Requirements driven falsification with coverage metrics

Adel Dokhanchi
Arizona State University
adokhanc@asu.edu

Aditya Zutshi
University of Colorado, Boulder
aditya.zutshi@colorado.edu

Rahul T. Sriniva
Arizona State University
rthekkal@asu.edu

Sriram Sankaranarayanan
University of Colorado, Boulder
srirams@colorado.edu

Georgios Fainekos
Arizona State University
fainekos@asu.edu

ABSTRACT

Specification guided falsification methods for hybrid systems have recently demonstrated their value in detecting design errors in models of safety critical systems. In specification guided falsification, the correctness problem, i.e., does the system satisfy the specification, is converted into an optimization problem where local negative minima indicate design errors. Due to the complexity of the resulting optimization problem, the problem is solved iteratively by performing a number of simulations on the system. Even though it is theoretically guaranteed that falsification methods will eventually find the bugs in the system, in practice, the performance of these methods, i.e., how many tests/simulations are executed before a bug is detected, depends on the specification, on the system and on the optimization method. In this paper, we define and utilize coverage metrics on the state space of hybrid systems in order to improve the performance of the falsification methods.

1. INTRODUCTION

Despite the significant progress in control synthesis for hybrid systems [47], the problem still remains challenging in practice. Among the reasons for the difficulty of the synthesis problem are the large number of continuous state variables, the highly non-linear system dynamics and the interconnected components, many of which are not available in a form amenable to symbolic analysis. Thus, currently, control synthesis methods are utilized for specific operating modes of the system and the different operating modes are combined in an ad-hoc way based on rule tables and engineering experience. However, such a process is error prone as the large number of recalls in safety critical applications indicates.

Therefore, hybrid system verification has been proposed as an alternative. In verification, the user defines a set of bad behaviors of the system and the goal is to prove that no bad behavior is possible. Along these lines, a number of methods have been developed ranging from reachability computation techniques [16, 28] to theorem provers [44]. Despite the successful application of these methods in specific applications, e.g., [34], or on very large linear hybrid systems [28], these methods cannot, in general, be applied in an *automatic* way to problems of industrial size. The reasons are similar - if not identical - to the reasons that limit the applicability of control synthesis methods.

Another approach to verifying properties of hybrid systems relies on testing or simulation guided-methodologies [37]. Such approaches have been found to provide valuable insights and detect errors in problems of industrial size and complexity [35, 26, 31]. One specific class of such methods is referred to as specification robustness guided falsification [5, 42]. As the name implies, such methods search for behaviors that violate a formal specification by executing a number of judiciously chosen tests.

The formal specifications can be stated in Metric Temporal Logic (MTL) [39] if the real-valued signals are abstracted into Boolean-valued signals, or more explicitly in Signal Temporal Logic (STL) [41] if not. The important component of robustness guided falsification methods is that formal specifications do not valuate using Boolean semantics, but quantitative multi-valued semantics [25, 24, 22, 10]. Namely, given a system trajectory (behavior), the robust semantics evaluate to positive values if the trajectory satisfies the specification and to negative values if the trajectory violates the specification. Moreover, the magnitude of the robust evaluation indicates how robustly the behavior satisfies or violates the specification. Thus, a falsification algorithm should try to generate tests that reduce the robustness of the system behavior with respect to the specification and, eventually, become negative. This is essentially the application of a range of stochastic or deterministic optimization algorithms such as Simulated Annealing [5], Cross-entropy [45], Ant Colony Optimization [13], and Nelder-Mead [20] to the falsification problem.

As mentioned earlier, the aforementioned techniques have shown their value in a number of applications. Moreover, theoretically speaking, they are guarantee to eventually detect system errors as long as these are not of measure zero in the search space, e.g., see [3, 7]. In the falsification problem, the search space consists of initial conditions, input signals and other system parameters. However, the algorithm performance, i.e., how many tests are required before a bad

behavior is detected, depends on the landscape induced by the specification robustness over the search space of the hybrid system. In particular, these methods apply more effectively to continuous (non-hybrid) dynamical systems. Even though continuous systems may still have a large number of local minima, their basin of attraction is usually large. The latter fact helps algorithms to locate faster regions of interest in the search space.

However, the same does not hold for hybrid systems. Local minima can now appear anywhere in the search space with very small or non-smooth basins of attraction. Therefore, the probability that these regions will be detected becomes very small. In addition, gradient descent algorithms [4, 9, 2] will not work in such cases. However, if we can extract information about potential discontinuities in the hybrid system, then we can narrow our search to such regions which otherwise would have been rather improbable to sample from.

In this paper, we make several contributions. First, we show how we can instrument Simulink/Stateflow models in order to guide the falsification search toward promising regions. Second, we define coverage metrics not on the continuous state space of the hybrid system, but on its components that introduce discontinuities. In this way, we let the stochastic algorithms search over locally continuous spaces while we implicitly guide the search towards promising operating modes of the hybrid system. This is important because computing coverage for systems with many real-valued state variables becomes an intractable task. Furthermore, our methods may still be utilized in testing actual implementations or testing models which contain black-box components. Finally, we have modified our robustness computation engine TaLiRo [24, 26] to handle this additional information and we have implemented everything within our S-TaLiRo toolbox [1, 14].

2. PROBLEM FORMULATION

In this work, we consider models of hybrid systems or, more generally, of Cyber Physical Systems (CPS) as models developed within a Model Based Development (MBD) language such as Ptolemy [23] or Simulink/Stateflow. We want to test models with respect to requirements (specifications) presented in Metric Temporal Logic (MTL) [39]. MTL is a well know formalism for stating real-time properties.

In this paper, we assume that $\mathbb{R}$ is the set of real numbers, $\mathbb{R}_+$ is the set of non-negative real numbers, and $\overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$. Also, $\mathbb{N}$ is the set of natural numbers including 0 and $\mathbb{Z}$ the integers. Given two sets A and B, B^A is the set of all functions from A to B, i.e., for any $f \in B^A$ we have $f : A \to B$. We define $\mathcal{P}(A)$ to be the power set of the set A. Since we are considering testing or simulation-guided methods, we fix $T \in \mathbb{R}_+$ to be the maximum simulation time.

2.1 System Representation

Formally, we view a system Σ as a mapping from initial conditions $\mathcal{X}_0$, system parameters P and input signals U^R to output signals Y^R. Here, R represents an abstract time domain. For example, $R = \mathbb{N}\times\mathbb{R}$ when we want to talk about solutions to trajectories of hybrid systems [40]. However, here we will just assume that $R = [0, T]$ to avoid many technical issues that do not contribute to the results in this paper. In the following, $[0, T]$ can be thought as the dense physical time domain for the testing or the simulation. Also, U is the set of input values (input space) and Y is the set of output values (output space).

