initial input $x \in X_0$;
2 **while** $(f(x) \geq \varepsilon)$ **do**
/* Select x' using Prop. Scheme */
3 $x' \leftarrow \mathsf{PS}(x)$;
4 $\alpha \leftarrow \mathsf{exp}(-\beta(f(x') - f(x)))$;
5 $r \leftarrow \mathsf{UniformRandomReal}(0, 1)$;
6 **if** $(r \leq \alpha)$ **then** /* Accept proposal? */
7 $x \leftarrow x'$;
end
end

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 X_0. Let $f(x) = \mathcal{F}_\varphi(x)$ be a computable robustness function, given a property φ. We seek to minimize f over the inputs in the set X_0. We wish to sample X_0 such that any two points $x, x' \in X_0$ with robustness values $f(x)$ and $f(x')$ are sampled with probability proportional to $\frac{e^{-\beta f_\varphi(x)}}{e^{-\beta f_\varphi(x')}}$, where $\beta > 0$ is a "temperature" parameter explained in the following.

Algorithm 1 shows the schematic implementation of the algorithm. Each iteration of the sampler generates a new *proposal* $x' \in X_0$ from the current sample x using some *proposal scheme* defined by the user (Line 3). The objective $f(x')$ is computed for this proposal. Subsequently, we compute the ratio $\alpha = e^{-\beta(f(x')-f(x))}$ (Line 4) and accept the proposal randomly, with probability α (Line 5). Note that if $\alpha \geq 1$ (i.e, $f(x') \leq f(x)$), then the proposal is accepted with certainty. Even if $f(x') > f(x)$ the proposal may still be accepted with some non-zero probability. If the proposal is accepted then x' becomes a new sample. Failing this, x remains the current sample. In general, MCMC techniques require the design of a *proposal scheme* for choosing a proposal x' given the current sample x. The convergence of the sampling to the underlying distribution defined by f, depends critically on the choice of this proposal distribution.

Proposal Scheme: A proposal scheme is generally defined by a probability distribution $P(x'|x)$ that specifies the probability of proposing a new sample input x' given the current sample x. In general, there are two requirements that a proposal scheme needs to satisfy so that its use in Algorithm 1 converges to the distribution defined by $f(x)$.

Detailed Balance. The *detailed balance* requirement, see [Chib and Greenberg 1995]), we require that $f(x')P(x'|x) = f(x)P(x|x')$.
Ergodicity. Given any two inputs $x, x' \in X_0$, it should be possible with nonzero probability to generate a series of proposals $x, x_1, \ldots, x'$ that takes us from input x to x'. This is necessary in order to guarantee that the entire input state space is covered.

Convergence: Convergence of the sampling scheme guarantees that eventually after drawing a large but finite number of samples, the distribution of the samples approaches the distribution defined by the robustness function f. We will discuss convergence under the simplifying but practically relevant assumption of discreteness.

We assume that the space of inputs X_0 is *bounded and discrete*, consisting of a large but finite number of points. This assumption is always relevant in practice, since the inputs in X_0 that we consider are finitely represented floating point numbers inside a computer. As a result, the proposal scheme P defines a discrete Markov chain on the space of inputs. Convergence of MCMC sampling follows directly from the convergence of random walks on ergodic Markov Chains [Randall 2006; Chib and Greenberg 1995; Rubinstein and Kroese 2008].

The function $f(x)$ over X_0 induces a discrete probability distribution $p(x) = \frac{1}{M}e^{-\beta f(x)}$, where M is an unknown 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 γ denote the frequency function mapping subsets of the input space to the number of times a sample was drawn from the set. Let $P(S) = \sum_{x \in S} p(x)$ denote the volume of the probability function for a set $S \subseteq X_0$.

THEOREM 3.2. *In the limit, the acceptance rejection sampling technique (almost surely) generates samples according to the distribution* p, $P(S) = \mathsf{lim}_{N\to\infty} \frac{\gamma(S)}{N}$

As a direct consequence, one may conclude, for instance, that an input x_1 with $f(x_1) = -100$ is *more likely* to be sampled as compared to some other input x_2 with $f(x_2) = 100$ in the *long run*.

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 $e^{-\beta f_\varphi(x)}$. This rate of convergence is governed by the *mixing time* of the *Markov chain* on the inputs defined by the proposal scheme. This time is invariably large (polynomial in the number of input points), and depends on the proposal scheme used [Randall 2006].

Importance of β*:* The overall algorithm itself can be seen as a *randomized* gradient descent, wherein at each step a new point 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: $p \sim e^{-\beta(f(x')-f(x))}$. In particular, if $f(x') \leq f(x)$, the new sample is accepted with certainty. Otherwise, it is accepted with probability $e^{-\beta(f(x')-f(x))}$. Informally, larger values of β ensure that only reductions to $f(x)$ are accepted whereas smaller values correspondingly increase the probability of accepting an increase in $f(x)$. 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.

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 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 x, the choice of the next sample x' must depend upon x.

A typical proposal scheme samples from a normal distribution centered at 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 x yields samples whose robustness values differ very little from $f(x)$, thus increasing the acceptance ratio. However, it is hard to respect the constraint $x' \in X_0$ using such a proposal scheme.

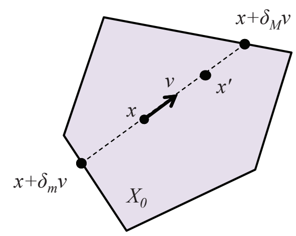

Fig. 4. Hit-and-run proposal scheme.

Hit-and-run proposal scheme. Hit-and-run schemes are useful in the presence of input domains such as $X_0 \subseteq \mathbb{R}^n$. For simplicity, we assume that X_0 is convex. Therefore, any line segment in some direction v starting from x has a maximum offset δ_M such that the entire segment between x and $x + \delta v$ lies inside X_0. At each step, we propose a new sample x' based on the current sample x. This is done in three steps:

(1) Choose a random unit vector v uniformly (or using a Gaussian distribution) (Cf. Fig. 4). In practice, one may choose a random vector h and generate a unit vector using $v = \frac{h}{|h|_2}$.
(2) Discover the interval $[\delta_m, \delta_M]$, such that $\forall \delta \in [\delta_m, \delta_M],\ x + \delta v \in X_0$. In other words, v yields a line segment containing the point x along the directions $\pm v$ and $[\delta_m, \delta_M]$ represent the minimum and maximum offsets possible along the direction v starting from x. If X_0 is a polyhedron, bounds $[\delta_m, \delta_M]$ may be obtained efficiently by using a variant of the *minimum ratio test*. For a more complex convex set X_0, value of δ_m (resp. δ_M) may be obtained by solving the one dimensional optimization problem $\mathsf{min}(\mathsf{max})\ \delta$ s.t. $x + \delta v \in X_0$, by using a *bisection* procedure given an initial guess on $[\delta_m, \delta_M]$.
(3) Finally, we choose a value $\delta \in [\delta_m, \delta_M]$ 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 [Smith 1996; Rubinstein and Kroese 2008].

However, care must be taken to ensure that the input space X_0 is not *skewed* along some direction v'. In the worst case, we may imagine X_0 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

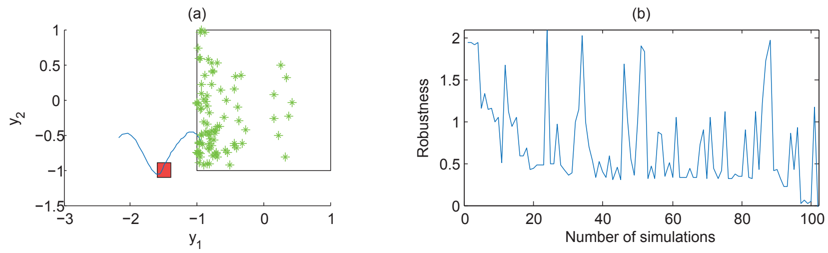

Fig. 5. (a) Time trajectory violating the property $\Box_{[0,2]}\neg a$, where $\mathcal{O}(a) = [-1.6, -1.4] \times [-.9, -1.1]$ along with the scatter plot of sampled inputs and (b) robustness value as a function of the simulation step number.

selecting unit directions to take the skew of X_0, embedding of X_0 inside a subspace spanned by the independent variables and, finally, applying a suitable transformation to X_0 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 [Smith 1984] proves the asymptotic convergence of hit and run sampling over arbitrary open subsets of $\mathbb{R}^n$. Lovasz [Lovasz 1999; Lovasz and Vempala 2006] 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* [Romeign and Smith 1994] and *improving hit-and-run* [Zabinsky et al. 1993] have combined hit-and-run sampling with Monte-Carlo to generate global optimization techniques.