The following three restrictions on the system are critical in order to be algorithmically searchable over an infinite space:

1. The input signals $u \in U^{[0,T]}$ (if any) must be piecewise continuous defined over a finite number of intervals over $[0, T]$. This assumption is necessary in order to be able to parameterize the input signal space over a finite set of parameters. Thus, in the following we assume that any $u \in U^{[0,T]}$ of interest can be represented by a vector of parameter variables p taking values from a set P_U.
2. The output space Y must be equipped with a non-trivial metric. For example, the discrete metric does not provide any useful quantitative information. Concrete examples will be provided later in Section 3.
3. The system Σ must be deterministic[1]. That is, for a specific initial condition x_0 and input signal u, there must exist a unique output signal $\mathbf{y}$.

The previous restrictions render the system Σ to be a function $\Delta_\Sigma : \mathcal{X}_0 \times P \times P_U \to Y^{[0,T]}$ which takes as input an initial condition vector $x_0 \in \mathcal{X}_0$ and two parameter vectors $p \in P$ and $p' \in P_U$, and produces as output a signal $\mathbf{y} : [0, T] \to Y$. Since we consider testing and/or simulation, we assume that there exists a sampling function $\tau : \mathbb{N} \to [0, T]$ that returns for each sample i its time stamp $\tau(i)$. In practice, τ is a partial function $\tau : N \to [0, T]$ with $N \subset \mathbb{N}$ and $|N| < \infty$. A *timed state sequence* or *trace* is the pair $\mu = (\mathbf{y} \circ \tau, \tau)$. We will also denote $\mathbf{y} \circ \tau$ by $\tilde{\mathbf{y}}$. The set of all timed state sequences of Σ that correspond to any sampling function τ will be denoted by $\mathcal{L}(\Sigma)$. That is, $\mathcal{L}(\Sigma) = \{(\mathbf{y} \circ \tau, \tau) \mid \exists\tau \in [0, T]^N \,.\, \exists x_0 \in \mathcal{X}_0 \,.\, \exists p \in P \,.\, \exists p' \in P_U \,.\, \mathbf{y} = \Delta_\Sigma(x_0, p, p')\}$.

2.2 The Falsification Problem

The falsification problem is the process of finding a witness output signal $\mathbf{y}'$ of the system Σ which does not satisfy a requirement represented in MTL. MTL can capture many system requirements by defining a set of atomic propositions AP which labels subsets of the output space Y. We define those subsets through an observation map $\mathcal{O} : AP \to \mathcal{P}(Y)$ where each $\pi \in AP$ is mapped to a set $\mathcal{O}(\pi) \subset Y$. MTL formulas are built over the set of atomic propositions AP, with combinations of the propositional and temporal logic operators. Propositional logic operators are the *conjunction* ($\wedge$), *disjunction* ($\vee$), *negation* ($\neg$), *implication* ($\to$) and *equivalence* ($\leftrightarrow$). Some of the temporal operators, which we will be using here, are *eventually* ($\Diamond_\mathcal{I}$), *always* ($\Box_\mathcal{I}$) and *until* ($\mathcal{U}_\mathcal{I}$). The subscript $\mathcal{I}$ imposes timing constraints on the temporal operators.

Problem 1 (MTL Falsification). *For an MTL specification φ, the MTL falsification problem consists of finding*

[1]We remark that this assumption can also be relaxed [7].

an output signal $\mathbf{y}$ *of the system* Σ *starting from some valid initial state* x_0 *under a parameter vector* p *and an input signal* u *such that* $\mathbf{y}$ *does not satisfy specification* φ.

The challenging problem here is how to guide the search for such a falsifying trajectory. Previously, we utilized the notion of robustness of temporal logic formulas [25, 24] in order to convert the falsification problem into an optimization problem [13, 42, 45]. Briefly, temporal logic robustness provides a metric of how much a trajectory satisfies a specification. Positive robustness implies that the trajectory satisfies the specification and, moreover, that there exists a neighborhood of trajectories (or signals) that also satisfy the specification. Negative robustness implies that the trajectory does not satisfy the specification. Therefore, in order to falsify the specification, we can use the temporal logic robustness as a cost function which we attempt to minimize.

EXAMPLE 1. *Consider the hybrid automaton* Σ_1 *given in Fig. 1. For a review of the syntax and semantics of hybrid automata please refer to [11, 47]. Briefly, if the initial state* $x_0 = (x_{01}, x_{02})$ *is in the set* S *(yellow box in Fig. 2), then* Σ_1 *follows the dynamics in location* l_2*, while if the initial system state* x_0 *is in* $[1,1]^2 \backslash S$ *(green box in Fig. 2), then* Σ_1 *follows the dynamics in location* l_1*. Moreover, if the system is operating under the dynamics in location* l_1 *and the state of the system enters the set* S*, then the system switches to location* l_2*. Sample trajectories with initial conditions over a grid of 0.05 intervals in each dimension over the set of initial conditions* $[-1,1]^2$ *are presented in Fig. 2.*

In this example, the specification is $\varphi(a,b)$ *where* $\varphi(\pi_1,\pi_2) = \square_{[0,2]}\neg\pi_1 \wedge \square_{[0,2]}\neg\pi_2 \equiv \square_{[0,2]}(\neg\pi_1 \wedge \neg\pi_2)$*, and the atomic propositions* a *and* b *map to the following sets*

$$\mathcal{O}(a) = [-1.8, -1.4] \times [-1.6, -1.4]$$

and

$$\mathcal{O}(b) = [3.7, 4.1] \times [-1.6, -1.4].$$

The atomic propositions a *and* b *correspond to the red boxes in Fig. 1. The specification reads always in the time interval [0,2] the system trajectories should not enter either of the two red boxes.*

The specification robustness landscape is presented in Fig. 3. In brief, the falsification problem reduces to finding negative values over the robustness landscape. Some example trajectories that do not satisfy the specification and start from the set $S = [0.85, 0.95]^2$ *are presented in Fig. 2.*

The general overview of the solution of the MTL falsification problem as an optimization problem appears in Fig. 4. The search/optimization algorithm proposes initial conditions, parameters and input signals. Then, the simulator or hardware in the loop system produces the system behavior over which the specification robustness is evaluated. The process is repeated until a maximum number of test/simulations is reached or a falsifying behavior is detected. Based on this framework, we have developed the Matlab toolbox

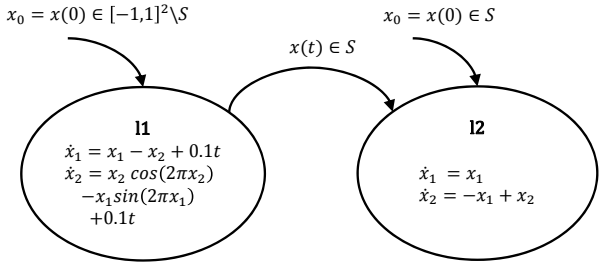

Figure 1: The simple hybrid automaton Σ_1 **of Example 1. In this example,** $S = [0.85, 0.95]^2$ **and** t **is the time variable.**

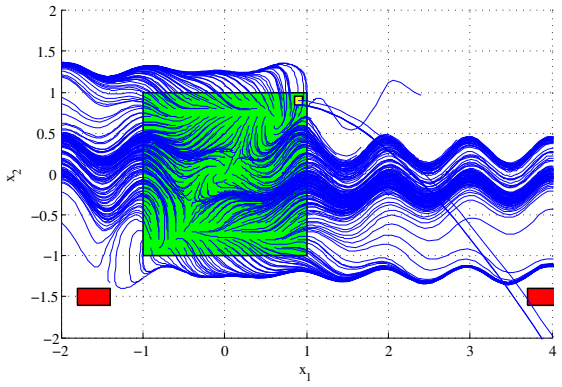

Figure 2: The trajectories of the hybrid automaton from Fig. 1.

S-TALIRO [14]. Given a system and its specification, S-TALIRO searches for a system trajectory that minimizes the robustness value of the specification.

EXAMPLE 2. *[Cont. Example 1] When* S-TALIRO *was run on the system of Fig. 1 using Simulated Annealing for 500 tests for 100 times, then it detected a falsifying behavior only 32 out of 100 times with an average number of 274 tests. The algorithm usually gets trapped in the large region of the local minimizers along* $x_2 = -1$*. Thus, we need to derive methods and heuristics to guide the search towards potentially more promising regions.*

2.3 Coverage Guarantees

The purpose of this work is twofold. First, it aims in achieving faster falsification with respect to the number of tests/simulations. Second, it aims to provide coverage guarantees after the falsification algorithm has concluded. In other words, if a falsification algorithm [13, 42, 45] could not provide a falsifying witness, at least it will be able to list every "mode" of the system that was accessed and, potentially, for how long. The latter is especially important in hybrid systems since falsifying trajectories can very well depend on the sequence of "modes" as well as the time the system remained in each mode. In a sense, part of what we are trying to achieve relates to the branch and condition coverage criteria of software testing [12].

We will formalize the coverage problem for hybrid systems

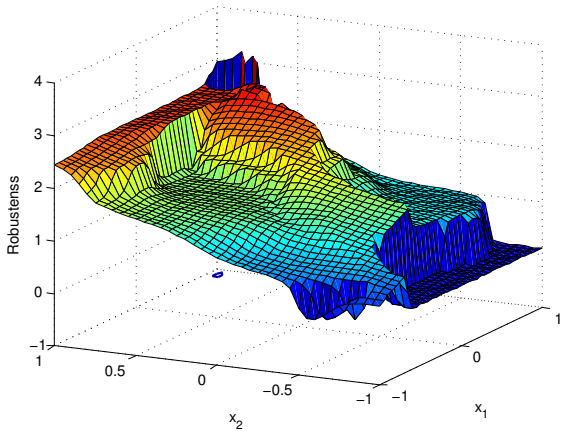

Figure 3: The specification robustness landscape over the set of initial conditions $[-1,1]^2$ for Example 1. The blue level set at robustness level -1 represents the initial conditions generating trajectories that falsify the specification.

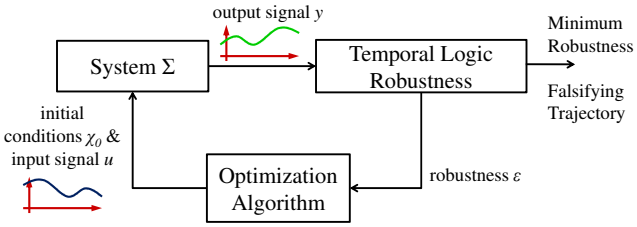

Figure 4: Overview of the solution to the MTL falsification problem posed as an optimization problem.

by still maintaining the input/output system definition of Section 2.1. We assume that the output space Y of the system Σ comprises of the original output space Y_Σ of the system (equipped with the nontrivial metric), an auxiliary output space Y_A (which for now we leave undefined) and a finite space Y_F. That is, $Y = Y_\Sigma \times Y_A \times Y_F$.

The output space Y_F plays an important role since it captures the state of important components in the system. For example, it can capture the state of switch blocks in state diagrams or the state of statechart machines. Thus, the coverage metrics can be defined as simply set coverage metrics over the space Y_F or more generally as coverage of sequences of symbols from Y_F^*. This paper focuses on the former coverage criterion.

Problem 2 (Coverage Guarantees). *Given a system Σ with output space $Y = Y_\Sigma \times Y_A \times Y_F$ and an MTL specification φ, solve the MTL falsification problem while at the same time maximizing the coverage of the set Y_F.*

Formally, if a falsification algorithm produces a finite set of tests $\mathcal{T} = \{(\tilde{\mathbf{y}}_i, \tau_i)\}_i$, then we are trying to maximize $|\cup_{(\tilde{\mathbf{y}},\tau)\in\mathcal{T}} \cup \mathbf{pr}_{Y_F}(\tilde{\mathbf{y}})|$. Informally speaking, $\mathbf{pr}_{Y_F}$ is the projection of sequence of tuples from Y on Y_F.

Our main approach to solving the problem will utilize the hybrid distance metrics that were introduced in [42, 5].

3. MTL AND ROBUST SEMANTICS

In this section, briefly review MTL and its robust semantics. We also present some extensions necessary for this work.

Definition 1 (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 $\varphi ::= \top \mid \pi \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \mathcal{U}_\mathcal{I} \varphi$, where $\pi \in AP$ and $\top$ is* true.

Using a *metric* $\mathbf{d}$ [46], we can define a distance function that captures how far away a point $y \in Y$ is from a set $S \subseteq Y$. Intuitively, the distance function assigns positive values when y is in the set S and negative values when y is outside the set S. The metric $\mathbf{d}$ must be at least a generalized quasi-metric as described in [5] which also includes the case where $\mathbf{d}$ is a metric as it was introduced in [25].