Example 3.3. Let $\mathbf{y}(t) = [\mathbf{y}_1(t)\ \mathbf{y}_2(t)]^T$. Consider the time varying system

$$\frac{d\mathbf{y}(t)}{dt} = \begin{bmatrix} \frac{d\mathbf{y}_1(t)}{dt} \\ \frac{d\mathbf{y}_2(t)}{dt} \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1(t) - \mathbf{y}_2(t) + 0.1t \\ \mathbf{y}_2(t)\cos(2\pi\mathbf{y}_2(t)) - \mathbf{y}_1(t)\sin(2\pi\mathbf{y}_1(t)) + 0.1t \end{bmatrix}$$

with initial condition $\mathbf{y}(0) = x_0 \in X_0 = [-1, 1] \times [-1, 1]$. In this case, $Y = \mathbb{R}^2$ and, thus, we choose to use the Euclidean metric. We wish to falsify the property $\Box_{[0,2]}\neg a$, 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 $t \in R = [0, 2]$. Figure 5(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 5(b) plots the robustness of the trajectory at each simulation step. We observe that the sampling is concentrated in the more promising regions in the set of initial conditions. ◇

3.3. Non-autonomous Systems

We now consider extensions to non-autonomous CPS. Again, for pragmatic reasons, we focus on the approximation $\tilde{\mathbf{y}} = \tilde{\Delta}_\Sigma(x_0, \tilde{u})$ of the actual trajectory $\mathbf{y} = \Delta_\Sigma(x_0, u)$. Here, the input signal $\tilde{u}$ is a discrete-time approximation of the actual continuous-time input signal u. Therefore, in a naive search for a falsifying input signal, we may consider each sampling instance as a search variable. However, such an approach is infeasible for long simulation times with fast sampling rates.

Our goal is to recast the search for control input signals $\tilde{u}$ in terms of a search in the set of parameters $\lambda \in \Lambda$ and $\tau \in R^m$, where $m << |\tilde{R}|$, i.e., m is substantially smaller than the number of samples from R. Since we have assumed that the input signal space can be parameterized on λ and τ, we can produce a discrete-time approximation $\tilde{u} = \tilde{\mathfrak{U}}(\lambda, \tau)$ to $u = \mathfrak{U}(\lambda, \tau)$ and, thus, we are able to represent realistic input signals. Now, our optimization problem becomes:

$$\min_{\langle x_0,\lambda,\tau\rangle \in X_0\times\Lambda\times R^m} f(x_0, \lambda, \tau) = \min_{\langle x_0,\lambda,\tau\rangle \in X_0\times\Lambda\times R^m} \tilde{\mathcal{D}}_\varphi(\tilde{\Delta}_\Sigma(x_0, \tilde{\mathfrak{U}}(\lambda, \tau))). \tag{6}$$

In practical terms, there exist numerous ways to parameterize the space of control inputs. We discuss a few such parameterizations below:

Piece-wise Constant Input: We partition the overall time interval $R = [0, T]$ into a set of intervals $\bigcup_{i=1}^{m} [\tau_{i-1}, \tau_i)$, wherein $\tau_0 = 0$ and $\tau_m = T$. For each interval $[\tau_{i-1}, \tau_i),\ i \geq 1$, the control $u(t)$ is restricted to be a constant value λ_{i-1}.

Piece-wise Linear Input: Piece-wise constant control may be extended to piecewise linear controls. Once again, we partition $R = [0, T]$ into m disjoint intervals. For each interval $[\tau_{i-1}, \tau_i]$, we restrict the form of each control input to be piece-wise linear, i.e., for $t \in [t_{i-1}, t_i)$, we have $u(t) = (1 - \alpha(t))\lambda_{i-1} + \alpha(t)\lambda_i$ where $\alpha(t) = (-\tau_{i-1})/(\tau_i - \tau_{i-1})$.

Spline Functions. We can choose a family of spline functions $\mathfrak{U}_S(\lambda, \tau)$. Details on utilizing splines to represent control input signals can be found in [Egerstedt and Martin 2009].

Example 3.4. In order to parameterize the input signal space of Example 2.1, we used a piece-wise constant signal with 7 control points uniformly distributed over the time domain $[0, 30]$. That is, our search for a minima is performed over a bounded 7 dimensional space. Furthermore, since the output space Y is $\mathbb{R}^2$, we are using the Euclidean metric for the distance computations in the formula defined in Example 2.7. The outcome of S-TALIRO appears in Fig. 2. As evident from the figure, the vehicle speed and the engine rotation indeed reach the specified thresholds. The Simulink model was simulated 41 times for this particular test. $\diamond$

4. FALSIFYING SYSTEMS WITH GENERALIZED QUASI-METRIC OUTPUT SPACES

In the previous sections, we demonstrated that MTL falsification of systems is possible as long as we can define a non-trivial metric on the output space. However, specifications on CPS usually have requirements on both the discrete output space of the system and the continuous output space. However, it is not straightforward to define metrics over such hybrid (discrete & continuous) output spaces. Therefore, in order to formulate and analyze such specifications, we need to relax our constraint on the system having metric output spaces.

Example 4.1. Let us revisit Example 2.1. We are looking to generate tests such that the system visits each state in the state chart `selection_state` (see Fig. 2), i.e., `steady_state`, `upshifting` and `downshifting`, when the vehicle speed exceeds 79. In this case, the output trajectory $\mathbf{y}$ of the system model must not only contain information about the physical system quantities, i.e., engine rotation and vehicle speed, but also about the current state chart state of the system. Therefore, the temporal logic analysis must be performed over the output space $Y = \{\texttt{steady\_state}, \texttt{upshifting}, \texttt{downshifting}\} \times \mathbb{R}^2$. $\diamond$

In this section, we first generalize Theorem 2.6 to signals over generalized quasi-metric output spaces. Then, we introduce the modeling formalism of hybrid automata and two interesting generalized quasi-metrics on output trajectories of hybrid automata.

4.1. Robustness of Signals over Generalized Quasi-Metrics Spaces

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. We can prove (see Appendix) by induction on the structure of formula φ that Theorem 2.8 also holds in the case where the metric d is replaced by a generalized quasi-metric $\mathbf{d}$.

THEOREM 4.2. *Given an output space $(Y, \mathbf{d})$, where $\mathbf{d}$ is an extended generalized quasi-metric, a formula $\varphi \in MTL$, an observation map $\mathcal{O} \in \mathcal{P}(Y)^{AP}$ and an output signal $\mathbf{y} \in Y^R$, the following hold:*

(1) *If $(\mathbf{y}, t, \mathcal{O}) \models \varphi$, then $[\![\varphi, \mathcal{O}]\!]_{\mathbf{d}}(\mathbf{y}, t) \succeq \mathbf{0}$. Conversely, if $[\![\varphi, \mathcal{O}]\!]_{\mathbf{d}}(\mathbf{y}, t) \succ 0$, then $(\mathbf{y}, t, \mathcal{O}) \models \varphi$.*
(2) *If $(\mathbf{y}, t, \mathcal{O}) \not\models \varphi$, then $[\![\varphi, \mathcal{O}]\!]_{\mathbf{d}}(\mathbf{y}, t) \preceq \mathbf{0}$. Conversely, if $[\![\varphi, \mathcal{O}]\!]_{\mathbf{d}}(\mathbf{y}, t) \prec 0$, then $(\mathbf{y}, t, \mathcal{O}) \not\models \varphi$.*
(3) *If for some time $t \in R$, $\varepsilon = [\![\varphi, \mathcal{O}]\!]_{\mathbf{d}}(\mathbf{y}, t) \neq \mathbf{0}$, then for all $\mathbf{y}' \in \mathcal{B}_{\rho_{\mathbf{d}}}(\mathbf{y}, |\varepsilon|)$, we have $(\mathbf{y}, t, \mathcal{O}) \models \varphi$ if and only if $(\mathbf{y}', t, \mathcal{O}) \models \varphi$.*

Note that now the definition of the robustness valuation function for a formula φ over a signal $\mathbf{y}$ at time t is a function $[\![\cdot,\cdot]\!]_{\mathbf{d}} : (MTL \times \mathcal{P}(Y)^{AP}) \to (Y^R \times R \to \mathbb{V})$. The set $\mathbb{V}$ must include the set V of the positively ordered monoid $(V, +, \preceq)$ in the definition of the generalized quasi-metric $\mathbf{d}$ and, also, it must be ordered under the same ordering relation $\preceq$. Furthermore, appropriate definitions of negation and absolute value are required as well as careful treatment of the absorbing elements (if any). Essentially, we need $(\mathbb{V}, +, \preceq)$ to be an Abelian group with two absorbing elements $\pm\infty$.

4.2. Generalized Quasi-Metrics for Hybrid Signals

In order to define quasi-metrics for hybrid signals, we need to take into account some information about the structure of the system that generates the output signals. Here, we will be using a generalization of hybrid automata [Alur et al. 1995] as a basic modeling language for CPS. We remark that our formalism resembles more hierarchical hybrid systems [Alur et al. 2003]

Definition 4.3 (*Hybrid System*). A *hybrid system* $\mathcal{H}$ consists of components $\langle H$, H_0, Y, $\mathbf{O}$, $\to$, $\mathbf{G}^t$, $\mathbf{R}$, $\mathbf{D}$, $\mathbf{U}\rangle$, wherein,

— $H = L \times X$ is the state space of the system and L is a finite set of locations (*modes* or *control locations*),
— $H_0 \subseteq H$ represents the set of initial conditions,
— $Y = L \times Z$ is the output space, where (Z, d) is a metric space
— $\mathbf{O} : X \to Z$ is an output map,
— $\to \subseteq L \times L$ is a set of (discrete) transitions such that for each $\langle \ell_1, \ell_2\rangle \in \to$, the system moves from $\ell_1 \in L$ to $\ell_2 \in L$ if the output state $z = \mathbf{O}(x)$ of the system before the transition is in the set $\mathbf{G}(\ell_1, \ell_2, x)$ and after the transition is at the point $z' = \mathbf{O}(x')$ where $x' = \mathbf{R}_{\langle \ell_1,\ell_2\rangle}(x)$,
— $\mathbf{G} : L \times L \times X \to \mathcal{P}(Z)$ is the guard set for the transitions between control locations,
— $\mathbf{R} : (L \times L) \to (X \to X)$ is the reset function for the transitions between control locations,
— $\mathbf{D} : L \to (X \times R \times \mathbf{U} \to X^R)$ is a mapping of each control location $\ell \in L$ to a deterministic subsystem, which given an initial condition x_0, an initial time t_0 and an input signal u, returns the unique state trajectory of the subsystem $\mathbf{x}_\ell = \mathbf{D}_\ell(x_0, t_0, u)$, and, finally,
— $\mathbf{U}$ is the set of possible input signals.

We remark that our definition of a hybrid system allows each control location to be any arbitrary subsystem as long as it is deterministic and its state can be fully described by the function $\mathbf{D}_\ell$. For example, each control location can be a hybrid system as well. The reason behind utilizing such a general model is that we are not necessarily interested in the whole structure of the hybrid system, but only on its part that is directly related to the functional specification that we are trying to falsify.

Example 4.4. The Simulink/Stateflow model in Example 2.1 has state-space

$$\{\texttt{first}, \texttt{second}, \texttt{third}, \texttt{fourth}\} \times \{\texttt{steady\_state}, \texttt{upshifting}, \texttt{downshifting}\} \times \mathbb{R}^2.$$

In Example 4.1, the specification requirements focus only on the state chart `selection_state`. Therefore, our hybrid system will have the following components of interest:

— $L = \{\texttt{steady\_state}, \texttt{upshifting}, \texttt{downshifting}\}$ and $\to$ as defined in Fig. 2.
— $X = \{\texttt{first}, \texttt{second}, \texttt{third}, \texttt{fourth}\} \times \mathbb{R}^2$, $Z = \mathbb{R}^2$ and $\mathbf{O}$ is the projection of X on $\mathbb{R}^2$.

The reset function $\mathbf{R}$ changes the state of the state chart `gear_state` and the guard $\mathbf{G}$ is computed by the `ThresholdCalculation` block in the Simulink model in Fig. 1. However, we are not interested in the components $\mathbf{R}$ and $\mathbf{G}$ in this example. $\diamond$

A *timed trace* of a hybrid automaton is a *finite*[2] sequence of states $\langle t, \ell, x\rangle \in R \times L \times X$ of the form $\langle t_0, \ell_0, x_0\rangle$, $\langle t_1, \ell_1, x_1\rangle$, $\langle t_2, \ell_2, x_2\rangle$, . . ., such that *initially*, at time t_0, we have $\langle \ell_0, x_0\rangle \in H_0$, and for each consecutive state pair $\langle t_i, \ell_i, x_i\rangle$, we

[2]Again, we implicitly assume that the system does not exhibit Zeno behaviors [Lygeros et al. 2003].

— either make discrete transition from ℓ_i to ℓ_{i+1} and set $x_{i+1} = \mathbf{R}_{\langle \ell_i, \ell_{i+1} \rangle}(x_i)$
— or we evolve under the subsystem $\mathbf{D}_{\ell_i}$ from x_i to x_{i+1}, i.e., $x_{i+1} = \mathbf{D}_{\ell_i}(x_i, t_i, u)(t_i + 1)$.

A hybrid system $\mathcal{H}$ is *deterministic* iff starting from some initial state $\langle t_0, \ell_0, x_0 \rangle$ there exists a unique timed trace. Given a timed trace, we can construct a hybrid system trajectory $\mathbf{y} : R \to Y$ by setting $\mathbf{y}(t) = \langle \mathbf{l}(t), \mathbf{z}(t) \rangle$ for $t \in [t_i, t_{i+1})$, where $\mathbf{l}(t) = \ell_i$ and $\mathbf{z}(t) = \mathbf{O}(\mathbf{x}(t))$ with $\mathbf{x}(t) = \mathbf{D}_{\ell_i}(x_i, t_i, u)(t)$. Therefore, again, we may view a hybrid system as a function $\Delta_{\mathcal{H}}$ from the set of initial conditions H_0 and the input signals $\mathbf{U}$ to output signals Y^R.

Let $\tilde{\Delta}_{\mathcal{H}}(h_0, \tilde{u})$ represent the approximate simulation function for a deterministic hybrid system $\mathcal{H}$. We assume that $\tilde{\Delta}_{\mathcal{H}}(h_0, \tilde{u})$ 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 [Esposito and Kumar 2004]. However, assuming that such a simulator is available (see [Sanfelice and Teel 2010] for conditions), we may translate the trace fitness function defined for continuous simulations to hybrid simulations with discrete transitions.

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. For the case of hybrid systems with reset maps, the robustness metrics used in Section 3 cannot be used to compare the hybrid states (ℓ, z) and (ℓ', z') in terms of some norm distance between z and z'. 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 (generalized quasi-) metrics for hybrid automata.

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. A similar metric was used for guiding the exploration in a model checker for hybrid systems in [Alur et al. 2003]. 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 $\mathcal{H}$, we let $\Gamma(\mathcal{H}) = (L, \to)$ represent the directed graph formed by its discrete modes and transitions. The shortest path distance from node ℓ to node ℓ' in the graph $\Gamma(\mathcal{H})$ will be denoted by $\pi(\ell, \ell')$. Note that $\pi(\ell, \ell') = \infty$ iff there is no path from ℓ to ℓ' in the graph $\Gamma(\mathcal{H})$. It is well known (and 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 on-the-fly by running a Breadth First Search (BFS) [Cormen et al. 2001] algorithm on the graph. It is well known that BFS runs in linear time on the size of the input graph. However, it is preferable to use an all-pairs shortest path algorithm [Cormen et al. 2001] to precompute the distances between all pairs of control locations of the hybrid automaton. In our implementation, we are using the Floyd-Warshall algorithm which has running time $\Theta(|L|^3)$.

In order to reason over output trajectories $\mathbf{y}$ in the hybrid state space Y, we need to introduce a generalized distance function [Seda and Hitzler 2008]. In the following, we will denote the hybrid space $L \times Z$ by $\mathbb{H}$ to indicate that a metric is defined over a particular space. Let $\mathbf{d_h} : \mathbb{H} \times \mathbb{H} \to \mathbb{V}_+^\infty$, where $\mathbb{V}_+^\infty = \mathbb{V}_+ \cup \langle +\infty, +\infty \rangle$ and $\mathbb{V}_+ = \mathbb{N} \times \mathbb{R}_+$, with definition for $h = \langle \ell, z \rangle \in \mathbb{H}$ and $h' = \langle \ell', z' \rangle \in \mathbb{H}$,

$$\mathbf{d_h}(h, h') = \begin{cases} \langle 0, d(z, z') \rangle & \text{if } \ell = \ell' \\ \left\langle \pi(\ell, \ell'), \min_{\ell'' \in \partial\mathcal{N}_\pi(\ell, \ell')} \mathbf{dist}_d(z, \mathbf{G}^t(\ell, \ell'')) \right\rangle & \text{otherwise} \end{cases}$$

where π is the shortest path metric, d is a metric on Z and $\partial\mathcal{N}_\pi(\ell, \ell') = \mathbf{Nxt}(\ell) \cap \mathcal{N}_\pi(\ell', \pi(\ell, \ell'))$. Here, $\mathbf{Nxt}(\ell) = \{\ell' \in L \mid \ell \to \ell'\}$ and $\mathbf{G}^t$ denotes that the guard set may be changing with respect to time. Informally, $\partial\mathcal{N}_\pi(\ell, \ell')$ is the “boundary” of all locations which are closer to ℓ' than ℓ and may be visited from ℓ within one transition. Therefore, when the two points h, h' are in the same control location, then the distance computation reduces to the distance computation between the points in the continuous state space. When the two points h, h' are in different control locations, then the distance is the path distance between the two control locations "weighted" by the distance to the closest guard that will enable the transition to the next control location that reduces the path distance. Essentially, the last condition is a heuristic that gives preference to shortest paths.

Next, we need to define an appropriate addition $+$ and a partial order $\preceq$ such that the triplet $(\mathbb{V}_+, +, \preceq)$ is a positively ordered commutative monoid. First, the addition is defined component-wise, that is, for $\langle k, r\rangle, \langle k', r'\rangle \in \mathbb{V}_+$, we define

$$\langle k, r\rangle + \langle k', r'\rangle = \langle k + k', r + r'\rangle$$

The commutativity property is immediately satisfied. Second, we order the set using the dictionary order. Given $\langle k, r\rangle$, $\langle k', r'\rangle \in \mathbb{Z}_\infty \times \overline{\mathbb{R}}$, we define the order relation $\prec$ as

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

It is easy to verify that the dictionary order is compatible with the addition as defined for $\mathbb{V}_+$. Hence, $\mathbb{V}_+$ has a smallest element, namely $\mathbf{0} = \langle 0, 0\rangle$, and $\mathbb{V}_+^\infty$ has an absorbing element, namely $+\infty = \langle +\infty, +\infty\rangle$, which is also the least upper bound. Finlay, Proposition A.1 in the Appendix demonstrates that the generalized distance $\mathbf{d_h}$ satisfies the identity and triangle inequality properties. In other words, $\mathbf{d_h}$ is a generalized quasi-metric on $\mathbb{H}$.