Definition 3.1 (Signed Distance). *Let $y \in Y$ be a point, $S \subseteq Y$ be a set and $\mathbf{d}$ be a metric. Then, we define the Signed Distance from y to S to be*

$$\mathbf{Dist}_\mathbf{d}(y,S) := \begin{cases} -\mathbf{dist}_\mathbf{d}(y,S) & \text{if } y \notin S \\ \mathbf{dist}_\mathbf{d}(y, Y \backslash S)\} & \text{if } y \in S \end{cases}$$

where

$$\mathbf{dist}_\mathbf{d}(y,S) := \inf\{\mathbf{d}(y,y') \mid y' \in S\}$$

and inf *is the infimum.*

We should point out that we use the extended definition of supremum ($\sqcup$) and infimum ($\sqcap$). 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, $\sup\emptyset := -\infty$ and $\inf\emptyset := +\infty$.

S-TaLiRo [14] originally supported the following metrics. When $Y = \mathbb{R}^n$, then we use the Euclidean metric $d(y_1,y_2) = \|y_1 - y_2\|$. When Y is a hybrid space, e.g., $Y = \mathbb{R}^n \times Q$ with $Y_\Sigma = \mathbb{R}^n$ and $Y_F = Q$, where Q is the set of states of a single statechart in the model, then we use the following generalized quasi-metric [42, 5] $\mathbf{d_h} : Y \times Y \to \langle \mathbb{N} \cup \infty, \overline{\mathbb{R}}_+ \rangle$ with definition:

$$\mathbf{d_h}(\langle x,q\rangle, \langle x',q'\rangle) = \begin{cases} \langle 0, d(x,x')\rangle & \text{if } q = q' \\ \left\langle \pi(q,q'), \min\limits_{q \to q''} {}^{\pi(q,q')-1}_{\sim\!\!>}{}_{q'} \mathbf{dist}_d(x, \mathbf{G}^t(q,q'')) \right\rangle & \text{otherwise} \end{cases}$$

where π is the shortest path metric on a graph, d is a metric on $\mathbb{R}^n$, and $\mathbf{G}^t$ denotes that the guard set that activates the transition from state q to state q' in the statechart. We assume that $\mathbf{G}^t(q,q'') \subseteq Y_\Sigma$. Note that we use the superscript t to indicate that the guard may be changing with respect to time. Finally, the min operator quantifies over all neighboring states q'' of q that are on a shortest path from q to q'. A more detailed discussion can be found in [5].

Therefore, when the two points $\langle x, q\rangle, \langle x', q'\rangle$ are in the same statechart state, then the distance computation reduces to the distance computation between the points in the continuous state space. When the two points $\langle x, q\rangle, \langle x', q'\rangle$ are in different statechart states, then the distance is the path distance between the two statechart states "weighted" by the distance to the closest guard that will enable the transition to the next statechart state that reduces the path distance. Essentially, the last condition is a heuristic that gives preference to shortest paths.

DEFINITION 3.2 (ROBUST SEMANTICS). *Consider an extended generalized quasi metric space $(Y, \mathbf{d})$. Let $\mu \in \mathcal{L}(\Sigma)$ and $\mathcal{O} : AP \to \mathcal{P}(Y)$, then the robust semantics of any formula $\varphi \in MTL$ with respect to μ at sample time $i \in N$ is recursively defined as:*

$$\begin{aligned}
[\![\top, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) &:= \bigsqcup Range(\mathbf{d}) \\
[\![\pi, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) &:= \mathbf{Dist}_{\mathbf{d}}(\tilde{\mathbf{y}}(i), \mathcal{O}(\pi)) \\
[\![\neg\varphi_1, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) &:= -[\![\varphi_1, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) \\
[\![\varphi_1 \vee \varphi_2, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) &:= [\![\varphi_1, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) \sqcup [\![\varphi_2, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) \\
[\![\varphi_1 \mathcal{U}_{\mathcal{I}} \varphi_2, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i) &:= \bigsqcup_{i' \in \tau^{-1}(\tau(i) +_R \mathcal{I})} ([\![\varphi_2, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i') \sqcap \\
&\quad \sqcap_{i \le i'' < i'} [\![\varphi_1, \mathcal{O}]\!]_{\mathbf{d}}(\mu, i'')
\end{aligned}$$

where $t +_R \mathcal{I} = \{t'' \in R \mid \exists t' \in \mathcal{I} \,.\, t'' = t + t'\}$, τ^{-1} is the inverse function of τ, and $-$ is an unary operator defining the "negative" values of the range of $\mathbf{d}$, i.e., $Range(\mathbf{d})$.

The semantics of the other operators can be defined using the above basic operators. For example, $\Diamond_{\mathcal{I}}\phi \equiv \mathbf{T}\mathcal{U}_{\mathcal{I}}\phi$ and $\Box_{\mathcal{I}}\phi \equiv \neg\Diamond_{\mathcal{I}}\neg\phi$.

EXAMPLE 3. *[Cont. Example 1] If we consider now the specification $\varphi' = \varphi(a', b')$ with $\mathcal{O}(a') = \mathcal{O}(a) \times \{l_1, l_2\}$ and $\mathcal{O}(b') = \mathcal{O}(b) \times \{l_1, l_2\}$, then the robustness landscape is identical to the one in Fig. 3. On the other hand, if we change in $\varphi'' = \varphi(a'', b'')$ the mapping of the atomic propositions to $\mathcal{O}(a'') = \mathcal{O}(a) \times \{l_2\}$ and $\mathcal{O}(b'') = \mathcal{O}(b) \times \{l_2\}$ and we map the resulting hybrid robustness values to the real line (see `map2line.m` function in* S-TALIRO*), then we get the robustness landscape of Fig. 5. It is clear that in this case the search is biased towards moving to mode l_2 of the simple hybrid automaton. Indeed, when* S-TALIRO *is run on the system of Fig. 1 with the new atomic propositions using Simulated Annealing for up to 500 tests 100 times, then it always finds a falsifying behavior within 62 tests on average. A successfully falsifying execution of* S-TALIRO *is presented in Fig. 6.*

As Example 3 hints, our goal is to use coverage metrics to guide the search towards less visited states or logical conditions.