The generalized distance function $\mathbf{d_h}$ requires computations of a point to each guard set in a control location. This may potentially increase the computational load or it could be the case that the computation of the distance to the guard might not be possible (for example, in certain Simulink/Stateflow models). Therefore, we also introduce the generalized distance function $\mathbf{d_h^0} : \mathbb{H} \times \mathbb{H} \to \mathbb{V}_+^\infty$ with definition

$$\mathbf{d_h^0}(h, h') = \begin{cases} \langle 0, d(z, z')\rangle & \text{if } \ell = \ell' \\ \langle \pi(\ell, \ell'), 0\rangle & \text{if } \ell \neq \ell' \text{ and } \pi(\ell, \ell') < +\infty \\ \langle +\infty, +\infty\rangle & \text{otherwise} \end{cases}$$

In this case, the distance function ignores the guard sets and simply checks whether the 2 points are in the same control location or not. The distance function $\mathbf{d_h^0}$ is a generalized quasi-metric as well.

Therefore, we are in position to reason about hybrid system trajectories by utilizing the MTL robustness Definition 2.6 and Theorem 4.2. 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.

Remark 4.5. Note that both functions $\mathbf{Dist}_{\mathbf{d_h}}$ and $\mathbf{Dist}_{\mathbf{d_h^0}}$ never evaluate to some value of the form $\langle k, \pm\infty\rangle$ with $k \in \mathbb{Z}$ (see Proposition A.3). This is important because the temporal logic robustness value is now going to be a member of the set $\mathbb{V}^\infty = \mathbb{V} \cup \{\pm\infty\}$ where $\mathbb{V} = \mathbb{Z} \times \mathbb{R}$. In order for the triplet $(\mathbb{V}, +, \preceq)$ to be an ordered Abelian group and, thus, the robust MTL semantics to have a proper definition of negation, each member of $\mathbb{V}$ must have an inverse. The negation for the MTL robust semantics induced by the aforementioned metrics is simply the pairwise negation. In Proposition A.3, we also demonstrate how the distance functions $\mathbf{Dist}_{\mathbf{d_h}}$ and $\mathbf{Dist}_{\mathbf{d_h^0}}$ can be computed based on the well known understood distance functions $\mathbf{Dist}_\pi$ and $\mathbf{Dist}_d$.

ALGORITHM 2: Parallel Monte-Carlo sampling algorithm.

Input: $H_0 \times \Lambda \times R^m$: Input Space, $f(\cdot)$: Robustness Function, ε: Robustness Threshold, $\mathsf{PS}(\cdot)$: Proposal Scheme
Output: $\langle h, \lambda, \tau \rangle \in H_0 \times \Lambda \times R^m$
begin
  Choose some initial input $\langle h, \lambda, \tau \rangle \in H_0 \times \Lambda \times R^m$;
  while *($f(h, \lambda, \tau) \geq \varepsilon$)* **do**
    `/* Select` $\langle h', \lambda', \tau' \rangle$ `using the Proposal Scheme */`
    $\langle h', \lambda', \tau' \rangle \leftarrow \mathsf{PS}(\langle h, \lambda, \tau \rangle)$;
    $\alpha_1 \leftarrow \mathsf{exp}(-\beta_1(f_1(h', \lambda', \tau') - f_1(h, \lambda, \tau)))$;
    $\alpha_2 \leftarrow \mathsf{exp}(-\beta_2(f_2(h', \lambda', \tau') - f_2(h, \lambda, \tau)))$;
    $r \leftarrow \mathsf{UniformRandomReal}(0, 1)$;
    if $(((f_1(h', \lambda', \tau') = f_1(h, \lambda, \tau)) \wedge (r \leq \alpha_2)) \vee ((f_1(h', \lambda', \tau') \neq f_1(h, \lambda, \tau)) \wedge (r \leq \alpha_1)))$ **then**
      $\langle h, \lambda, \tau \rangle \leftarrow \langle h', \lambda', \tau' \rangle$;
  end
end

4.3. Monte Carlo Sampling

One of the issues that arise when giving generalized (or "hybrid") robust semantics to MTL formulas is how to sample over the space $H_0 \times \Lambda \times R^m$. Recall that $\Lambda \times R^m$ is the space of parameters that parameterize the input signals. In other words, what is the probability distribution induced by the robustness function f? In general, this issue can only be addressed in a case-by-case scenario depending on the generalized metric $\mathbf{d}$ that is utilized.

In this work, for the generalized quasi-metric $\mathbf{d_h}$, we propose to use a Parallel Metropolis coupled Markov chain Monte Carlo algorithm (see Algorithm 2). For a point $\langle h_0, \lambda, \tau \rangle \in H_0 \times \Lambda \times R^m$, the robustness function is now $f(h_0, \lambda, \tau) = \tilde{\mathcal{D}}_\varphi(\tilde{\Delta}_\mathcal{H}(h_0, \tilde{\mathfrak{U}}(\lambda, \tau)))$. If $f(h_0, \lambda, \tau) = \langle k, r \rangle \in \mathbb{V}^\infty$, then we define $f_1(h_0, \lambda, \tau) = k \in \mathbb{Z}_\infty$ and $f_2(h_0, \lambda, \tau) = r \in \overline{\mathbb{R}}$. In brief, in Algorithm 2, an input $\langle h_1, \lambda_1, \tau_1 \rangle$ will be more likely sampled over an input $\langle h_1, \lambda_1, \tau_1 \rangle$, if $f_1(h_1, \lambda_1, \tau_1) = f_1(h_2, \lambda_2, \tau_2)$ and $f_2(h_1, \lambda_1, \tau_1) << f_2(h_2, \lambda_2, \tau_2)$, or, if $f_1(h_1, \lambda_1, \tau_1) \neq f_1(h_2, \lambda_2, \tau_2)$ and $f_1(h_1, \lambda_1, \tau_1) << f_1(h_2, \lambda_2, \tau_2)$. The discussion in Section 3.2 on the importance of β and the proposal schemes still applies. Similarly, we can define a sampling algorithm for the metric $\mathbf{d_h^0}$.

5. EXPERIMENTS

We have implemented our techniques and, in particular, the new metrics inside our Matlab toolbox S-TALIRO [Annapureddy et al. 2011]. Our toolbox is general enough to interact with various means for modeling CPS including Simulink/Stateflow models. We currently support full time bounded MTL for continuous as well as hybrid time trajectories. We remark that all the benchmark problems are distributed with S-TALIRO at `https://sites.google.com/a/asu.edu/s-taliro/` which also includes all the MTL specifications used in this section.

We performed 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 maximum number of 1000 tests, terminating early if a falsifying trajectory is found. Since these techniques are randomized, each experiment was repeated 100 times (runs) 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 and the corresponding running times.

Table I reports on the results of our comparison on two benchmark problems using different MTL properties or problem instances. The first benchmark problem is the Automatic Transmission (AT) model considered in Example 2.1. We consider a number of MTL specifications of increasing diffi-

Table I. Experimental Comparison of Monte-Carlo (MC) vs. Uniform Random (UR) falsification on benchmark problems with Euclidean output spaces. Each instance was run for 100 times and each run was executed for a maximum of 1000 tests. Legend: **#Fals.**: the number of runs falsified, **Robustness**: ⟨min, average, variance⟩ of the runs that were not falsified, **Time:** ⟨min, average, max⟩ time in seconds per run.

Problem	ψ	#Fals.		Robustness		Time (sec)	
		MC	UR	MC	UR	MC	UR
AT	ϕ_1^{AT}	97	100	$\langle 2.54, 7, 48.5\rangle$	-	$\langle 0.2, 11, 92\rangle$	$\langle 0.2, 3, 16\rangle$
AT	ϕ_2^{AT}	96	100	$\langle 3.03, 137, 6.6\cdot 10^4\rangle$	-	$\langle 0.2, 16, 94\rangle$	$\langle 0.2, 10, 48\rangle$
AT	ϕ_3^{AT}	51	0	$\langle 8\cdot 10^{-4}, 0.42, 1.2\rangle$	$\langle 0.04, 0.96, 0.35\rangle$	$\langle 7, 61, 94\rangle$	$\langle 93, 94, 99\rangle$
AT	ϕ_4^{AT}	0	0	$\langle 5.86, 5.95, 0.02\rangle$	$\langle 5.91, 6.06, 0.01\rangle$	$\langle 92, 93, 93\rangle$	$\langle 92, 92, 93\rangle$
AT	ϕ_5^{AT}	0	0	$\langle 0.15, 0.41, 2.55\rangle$	$\langle 0.25, 0.57, 0.06\rangle$	$\langle 93, 93, 94\rangle$	$\langle 92, 93, 94\rangle$
$P^{\Delta-\Sigma}_{[-0.45,0.45]}$	$\phi_{\Delta-\Sigma}$	84	81	$\langle 0.00, 0.04, 4.6\cdot 10^{-4}\rangle$	$\langle 0.00, 0.01, 1.2\cdot 10^{-4}\rangle$	$\langle 0.2, 19, 41\rangle$	$\langle 0.2, 19, 43\rangle$
$P^{\Delta-\Sigma}_{[-0.4,0.4]}$	$\phi_{\Delta-\Sigma}$	58	40	$\langle 0.00, 0.06, 7.9\cdot 10^{-4}\rangle$	$\langle 0.00, 0.03, 2.2\cdot 10^{-4}\rangle$	$\langle 0.7, 26, 39\rangle$	$\langle 0.3, 30, 38\rangle$
$P^{\Delta-\Sigma}_{[-0.35,0.35]}$	$\phi_{\Delta-\Sigma}$	21	1	$\langle 0.00, 0.07\ 2.1\cdot 10^{-3}\rangle$	$\langle 0.01, 0.06\ 7.9\cdot 10^{-4}\rangle$	$\langle 4.1, 35, 49\rangle$	$\langle 5.4, 37, 44\rangle$

Table II. Experimental Comparison of Monte-Carlo (MC) vs. Uniform Random (UR) falsification on benchmark problems with hybrid output spaces. Each instance was run for 100 times and each run was executed for a maximum of 1000 tests. Legend: **#Fals.**: the number of runs falsified, **Time:** ⟨min, average, max⟩ time in seconds per run, **MC-H**: MC with metric $\mathbf{d_h}$, **MC-H0**: MC with metric $\mathbf{d_h^0}$.

Problem	ψ	#Fals.			Time		
		MC-H	MC-H0	UR	MC-H	MC-H0	UR
AT	ϕ_6^{AT}	-	93	86	-	$\langle 0.4, 24, 138\rangle$	$\langle 0.4, 56, 139\rangle$
AT	ϕ_7^{AT}	-	94	55	-	$\langle 0.1, 25, 128\rangle$	$\langle 0.6, 81, 127\rangle$
AT	ϕ_8^{AT}	-	0	0	-	$\langle 110, 115, 139\rangle$	$\langle 109, 111, 115\rangle$
$\mathrm{NV}_{[0,25]}$	ϕ_1^{NV}	63	68	34	$\langle 4.2, 542, 831\rangle$	$\langle 34, 545, 865\rangle$	$\langle 44, 623, 817\rangle$
$\mathrm{NV}_{[0,12]}$	ϕ_2^{NV}	100	100	100	$\langle 1.1, 24, 140\rangle$	$\langle 1.7, 25, 168\rangle$	$\langle 0.9, 22, 108\rangle$
$\mathrm{NV}_{[0,12]}$	ϕ_3^{NV}	100	100	100	$\langle 0.8, 8.7, 62\rangle$	$\langle 0.8, 17, 503\rangle$	$\langle 0.7, 4.0, 22\rangle$
$\mathrm{NV}_{[0,12]}$	ϕ_4^{NV}	100	100	100	$\langle 1.2, 18, 85\rangle$	$\langle 1.4, 26, 66\rangle$	$\langle 0.8, 35, 427\rangle$
$\mathrm{NV}_{[0,12]}$	ϕ_5^{NV}	38	47	5	$\langle 21.0, 419, 595\rangle$	$\langle 15, 390, 584\rangle$	$\langle 9.4, 404, 437\rangle$

culty to falsify. As an example, formula ϕ_1^{AT} is described in Example 2.7. The second benchmark is a Simulink model of a 3rd order $\Delta-\Sigma$ modulator whose description can be found in [Dang et al. 2004]. The 3rd order $\Delta-\Sigma$ modulator has unknown initial conditions in the set $[-0.1, 0.1]^3$ and a one dimensional input signal that takes values in a set $[u_m, u_M]$. The problem instances in Table I indicate the bounds on the input signal $[u_m, u_M]$. The specification for the $\Delta-\Sigma$ modulator is that the state of the system should always remain in the set $[-1, 1]^3$.

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. If the property can be easily falsified, then it is advantageous to utilize uniform random search. MC for easy problem instances seems to converge and get trapped at local minima. In practice, we may periodically reset the MC simulation using random restarts. However, such restarts were not used in our experimental comparison. The use of MC is clearly advantageous when the problem is challenging. In hard problem instances, MC can falsify the specification when UR fails to falsify. Moreover, even when falsification fails, MC still computes lower minimum and average robustness values with the same computational cost. Further experimental results that attest the same conclusions can be found in [Nghiem et al. 2010].

Table II compares the performance of the falsification algorithm on benchmark problems with hybrid output space. We compared UR with MC on two benchmark problems on various temporal logic formulas of increasing difficulty to falsify. The first benchmark problem was AT. As opposed to the previous experiments, the specifications now not only place conditions on the continuous state of the system, but also on the discrete locations. As an example, formula ϕ_6^{AT} is informally described in Example 4.1. Since S-TALIRO does not support yet automatic extraction of guard conditions, we compared only UR with MC using the metric $\mathbf{d_h^0}$ for the distance computations.

The second example that we consider is the Navigation (NV) benchmark problem from [Fehnker and Ivančić 2004]. This is a hybrid automaton benchmark problem and both the control locations and the guards of the transitions are available to us. Thus, we compared the performance of the Monte Carlo sampling algorithm under the metrics $\mathbf{d_h}$ and $\mathbf{d_h^0}$ with the performance of Uniform Random sampling under the $\mathbf{d_h}$ metric. The problem instance that is used in our experiments is presented in [Nghiem et al. 2010].

First, we observe that on easy problem instances, i.e., ϕ_2^{NV}-ϕ_4^{NV}, the performance of all algorithms is comparable in terms of computation time. On hard problem instances, both MC-H and MC-H0 outperform UR in terms of numbers of falsifications.

The experimental results indicate that the best way to approach hybrid system falsification / verification is with a layered approach. Assuming that at the initial design stages the errors are abundant, then it is preferable to run random sampling for the falsification process. As the system design becomes more mature, then Monte Carlo sampling with the new metrics introduced in this paper can be utilized for the falsification. When the level of confidence in the system design has increased and potentially the system design is robust enough, then the designer may use a reachability analysis algorithm (for example SpaceEx [Frehse et al. 2011]). However, we remark that currently reachability analysis tools cannot handle arbitrary MTL specifications. A more detailed discussion on system verification that compares the advantages/disadvantages of falsification and reachability methods can be found in [Abbas and Fainekos 2011a; 2011b].

6. RELATED WORK

Due to the known undecidability results in the analysis of hybrid systems [Alur et al. 1995] and the state explosion problem of the reachability computation algorithms (see [Julius et al. 2007] for some related references), a lot of recent research activity has concentrated on testing approaches to the verification of continuous and hybrid systems [Kapinski et al. 2003; Zhao et al. 2003].

The use of Monte Carlo techniques for model checking has been considered previously by Grosu and Smolka [Grosu and Smolka 2005]. 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 [Sankaranarayanan et al. 2007]. However, our previous efforts did not have the 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 [Esposito et al. 2004; Bhatia and Frazzoli 2004; Branicky et al. 2006; Nahhal and Dang 2007; Plaku et al. 2007]. 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 [Donzé and Maler 2007; Girard and Pappas 2006; Julius et al. 2007; Lerda et al. 2008]. In robust testing, a simulation trajectory can represent a neighborhood of trajectories achieving, thus, better coverage guarantees. Recently, the authors in [Dang et al. 2008] 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 [Plaku et al. 2009; Rizk et al. 2008; Fainekos et al. 2006]. The work that is the closest to ours appears in [Rizk et al. 2008]. 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., [Rizk et al. 2008] 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). SMC targets stochastic system models such as continuous-time Markov chains [Younes and Simmons 2006] or Stochastic Hybrid Automata (SHA) [Clarke et al. 2009]. For example, in order to model imperfect sensors in Example 2.1, we may add Gaussian noise to the sensor that reads the engine speed. Then, the resulting system would be a SHA. The goal of SMC is to asses the probability that a system satisfies a given *probabilistic* temporal logic property φ. This probability can be safely approximated using Wald's probabilistic ratio test. SMC, 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 whether our algorithm has converged to the global minimum of the temporal logic robustness function. However, this is an on-going endeavor.

Remark 6.1. Our method does not try to assess the probability of failure, but to detect a failure. That is, our goal is to provide the engineer with tools in order to detect design problems in the system rather than perform a failure analysis. In our framework, if a failure is detected, then the designer has a counterexample to work with in order to "debug" the system. Moreover, if a failure is not detected, then the designer is still provided with the least robust behavior found. The fact that the system might be correct with probability one does not imply that the system is *robustly* correct. Therefore, we view SMC and our approach as complementary. In an MBD cycle, the model should be first assessed for its robustly correct behavior, and, then, a failure analysis should be performed under various failure models and requirements.

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 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 on the hard 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. The techniques and the methods that were introduced in this paper have been implemented in our Matlab toolbox S-TALIRO [Annapureddy et al. 2011].

ACKNOWLEDGMENTS

The authors would like to thank Truong Nghiem and Professor George Pappas for the useful discussions and the reviewers for the very careful reading of the manuscript and their numerous comments which have improved the paper.

REFERENCES

ABBAS, H. AND FAINEKOS, G. 2011a. Linear hybrid system falsification through descent. Technical Report arXiv:1105.1733.

ABBAS, H. AND FAINEKOS, G. 2011b. Linear hybrid system falsification through local search. In *Automated Technology for Verification and Analysis*. LNCS Series, vol. 6996. Springer, 503–510.

ALUR, R., COURCOUBETIS, C., HALBWACHS, N., HENZINGER, T. A., HO, P.-H., NICOLLIN, X., OLIVERO, A., SIFAKIS, J., AND YOVINE, S. 1995. The algorithmic analysis of hybrid systems. *Theoretical Computer Science 138,* 1, 3–34.

ALUR, R., DANG, T., ESPOSITO, J., HUR, Y., IVANCIC, F., KUMAR, V., LEE, I., MISHRA, P., PAPPAS, G. J., AND SOKOLSKY, O. 2003. Hierarchical modeling and analysis of embedded systems. *Proceedings of the IEEE 91,* 1, 11–28.