When dealing with industrial size models it is unrealistic to assume that there is going to be a single statechart or that it is going to be efficient to flatten all the different statecharts into one. Thus, for the purposes of this work, we have extended S-TALIRO to handle multiple finite state

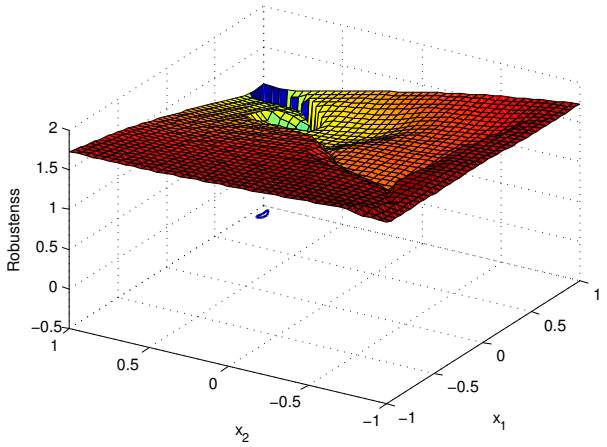

Figure 5: The resulting robustness landscape for specification φ'' in Example 3.

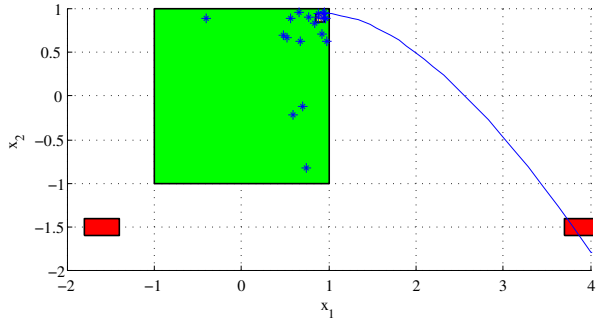

Figure 6: Example 3: The samples selected during the falsification process and the falsifying witness.

components. We extend the metric in a natural way using the maximum pairwise hybrid distance. Namely, when $Y = \mathbb{R}^n \times Q_1 \times \ldots \times Q_m$, where $Q_1, \ldots, Q_m$ are the sets of states of m statecharts, we use the following metric:

$$\begin{aligned}
&\mathbf{d}_{\mathbf{h}}^{\max}(\langle x, q_1, \ldots, q_m\rangle, \langle x', q_1', \ldots, q_m'\rangle) = \\
&\max\{\mathbf{d}_{\mathbf{h}}(\langle x, q_1\rangle, \langle x', q_1'\rangle), \ldots, \mathbf{d}_{\mathbf{h}}(\langle x, q_m\rangle, \langle x', q_m'\rangle).\}
\end{aligned}$$

We remark that later in the text we will present how we instrument the model under test to expose additional real-valued signals to the testing algorithm. These signals will become part of the auxiliary output space Y_A. For the purposes of computing the distance metric, the output spaces Y_Σ and Y_A will be treated as a Euclidean metric space. That is, in $\mathbf{d}_{\mathbf{h}}^{\max}$ we assume that $Y_\Sigma \times Y_A = \mathbb{R}^n$. Therefore, $Y_F = Q_1 \times \ldots \times Q_m$.

4. COVERAGE GUIDED FALSIFICATION

In this section, we present the basic steps behind our coverage guided specification falsification framework. In the following, we will refer to the finite system states as *modes* to differentiate them from the continuous infinite state variables of the system.

In case no mode information or multiple modes are dictated for some atomic propositions, the falsification algorithm starts with the default search algorithm without any coverage guidance. As the search process evolves, coverage statistics are collected for the output space Y_F. Some of the proposed coverage metrics are:

1. Number of unique occurrences[2] of each mode $q_i \in Q_i$ for each statechart Q_i.
2. Percentage of test time (samples) that the system remained in each mode $q_i \in Q_i$ for each statechart Q_i.
3. Number of unique occurrences of each combination of modes $\langle q_1, \ldots, q_m \rangle$ of the statecharts.
4. Percentage of test time (samples) that the system remained in each combination of modes $\langle q_1, \ldots, q_m \rangle$ of the statecharts.

Since these are all heuristic metrics, more options are possible especially when considering sequences of modes. Another point which will need to be experimentally studied in the future is whether to collect such statistics over all test cases or the test cases with smallest robustness value found.

If the falsification process terminates successfully, then the coverage information can also be returned to the user. If the falsification process fails, then the search algorithm switches to the coverage guided search. In coverage guided search, preference is first given to combinations of modes with the least coverage values.

Due to the complexity of the systems that we consider, in general, it is not possible to know whether a specific combination of modes is reachable [30]. Clearly, it is also not possible, in general, to compute initial conditions and parameters that would make a specific combination of modes reachable. Therefore, we modify our specification to bias the search towards the desired combination of modes while still trying to falsify the original specification. In particular, if the original specification is φ and the least visited (or not visited at all) set of combinations of modes is $Q_{Des} \subseteq Q_1 \times \ldots \times Q_m$, then we define a new atomic proposition p_{Des} with $\mathcal{O}(p_{Des}) = \mathbb{R}^n \times Q_{Des}$ and we create a new MTL formula

$$\varphi' = \varphi \vee \neg\Diamond(p_{Des})$$

The formula φ' now becomes the target for the falsification process. Hence, now, the falsification algorithm while it tries to minimize the robustness, it indirectly pushes the system to go to the modes provided by the atomic proposition p_{Des} since its robustness will be minimized as well.

The basic architecture of the coverage guided falsification algorithm is presented in Fig. 7.

EXAMPLE 4. *[Cont. Example 1] We remark that the specification $\varphi(a,b) \vee \neg\Diamond(p_{Des})$ with $\mathcal{O}(p_{Des}) = \mathbb{R}^2 \times \{l_2\}$ induces the same robustness landscape to the one in Fig. 5.*

[2]I.e., we ignore consecutive repetitions of the same mode.

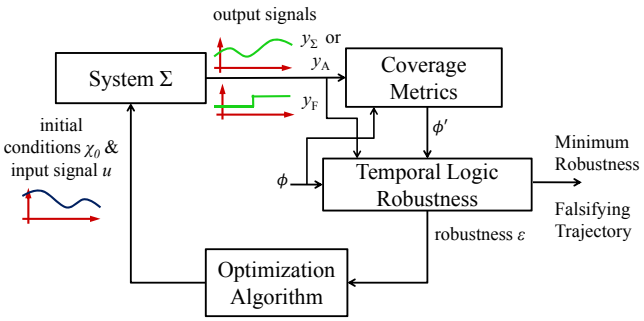

Figure 7: Overview of the basic components of the coverage guided falsification framework.