ALUR, R., DANG, T., AND IVANČIĆ, F. 2003. Progress on reachability analysis of hybrid systems using predicate abstraction. In *Hybrid Systems: Computation and Control*. LNCS Series, vol. 2623. Springer, 4–19.

ALUR, R., HENZINGER, T. A., LAFFERRIERE, G., AND PAPPAS, G. J. 2000. Discrete abstractions of hybrid systems. *Proceedings of the IEEE 88,* 2, 971–984.

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

ANNAPUREDDY, Y. S. R., LIU, C., FAINEKOS, G. E., AND SANKARANARAYANAN, S. 2011. S-taliro: A tool for temporal logic falsification for hybrid systems. In *Tools and algorithms for the construction and analysis of systems*. LNCS Series, vol. 6605. Springer, 254–257.

BANDEMER, H. AND GOTTWALD, S. 1995. *Fuzzy sets, fuzzy logic, fuzzy methods, with applications.* Wiley.

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

BOYD, S. AND VANDENBERGHE, S. 2004. *Convex Optimization*. Cambridge University Press. cf. `http://www.stanford.edu/~boyd/cvxbook.html`.

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

CHIB, S. AND GREENBERG, E. 1995. Understanding the Metropolis-Hastings algorithm. *The American Statistician 49,* 4, 327–335.

CLARKE, E., DONZE, A., AND LEGAY, A. 2009. 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*. Lecture Notes in Computer Science Series, vol. 5394/2009. 149–163.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. 2001. *Introduction to Algorithms* Second Ed. MIT Press/McGraw-Hill.

DANG, T., DONZÉ, A., AND MALER, O. 2004. Verification of analog and mixed-signal circuits using hybrid system techniques. In *5th International Conference on Formal Methods in Computer-Aided Design*. LNCS Series, vol. 3312. Springer, 21–36.

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

DE ALFARO, L., FAELLA, M., AND STOELINGA, M. 2004. Linear and branching metrics for quantitative transition systems. In *Proceedings of the 31st ICALP*. LNCS Series, vol. 3142. Springer, 97–109.

DONZÉ, A. AND MALER, O. 2007. Systematic simulation using sensitivity analysis. In *HSCC*. LNCS Series, vol. 4416. Springer, 174–189.

EGERSTEDT, M. AND MARTIN, C. 2009. *Control Theoretic Splines: Optimal Control, Statistics, and Path Planning*. Princeton University Press.

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

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

ESTEREL TECHNOLOGIES. 2011. Scade success stories. Online. `http://www.esterel-technologies.com/technology/success-stories/`.

FAINEKOS, G. E., GIRARD, A., AND PAPPAS, G. J. 2006. Temporal logic verification using simulation. In *FORMATS*. LNCS Series, vol. 4202. Springer, 171–186.

FAINEKOS, G. E. AND PAPPAS, G. J. 2006. Robustness of temporal logic specifications for finite state sequences in metric spaces. Tech. Rep. MS-CIS-06-05, Dept. of CIS, Univ. of Pennsylvania. May.

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

FAINEKOS, G. E., SANKARANARAYANAN, S., IVANČIĆ, F., AND GUPTA, A. 2009. Robustness of model-based simulations. In *IEEE Real-Time Systems Symposium*. IEEE Press, 345–354.

FEHNKER, A. AND IVANČIĆ, F. 2004. Benchmarks for hybrid systems verification. In *HSCC*. LNCS Series, vol. 2993. springer, 326–341.

FREHSE, G., GUERNIC, C. L., DONZ, A., COTTON, S., RAY, R., LEBELTEL, O., RIPADO, R., GIRARD, A., DANG, T., AND MALER, O. 2011. Spaceex: Scalable verification of hybrid systems. In *Proceedings of the 23d CAV*.

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

GIRARD, A. AND PAPPAS, G. J. 2006. Verification using simulation. In *HSCC*. LNCS Series, vol. 3927. Springer, 272 – 286.

GROSU, R. AND SMOLKA, S. 2005. Monte carlo model checking. In *TACAS*. LNCS Series, vol. 3440. Springer, 271–286.

HENZINGER, T. A. 1996. The theory of hybrid automata. In *Logic In Computer Science (LICS 1996)*. IEEE Computer Society Press, 278–292.

HENZINGER, T. A., KOPKE, P. W., PURI, A., AND VARAIYA, P. 1998. What's decidable about hybrid automata? *J. Comput. Syst. Sci. 57,* 1, 94–124.

JULIUS, A. A., FAINEKOS, G. E., ANAND, M., LEE, I., AND PAPPAS, G. J. 2007. Robust test generation and coverage for hybrid systems. In *HSCC*. Number 4416 in LNCS. Springer, 329–342.

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

KOPPERMAN, R. 1988. All topologies come from generalized metrics. *Amer. Math. Monthly 95*, 89–97.

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

LAMINE, K. B. AND KABANZA, F. 2000. Using fuzzy temporal logic for monitoring behavior-based mobile robots. In *Proceedings of the IASTED International Conference Robotics and Applications*, M. Hamza, Ed. IASTED/ACTA Press, 116122.

LEE, E. A. AND VARAIYA, P. 2003. *Structure and Interpretation of Signals and Systems*. Addison Wesley.

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

LOVASZ, L. 1999. Hit-and-run is fast and run. *Mathematical Programming 86*, 443–461.

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

LYGEROS, J., JOHANSSON, K. H., SIMIC, S. N., ZHANG, J., AND SASTRY, S. 2003. Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control 48*, 2–17.

MATHWORKS. 2011. Simulink user stories. Online. `http://www.mathworks.com/products/simulink/userstories.html`.

NAHHAL, T. AND DANG, T. 2007. Test coverage for continuous and hybrid systems. In *CAV*. LNCS Series, vol. 4590. Springer, 449–462.

NGHIEM, T., SANKARANARAYANAN, S., FAINEKOS, G. E., IVANCIC, F., GUPTA, A., AND PAPPAS, G. J. 2010. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*. ACM Press, 211–220.

PLAKU, E., KAVRAKI, L. E., AND VARDI, M. Y. 2007. Hybrid systems: From verification to falsification. In *CAV*. LNCS Series, vol. 4590. Springer, 463–476.

PLAKU, E., KAVRAKI, L. E., AND VARDI, M. Y. 2009. Falsification of ltl safety properties in hybrid systems. In *TACAS*. LNCS Series, vol. 5505. Springer, 368 – 382.

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

RANDALL, D. 2006. Rapidly mixing markov chains with applications in computer science and physics. *Computing in Science and Engineering 8,* 2.

RIZK, A., BATT, G., FAGES, F., AND SOLIMAN, S. 2008. 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*. Number 5307 in LNCS. Springer, 251–268.

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

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

SANFELICE, R. G. AND TEEL, A. R. 2010. Dynamical properties of hybrid systems simulators. *Automatica 46,* 2, 239–248.

SANKARANARAYANAN, S., CHANG, R. M., JIANG, G., AND IVANCIC, F. 2007. State space exploration using feedback constraint generation and monte-carlo sampling. In *ESEC/SIGSOFT FSE*. ACM, 321–330.

SEDA, A. K. AND HITZLER, P. 2008. Generalized distance functions in the theory of computation. *The Computer Journal 53,* 4, bxm108443–464.

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

SMITH, R. L. 1996. 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*. IEEE Computer Society, 260–264.

SONTAG, E. D. 1998. *Mathematical Control Theory: Deterministic Finite Dimensional Systems* 2nd Ed. Springer.

TRIPAKIS, S. AND DANG, T. 2009. *Model-Based Design for Embedded Systems*. CRC Press, Chapter Modeling, Verification and Testing using Timed and Hybrid Automata, 383–436.

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

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

ZHAO, Q., KROGH, B. H., AND HUBBARD, P. 2003. Generating test inputs for embedded control systems. *IEEE Control Systems Magazine Aug.*, 49–57.

APPENDIX

PROOF OF THEOREM 4.2. The proof is by induction on the structure of the formula.

(1) We will present only the base cases, since the other cases are identical with those in the proofs in [Fainekos and Pappas 2009] and [Fainekos and Pappas 2006].
- If $[\![p, \mathcal{O}]\!]_{\mathbf{d}}(\mathbf{y}, t) \succ \mathbf{0}$, then by definition $\mathbf{Dist}_{\mathbf{d}}(\mathbf{y}(t), \mathcal{O}(p)) \succ \mathbf{0}$, which implies that $\mathbf{y}(t) \in \mathcal{O}(p)$ and, thus, that $(\mathbf{y}, t, \mathcal{O}) \models p$.
- If $(\mathbf{y}, t, \mathcal{O}) \models p$, then by definition $\mathbf{y}(t) \in \mathcal{O}(p)$, which implies that $\mathbf{depth}_{\mathbf{d}}(\mathbf{y}(t), \mathcal{O}(p)) = \mathbf{Dist}_{\mathbf{d}}(\mathbf{y}(t), \mathcal{O}(p)) \succeq \mathbf{0}$, and, thus, that $[\![\phi, \mathcal{O}]\!]_{\mathbf{d}}(\mathbf{y}, t) \succeq \mathbf{0}$.

Note that the equality in the first case fails when the signal value $\mathbf{y}(t)$ is right on the boundary of the set $\mathcal{O}(p)$, i.e., $\mathbf{y}(t) \in \partial\mathcal{O}(p)$. If $[\![p, \mathcal{O}]\!]_{\mathbf{d}}(\mathbf{y}, t) = \mathbf{0}$, then we cannot distinguish whether $(\mathbf{y}, t, \mathcal{O}) \models p$ or $(\mathbf{y}, t, \mathcal{O}) \not\models p$.

(2) Similar to the previous proof.

(3) We will present the base case and the negation (the other cases are based on the definition of supremum and infimum over the partial order $\preceq$ of $\mathbf{d}$ and are similar to the negation).
- **Base case:**
  - If $[\![p, \mathcal{O}]\!]_{\mathbf{d}}(\mathbf{y}, t) = \varepsilon \succ \mathbf{0}$, then $(\mathbf{y}, t, \mathcal{O}) \models p$ and by definition $\mathbf{depth}_{\mathbf{d}}(\mathbf{y}(t), \mathcal{O}(p)) = \varepsilon \succ \mathbf{0}$, which implies that $\mathcal{B}_{\mathbf{d}}(\mathbf{y}(t), \varepsilon) \subseteq \mathcal{O}(p)$. Since $\mathbf{y}' \in \mathcal{B}_{\rho_{\mathbf{d}}}(\mathbf{y}, \varepsilon)$, we have $\rho_{\mathbf{d}}(\mathbf{y}, \mathbf{y}') = \sup_{t \in R} \mathbf{d}(\mathbf{y}(t), \mathbf{y}'(t)) \prec \varepsilon$. That is, $\mathbf{d}(\mathbf{y}(t), \mathbf{y}'(t)) \prec \varepsilon$ and, thus, $\mathbf{y}'(t) \in \mathcal{B}_{\mathbf{d}}(\mathbf{y}(t), \varepsilon) \subseteq \mathcal{O}(p)$. Hence, $(\mathbf{y}', t, \mathcal{O}) \models p$.
  - Similar to the previous case.
- **Negation:**
  - Positive case: If $[\![\neg\phi, \mathcal{O}]\!]_{\mathbf{d}}(\mathbf{y}, t) = \varepsilon \succ \mathbf{0}$, then (i) $(\mathbf{y}, t, \mathcal{O}) \models \neg\phi$, i.e., $(\mathbf{y}, t, \mathcal{O}) \not\models \phi$, and (ii) $[\![\phi, \mathcal{O}]\!]_{\mathbf{d}}(\mathbf{y}, t) = -\varepsilon \prec \mathbf{0}$. Then, by (ii) and the induction hypothesis we have that for all $\mathbf{y}' \in \mathcal{B}_{\rho_{\mathbf{d}}}(\mathbf{y}, \varepsilon)$, $(\mathbf{y}', t, \mathcal{O}) \not\models \phi$.
  - Negative case: Similar to the previous case.

We chose to present negation in order to demonstrate the properties that the negation must satisfy. □

PROPOSITION A.1. *The generalized distance function* $\mathbf{d_h}$ *is a quasi-metric.*

PROOF. We will need to demonstrate that the identity property and the triangle inequality hold. In the following, we let $h_i = (\ell_i, z_i) \in \mathbb{H}$ with $i = 1, 2, 3$.

Identity: Since π is a quasi-metric, we have $\pi(\ell_1, \ell_2) = 0$ iff $\ell_1 = \ell_2$. Since d is a metric, we have $d(z_1, z_2) = 0$ iff $z_1 = z_2$. Hence, $\mathbf{d_h}(h_1, h_2) = \langle 0, 0 \rangle$ iff $h_1 = h_2$.

Triangle Inequality: We need to show that for all $h_1, h_2, h_3 \in \mathbb{H}$, $\mathbf{d_h}(h_1, h_2) \leq \mathbf{d_h}(h_1, h_3) + \mathbf{d_h}(h_3, h_2)$. We proceed by case by case analysis:

(1) **Case** $\ell_1 = \ell_2 = \ell_3$**:** Then,

$$\begin{aligned}\mathbf{d_h}(h_1, h_2) &= \langle 0, d(z_1, z_2)\rangle \leq \langle 0, d(z_1, z_3) + d(z_3, z_2)\rangle \\ &= \langle 0, d(z_1, z_3)\rangle + \langle 0, d(z_3, z_2)\rangle = \mathbf{d_h}(h_1, h_3) + \mathbf{d_h}(h_3, h_2)\end{aligned}$$

(2) **Case** $\ell_1 = \ell_2 \neq \ell_3$**:** Then, $\pi(\ell_1, \ell_3) > 0$ and $\pi(\ell_3, \ell_2) > 0$ and

$$\begin{aligned}&\mathbf{d_h}(h_1, h_2) = \langle 0, d(z_1, z_2)\rangle \leq \langle \pi(\ell_1, \ell_3) + \pi(\ell_3, \ell_2), 0\rangle \leq \langle \pi(\ell_1, \ell_3), 0\rangle + \langle \pi(\ell_3, \ell_2), 0\rangle \\ &\leq \left\langle \pi(\ell_1, \ell_3), \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_3)} \mathbf{dist}_d(z_1, \mathbf{G}^t(\ell_1, \ell))\right\rangle + \left\langle \pi(\ell_3, \ell_2), \min_{\ell \in \partial\mathcal{N}_\pi(\ell_3,\ell_2)} \mathbf{dist}_d(z_3, \mathbf{G}^t(\ell_3, \ell))\right\rangle \\ &= \mathbf{d_h}(h_1, h_3) + \mathbf{d_h}(h_3, h_2)\end{aligned}$$

(3) **Case** $\ell_1 \neq \ell_2$ **and** $\ell_1 = \ell_3$**:** Then,

$$\mathbf{d_h}(h_1, h_2) = \left\langle \pi(\ell_1, \ell_2), \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_2)} \mathbf{dist}_d(z_1, \mathbf{G}^t(\ell_1, \ell))\right\rangle$$

But, $\pi(\ell_1, \ell_2) = 0 + \pi(\ell_3, \ell_2) = \pi(\ell_1, \ell_3) + \pi(\ell_3, \ell_2)$, and, also,

$$\begin{aligned}\min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_2)} \mathbf{dist}_d(z_1, \mathbf{G}^t(\ell_1, \ell)) &= \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_2)} \inf\{d(z_1, z) \mid z \in \mathbf{G}^t(\ell_1, \ell))\} \\ &\leq \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_2)} \inf\{d(z_1, z_3) + d(z_3, z) \mid z \in \mathbf{G}^t(\ell_1, \ell)\} \\ &= d(z_1, z_3) + \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_2)} \inf\{d(z_3, z) \mid z \in \mathbf{G}^t(\ell_1, \ell)\} \\ &\overset{(\ell_1=\ell_3)}{=} d(z_1, z_3) + \min_{\ell \in \partial\mathcal{N}_\pi(\ell_3,\ell_2)} \mathbf{dist}_d(z_3, \mathbf{G}^t(\ell_3, \ell))\end{aligned}$$

$$\begin{aligned}\text{Thus, } \mathbf{d_h}(h_1, h_2) &= \left\langle \pi(\ell_1, \ell_2), \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_2)} \mathbf{dist}_d(z_1, \mathbf{G}^t(\ell_1, \ell))\right\rangle \\ &\leq \left\langle \pi(\ell_1, \ell_3) + \pi(\ell_3, \ell_2), d(z_1, z_3) + \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_2)} \mathbf{dist}_d(z_3, \mathbf{G}^t(\ell_3, \ell))\right\rangle \\ &= \left\langle \cancelto{0}{\pi(\ell_1, \ell_3),} d(z_1, z_3)\right\rangle + \left\langle \pi(\ell_3, \ell_2), \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_2)} \mathbf{dist}_d(z_3, \mathbf{G}^t(\ell_3, \ell))\right\rangle \\ &= \mathbf{d_h}(h_1, h_3) + \mathbf{d_h}(h_3, h_2)\end{aligned}$$

(4) **Case** $\ell_1 \neq \ell_2$ **and** $\ell_2 = \ell_3$**:** Then,

$$\mathbf{d_h}(h_1, h_2) = \left\langle \pi(\ell_1, \ell_2), \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_2)} \mathbf{dist}_d(z_1, \mathbf{G}^t(\ell_1, \ell))\right\rangle$$

But, $\pi(\ell_1, \ell_2) = \pi(\ell_1, \ell_3) + 0 = \pi(\ell_1, \ell_3) + \pi(\ell_3, \ell_2)$, and, also,

$$\begin{aligned}\min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_2)} \mathbf{dist}_d(z_1, \mathbf{G}^t(\ell_1, \ell)) &\overset{(\ell_2=\ell_3)}{=} \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_3)} \mathbf{dist}_d(z_1, \mathbf{G}^t(\ell_1, \ell)) \\ &\leq \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_3)} \mathbf{dist}_d(z_1, \mathbf{G}^t(\ell_1, \ell)) + d(z_3, z_2)\end{aligned}$$

since $d(z_3, z_2) \geq 0$. Thus,

$$\begin{aligned}\mathbf{d_h}(h_1, h_2) &= \left\langle \pi(\ell_1, \ell_2), \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_2)} \mathbf{dist}_d(z_1, \mathbf{G}^t(\ell_1, \ell))\right\rangle \\ &\leq \left\langle \pi(\ell_1, \ell_3) + \pi(\ell_3, \ell_2), \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_3)} \mathbf{dist}_d(z_1, \mathbf{G}^t(\ell_1, \ell)) + d(z_3, z_2)\right\rangle\end{aligned}$$

$$= \left\langle \pi(\ell_1, \ell_3), \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_3)} \mathbf{dist}_d(z_1, \mathbf{G}^t(\ell_1, \ell)) \right\rangle + \left\langle \cancelto{0}{\pi(\ell_3, \ell_2),} d(z_3, z_2) \right\rangle$$
$$= \mathbf{d_h}(h_1, h_3) + \mathbf{d_h}(h_3, h_2)$$