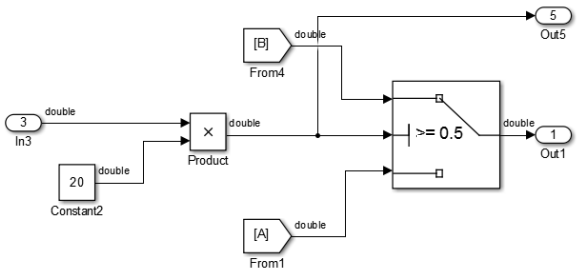

Figure 8: Switch block case 1.

5. IMPLEMENTATION

Next, we discuss details regarding our S-TALIRO implementation of the coverage guided falsification for Simulink/Stateflow models. An important component of the process is the instrumentation of the models. This is important in order to handle automatically industrial size models.

5.1 Model Instrumentation

In the model instrumentation, a Simulink model is analyzed and information is collected about blocks that introduce nonlinearities in a model. For instance, targets for instrumentation are switch blocks, absolute value blocks, saturation blocks, lookup tables, etc. Here, we will only present the process for switch blocks through some examples since all other blocks are treated in similar ways.

In Fig. 8 and 9, we present 2 cases of interest for switch blocks. In both cases, the switch block chooses either signal A or signal B based on the switching condition provided in the middle port. Each such switch can be modeled using a two state statechart with transition guards the switching conditions of the blocks (see Fig. 10).

Our instrumentation algorithm analyzes locally the switch blocks and introduces output ports that become elements in the output spaces Y_A and Y_F. In particular, consider the switch in Fig. 8. In this case, `Out1` is part of the output space Y_Σ. Through the instrumentation, we introduce the output port `Out5` to the set Y_A. Moreover, in an outer harness for the model, we compute whether the statechart is in state A or B and this value becomes an element in the output set Y_F. The information that corresponds to the left statechart

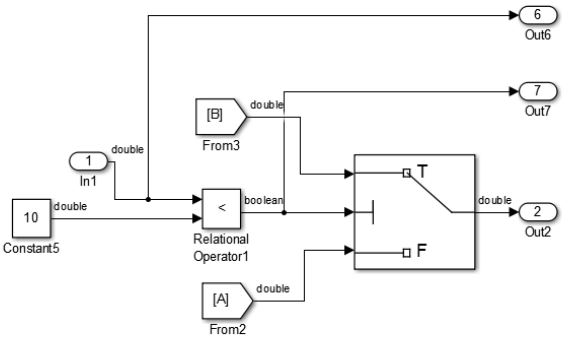

Figure 9: Switch block case 2.

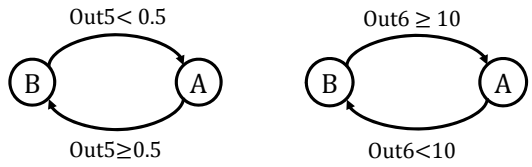

Figure 10: Left: Statechart corresponding to switch in Fig.8. Right: Statechart corresponding to switch in Fig.9.

in Fig. 10 is also automatically extracted and utilized in the computation of the distance metric $\mathbf{d}_{\mathbf{h}}^{\max}$.

The instrumentation for the case of the switch in Fig. 9 is similar. In this case, `Out2` is part of the output space Y_Σ. The instrumentation introduces `Out6` and `Out7` as elements of the sets Y_A and Y_F, respectively. The output port `Out7` captures the state of the switch, while the value in the output port `Out6` along with the right statechart in Fig. 10 are needed for computing the distance metric $\mathbf{d}_{\mathbf{h}}^{\max}$.

In the previous examples, any computed coverage information directly corresponds to state coverage for the switch block. In other words, condition coverage is identical to mode coverage. However, in many cases, a Boolean input signal to the middle port of a switch block may be the output of a blackbox or whitebox Boolean circuit. In the former case, we have to resort to condition coverage in the relational operator blocks as in Fig. 9. In the latter case, we would have to solve a satisfiability problem in order to directly control the state of the switch block.

Currently, the Simulink/Stateflow model instrumentation is performed automatically by S-TaLiRo for a few commonly occurring blocks. Our instrumentation function recursively instruments any subsystems unless they are masked linked systems. However, the users can manually instrument any block even if such work can be tedious.

5.2 S-TaLiRo Details

Figure 11 provides an overview of the modular architecture of S-TaLiRo along with the data flow among the different components. The user provides the model, the specification and whether she would like to execute coverage guided falsification. The internal components are transparent to the user unless the user would like to manually instrument the model or parts of the model.

Among the current options provided to the user for coverage guided falsification are:

1. The number of MTL formula updates.
2. The number of tests/simulations before the MTL formula is updated.
3. The coverage metric utilized.
4. Whether all the tests/simulations are utilized in the computation of the coverage metrics.
5. Whether in a formula update the next mode combination is chosen randomly or in some other order.

S-TaLiRo can be downloaded from the website [1] along with the examples in this paper.

6. RELATED WORK

Recently, the research direction of simulation guided approaches for verification has gained a lot of traction. Among the reasons for the recent research focus is that simulation guided methods bridge the gap between exhaustive formal verification and ad hoc testing: they are applicable to industrial size problems while providing probabilistic guarantees of completeness.

A survey of recent results on simulation based verification methods has appeared in [37] while older results are surveyed in [4, 48]. For the sake of completeness, here we will briefly mention some methods which are not directly related to specification robustness guided falsification.

Three major research directions in simulation based verification are (i) *robust testing* [36, 27, 21, 32, 33, 19], (ii) *randomized tree exploration testing* [15, 43, 18] and (iii) the combination of the two aforementioned approaches [29, 17]. In robust testing, the underlying principle is that each simulation actually corresponds to a neighborhood of simulations. Therefore, it becomes possible to exhaustively cover the search space of the hybrid system with only a finite number of simulations and, thus, prove system correctness up to a finite time horizon. One drawback of robust testing is that it only applies to certain types of systems for which bounds on the divergence of nearby system trajectories can be computed. The aforementioned methods [36, 27, 21, 32, 33, 19] essentially differ on the theory utilized for computing the divergence bounds.

On the other hand, tree exploration methods [15, 43, 18] try to grow a tree in the state or output space of the system. Among the advantages of these methods are: (1) they can offer coverage guarantees on both the continuous and the discrete state space of a hybrid system; (2) they are very efficient on systems with complex nonlinear continuous dynamics; and (3) they can handle systems with blackbox components. On the other hand, it is not easy to use such methods to falsify real-time specifications and they cannot be effectively used in hardware-in-the-loop testing.