(5) **Case $\ell_1 \neq \ell_2$, $\ell_1 \neq \ell_3$ and $\ell_2 \neq \ell_3$:** Then, $\pi(\ell_1, \ell_2) \leq \pi(\ell_1, \ell_3) + \pi(\ell_3, \ell_2)$, and

$$\mathbf{d_h}(h_1, h_2) = \left\langle \pi(\ell_1, \ell_2), \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_2)} \mathbf{dist}_d(z_1, \mathbf{G}^t(\ell_1, \ell)) \right\rangle$$
$$\leq \langle \pi(\ell_1, \ell_3) + \pi(\ell_3, \ell_2), 0\rangle = \langle \pi(\ell_1, \ell_3), 0\rangle + \langle \pi(\ell_3, \ell_2), 0\rangle$$
$$\leq \left\langle \pi(\ell_1, \ell_3), \min_{\ell \in \partial\mathcal{N}_\pi(\ell_1,\ell_3)} \mathbf{dist}_d(z_1, \mathbf{G}^t(\ell_1, \ell)) \right\rangle + \left\langle \pi(\ell_3, \ell_2), \min_{\ell \in \partial\mathcal{N}_\pi(\ell_3,\ell_2)} \mathbf{dist}_d(z_3, \mathbf{G}^t(\ell_3, \ell)) \right\rangle$$
$$= \mathbf{d_h}(h_1, h_3) + \mathbf{d_h}(h_3, h_2) \quad \square$$

PROPOSITION A.2. *The generalized distance function $\mathbf{d_h^0}$ is a quasi-metric.*

PROOF. The proof is similar to the proof of Proposition A.1. $\square$

PROPOSITION A.3. *Let the current point be $h = \langle \ell, z\rangle$ and $\mathcal{O}(p) = L_p \times Z_p$, then $\mathbf{Dist}_{\mathbf{d_h^0}}(h, \mathcal{O}(p)) \neq \langle k, \pm\infty\rangle$ for any $k \in \mathbb{Z}$. Similarly for $\mathbf{Dist}_{\mathbf{d_h}}(h, \mathcal{O}(p))$.*

PROOF. Actually, we will show that $\mathbf{Dist}_{\mathbf{d_h^0}}(h, \mathcal{O}(p)) = \langle k, \pm\infty\rangle$ iff $k = \pm\infty$.

(1) $h \notin O(p)$ and $\ell \notin L_p$ and if L_p is not reachable from ℓ, then for any $\ell' \in L_p$, we have $\pi(\ell, \ell') = +\infty$. Thus, $\partial\mathcal{N}_\pi(\ell, \ell') = \emptyset$ and $\min_{\ell'' \in \partial\mathcal{N}_\pi(\ell,\ell')} \mathbf{dist}_d(z, \mathbf{G}^t(\ell, \ell'')) = +\infty$. Hence, $\mathbf{Dist}_{\mathbf{d_h}}(h, \mathcal{O}(p)) = -\mathbf{dist}_{\mathbf{d_h}}(h, \mathcal{O}(p)) = \langle -\infty, -\infty\rangle$. Also, $\mathbf{Dist}_{\mathbf{d_h^0}}(h, \mathcal{O}(p)) = \langle -\infty, -\infty\rangle$ by definition.

(2) If $h \notin O(p)$ and $\ell \notin L_p$ and if L_p is reachable from ℓ, then $\partial\mathcal{N}_\pi(\ell, \ell') \neq \emptyset$ since (i) at least one of the neighbors of ℓ will have distance to L_p less than $\mathbf{dist}_\pi(\ell, L_p)$ and (ii) we have assumed that $\mathbf{G}^t(\ell, \ell') \neq \emptyset$ for all $\ell' \in \mathbf{Nxt}(\ell)$. Then, $\mathbf{dist}_d(z, \mathbf{G}^t(\ell, \ell'')) < +\infty$ for all $\ell'' \in \partial\mathcal{N}_\pi(\ell, \ell')$. Let $\ell^* \in \arg\min\{\langle \pi(\ell, \ell'), \min_{\ell'' \in \partial\mathcal{N}_\pi(\ell,\ell')} \mathbf{dist}_d(z, \mathbf{G}^t(\ell, \ell''))\rangle \mid \ell' \in L_p\}$ and set $\delta^* = \min_{\ell'' \in \partial\mathcal{N}_\pi(\ell,\ell^*)} \mathbf{dist}_d(z, \mathbf{G}^t(\ell, \ell'')) < +\infty$. Therefore, $\mathbf{Dist}_{\mathbf{d_h}}(h, \mathcal{O}(p)) = -\mathbf{dist}_{\mathbf{d_h}}(h, \mathcal{O}(p)) = \langle -\pi(\ell, \ell^*), -\delta^*\rangle$. Finally, by definition, we have $\mathbf{Dist}_{\mathbf{d_h^0}}(h, \mathcal{O}(p)) = -\mathbf{dist}_{\mathbf{d_h}}(h, \mathcal{O}(p)) = \langle -\mathbf{dist}_\pi(\ell, L_p), 0\rangle$.

(3) If $h \notin O(p)$, but $\ell \in L_p$, i.e., $z \in Z_p$, then $\mathbf{Dist}_{\mathbf{d_h}}(h, \mathcal{O}(p)) = -\mathbf{dist}_{\mathbf{d_h}}(h, \mathcal{O}(p)) = -\mathbf{dist}_{\mathbf{d_h}}(h, (L_p\backslash\{\ell\} \times Z_p) \cup (\{\ell\} \times Z_p)) = -\min\{\mathbf{dist}_{\mathbf{d_h}}(h, L_p\backslash\{\ell\} \times Z_p), \mathbf{dist}_{\mathbf{d_h}}(h, \{\ell\} \times Z_p)\} = -\mathbf{dist}_{\mathbf{d_h}}(h, \{\ell\} \times Z_p)\} = \langle 0, -\mathbf{dist}_d(z, Z_p)\rangle$. However, $\mathbf{dist}_d(z, Z_p) < +\infty$ since $\emptyset \subset \mathcal{O}(p) \subset Y$ by assumption. Similarly for $\mathbf{Dist}_{\mathbf{d_h^0}}(h, \mathcal{O}(p))$.

(4) If $h \in O(p)$ and $Z_p \subset Z$, then $\mathbf{Dist}_{\mathbf{d_h}}(h, \mathcal{O}(p)) = \mathbf{depth}_{\mathbf{d_h}}(h, \mathcal{O}(p)) = \mathbf{dist}_{\mathbf{d_h}}(h, Y\backslash\mathcal{O}(p)) = \mathbf{dist}_{\mathbf{d_h}}(h, ((L\backslash L_p) \times Z) \cup (L \times (Z\backslash Z_p))) = \min\{\mathbf{dist}_{\mathbf{d_h}}(h, (L\backslash L_p) \times Y), \mathbf{dist}_{\mathbf{d_h}}(h, L \times (Z\backslash Z_p))\} = \mathbf{dist}_{\mathbf{d_h}}(h, L \times (Z\backslash Z_p))\} = \langle 0, \mathbf{dist}_d(z, Z_p)\rangle$ since $\ell \in L_p \subseteq L$. However, $\mathbf{dist}_d(z, Z_p) < +\infty$ since $\emptyset \subset \mathcal{O}(p) \subset Y$ by assumption. Similarly for $\mathbf{Dist}_{\mathbf{d_h^0}}(h, \mathcal{O}(p))$.

(5) If $h \in O(p)$ and $Z_p = Z$, i.e., $L_p \subset L$, then $\mathbf{Dist}_{\mathbf{d_h}}(h, \mathcal{O}(p)) = \mathbf{depth}_{\mathbf{d_h}}(h, \mathcal{O}(p)) = \mathbf{dist}_{\mathbf{d_h}}(h, Y\backslash\mathcal{O}(p)) = \mathbf{dist}_{\mathbf{d_h}}(h, ((L\backslash L_p) \times Z) \cup (L \times (Z\backslash Z_p))) = \mathbf{dist}_{\mathbf{d_h}}(h, ((L\backslash L_p) \times Z) \cup (L \times \emptyset)) = \mathbf{dist}_{\mathbf{d_h}}(h, (L\backslash L_p) \times Z)\}$. Now, we have two cases:
— if $L\backslash L_p$ is reachable from ℓ, then as in case (2), we have $\mathbf{Dist}_{\mathbf{d_h}}(h, \mathcal{O}(p)) = \langle \pi(\ell, \ell^*), \delta^*\rangle$ with $\delta^* < +\infty$.
— if $L\backslash L_p$ is not reachable from ℓ, then $\mathbf{dist}_\pi(\ell, L\backslash L_p) = +\infty$ and as in case (1), we also have $\partial\mathcal{N}_\pi(\ell, \ell') = \emptyset$ for all $\ell' \in L\backslash L_p$. Thus, $\mathbf{Dist}_{\mathbf{d_h}}(h, \mathcal{O}(p)) = \langle +\infty, +\infty\rangle$.

Similarly, we can derive the value of $\mathbf{Dist}_{\mathbf{d_h^0}}(h, \mathcal{O}(p))$.

This concludes the proof since we have considered all possible cases. $\square$