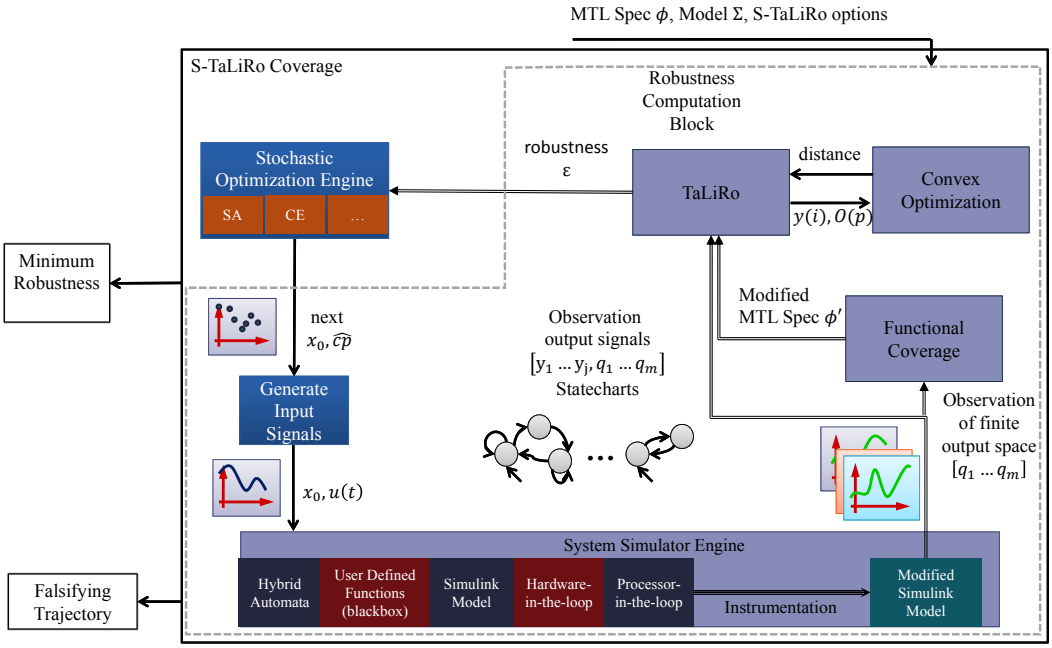

Figure 11: S-TaLiRo architecture for coverage guided falsification.

It is important to note that the aforementioned techniques have found applications beyond the falsification problem as defined in Section 2.2. Interesting applications range from the conformance testing problem [6, 8] to the specification mining problem [35, 49, 38]. Thus, the results developed in this paper can also be applied to these problems as well.

A more recent simulation based verification framework is referred to as *trajectory splicing*. Trajectory splicing was initially proposed as a complementary falsification technique to traditional overapproximate verification techniques which usually provide an abstract counter example if they fail. Trajectory splicing can be used produce a concrete violation when it exists. The first version [51] used off the shelf optimization techniques and the later versions switched to a graph based simulation framework capable of handling black box dynamics [50]. Symbolic execution of controller code was subsequently incorporated [52].

The idea behind trajectory splicing is to iteratively search in the space of trajectory segments. However, such disconnected segments do not form concrete violations due to the gaps that exist between the ending state of one segment and the starting state of the subsequent segment. Therefore, a local search is used to minimize the gap between these segments, effectively splicing them together to form a concrete trajectory.

7. CONCLUSIONS

We have presented our recent work in improving property falsification methods for hybrid systems by incorporating coverage metrics. Coverage metrics can assist falsification methods by biasing/guiding the search towards regions of the state space that have not been explored sufficiently before. This is an important improvement since many hybrid systems have discontinuities which renter certain behaviors rare events. Our new work utilizes structural information about these discontinuities in order to sample more frequently around regions that correspond to rare system behaviors. Our current work has been implemented in our toolbox S-TaLiRo [14, 1].

This research direction is still in its infancy. Many coverage metrics that have been developed for software may be adapted to testing hybrid systems. Especially, we would like to focus on coverage metrics for achieving path coverage in hybrid systems. Another practical direction is to experiment with these testing methods on hardware-in-the-loop systems. Finally, our current research topic is the integration of trajectory splicing methods with the falsification results in this paper.

Acknowledgments

The authors would like to thank the Model Based Development group at the Toyota Technical Center for the many fruitful discussions on this topic and their continuing support. This research was funded in part by NSF awards CNS-1319560, CNS-1319457, CNS-1350420, IIP-1361926 and the NSF I/UCRC Center for Embedded Systems.

8. REFERENCES

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

[2] H. Abbas and G. Fainekos. Linear hybrid system falsification through local search. In *Automated Technology for Verification and Analysis*, volume 6996 of *LNCS*, pages 503–510. Springer, 2011.

[3] H. Abbas and G. Fainekos. Convergence proofs for simulated annealing falsification of safety properties. In *Proc. of 50th Annual Allerton Conference on*

Communication, Control, and Computing. IEEE Press, 2012.

[4] H. Abbas and G. Fainekos. Computing descent direction of mtl robustness for non-linear systems. In *American Control Conference*, 2013. [Under review].

[5] H. Abbas, G. E. Fainekos, S. Sankaranarayanan, F. Ivancic, and A. Gupta. Probabilistic temporal logic falsification of cyber-physical systems. *ACM Transactions on Embedded Computing Systems*, 12(s2), May 2013.

[6] H. Abbas, B. Hoxha, G. Fainekos, J. V. Deshmukh, J. Kapinski, and K. Ueda. Conformance testing as falsification for cyber-physical systems. Technical Report 1401.5200, arXiv, 2014.

[7] H. Abbas, B. Hoxha, G. Fainekos, and K. Ueda. Robustness-guided temporal logic testing and verification for stochastic cyber-physical systems. In *Proc. of IEEE International Conference on CYBER Technology in Automation, Control, and Intelligent Systems*, 2014.

[8] H. Abbas, H. Mittelmann, and G. Fainekos. Formal property verification in a conformance testing framework. In *12th ACM-IEEE International Conference on Formal Methods and Models for System Design*, 2014.

[9] H. Abbas, A. Winn, G. Fainekos, and A. A. Julius. Functional gradient descent method for metric temporal logic specifications. In *American Control Conference*, 2014. [Under review].

[10] T. Akazaki and I. Hasuo. Time robustness in mtl and expressivity in hybrid system falsification. In *Computer Aided Verification*, volume 9207 of *LNCS*, pages 356–374. Springer, 2015.

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

[12] P. Ammann and J. Offutt. *Introduction to Software Testing*. Cambridge University Press, 2008.

[13] Y. S. R. Annapureddy and G. E. Fainekos. Ant colonies for temporal logic falsification of hybrid systems. In *Proceedings of the 36th Annual Conference of IEEE Industrial Electronics*, pages 91–96, 2010.

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

[15] M. Branicky, M. Curtiss, J. Levine, and S. Morgan. Sampling-based planning, control and verification of hybrid systems. *IEE Proc.-Control Theory Appl.*, 153(5):575–590, 2006.

[16] X. Chen, E. Abraham, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Computer-Aided Verification*, 2013.

[17] T. Dang, A. Donze, O. Maler, and N. Shalev. Sensitive state-space exploration. In *Proc. of the 47th IEEE Conference on Decision and Control*, pages 4049–4054, Dec. 2008.

[18] T. Dang and N. Shalev. State estimation and property-guided exploration for hybrid systems testing. In *International Conference Testing Software and Systems*, volume 7641 of *LNCS*, pages 152–167. Springer, 2012.

[19] Y. Deng, A. Rajhans, and A. A. Julius. Strong: A trajectory-based verification toolbox for hybrid systems. In *Quantitative Evaluation of Systems*, volume 8054 of *LNCS*, pages 165–168. Springer, 2013.

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

[21] A. Donze and O. Maler. Systematic simulation using sensitivity analysis. In *Hybrid Systems: Computation and Control*, volume 4416 of *LNCS*, pages 174–189. Springer, 2007.

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

[23] J. Eker, J. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Sachs, and Y. Xiong. Taming heterogeneity - the ptolemy approach. *Proceedings of the IEEE*, 91(1):127–144, Jan. 2003.

[24] G. Fainekos and G. J. Pappas. Robustness of temporal logic specifications. In *Formal Approaches to Testing and Runtime Verification*, volume 4262 of *LNCS*, pages 178–192. Springer, 2006.

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

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

[27] G. E. Fainekos, A. Girard, and G. J. Pappas. Temporal logic verification using simulation. In E. Asarin and P. Bouyer, editors, *Proceedings of the 4th International Conference on Formal Modelling and Analysis of Timed Systems*, volume 4202 of *LNCS*, pages 171–186. Springer, 2006.

[28] G. Frehse, C. L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *Proceedings of the 23d CAV*, 2011.

[29] A. Girard and G. J. Pappas. Verification using simulation. In *Hybrid Systems: Computation and Control (HSCC)*, volume 3927 of *LNCS*, pages 272 – 286. Springer, 2006.

[30] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *J. Comput. Syst. Sci.*, 57(1):94–124, 1998.

[31] B. Hoxha, H. Abbas, and G. Fainekos. Using s-taliro on industrial size automotive models. In *Proc. of Applied Verification for Continuous and Hybrid Systems*, 2014.

[32] Z. Huang and S. Mitra. Computing bounded reach sets from sampled simulation traces. In *The 15th International Conference on Hybrid Systems: Computation and Control (HSCC 2012), Beijing, China.*, 2012.

[33] Z. Huang and S. Mitra. Proofs from simulations and modular annotations. In *Proceedings of the 17th International Conference on Hybrid Systems:*

Computation and Control, pages 183–192, 2014.

[34] J.-B. Jeannin, K. Ghorbal, Y. Kouskoulas, R. Gardner, A. Schmidt, and E. Z. A. Platzer. A formally verified hybrid system for the next-generation airborne collision avoidance system. In *TACAS*, volume 9035 of *LNCS*, pages 21–36. Springer, 2015.

[35] X. Jin, A. Donze, J. Deshmukh, and S. Seshia. Mining requirements from closed-loop control models. In *Hybrid Systems: Computation and Control*. ACM Press, 2013.

[36] A. A. Julius, G. E. Fainekos, M. Anand, I. Lee, and G. J. Pappas. Robust test generation and coverage for hybrid systems. In *Hybrid Systems: Computation and Control*, volume 4416 of *LNCS*, pages 329–342. Springer, 2007.

[37] J. Kapinski, J. Deshmukh, X. Jin, H. Ito, and K. R. Butts. Simulation-guided approaches for verification of automotive powertrain control systems. In *American Control Conference*, 2015.

[38] Z. Kong, A. Jones, A. M. Ayala, E. A. Gol, and C. Belta. Temporal logic inference for classification and prediction from data. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, 2014.

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

[40] J. Lygeros, K. H. Johansson, S. N. Simic, J. Zhang, and S. Sastry. Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control*, 48:2–17, 2003.

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

[42] T. Nghiem, S. Sankaranarayanan, G. E. Fainekos, F. Ivancic, A. Gupta, and G. J. Pappas. 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*, pages 211–220. ACM Press, 2010.

[43] E. Plaku, L. E. Kavraki, and M. Y. Vardi. Falsification of ltl safety properties in hybrid systems. In *Proc. of the Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 5505 of *LNCS*, pages 368 – 382. Springer, 2009.

[44] A. Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, Heidelberg, 2010.

[45] S. Sankaranarayanan and G. Fainekos. Falsification of temporal properties of hybrid systems using the cross-entropy method. In *ACM International Conference on Hybrid Systems: Computation and Control*, 2012.

[46] A. K. Seda and P. Hitzler. Generalized distance functions in the theory of computation. *The Computer Journal*, 53(4):bxm108443–464, 2008.

[47] P. Tabuada. *Verification and Control of Hybrid Systems: A Symbolic Approach*. Springer, 2009.

[48] S. Tripakis and T. Dang. *Model-Based Design for Embedded Systems*, chapter Modeling, Verification and Testing using Timed and Hybrid Automata, pages 383–436. CRC Press, 2009.

[49] H. Yang, B. Hoxha, and G. Fainekos. Querying parametric temporal logic properties on embedded systems. In *Int. Conference on Testing Software and Systems*, volume 7641, pages 136–151. Springer, 2012.

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

[51] A. Zutshi, S. Sankaranarayanan, J. V. Deshmukh, and J. Kapinski. A trajectory splicing approach to concretizing counterexamples for hybrid systems. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 3918–3925. IEEE, 2013.

[52] A. Zutshi, S. Sankaranarayanan, J. V. Deshmukh, J. Kapinski, and X. Jin. Falsification of safety properties for closed loop control systems. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 299–300. ACM, 2015.