A Trajectory Splicing Approach to Concretizing Counterexamples for Hybrid Systems

Aditya Zutshi Sriram Sankaranarayanan Jyotirmoy V. Deshmukh James Kapinski

***Abstract*— This paper examines techniques for finding falsifying trajectories of hybrid systems using an approach that we call *trajectory splicing*. Many formal verification techniques for hybrid systems, including flowpipe construction, can identify plausible abstract counterexamples for property violations. However, there is often a gap between the reported abstract counterexamples and the concrete system trajectories. Our approach starts with a candidate sequence of disconnected trajectory segments, each segment lying inside a discrete mode. 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, trajectory splicing uses local optimization to minimize the gap between these segments, effectively splicing them together to form a concrete trajectory.**

We demonstrate the use of our approach for falsifying safety properties of hybrid systems using standard optimization techniques. As such, our approach is not restricted to linear systems. We compare our approach with other falsification approaches including uniform random sampling and a robustness guided falsification approach used in the tool S-Taliro. Our preliminary evaluation clearly shows the potential of our approach to search for candidate trajectory segments and use them to find concrete property violations.

I. Introduction

Hybrid automata serve as computational models for dynamical systems that combine the continuous evolution of state variables interspersed with discrete mode transitions. Such systems naturally arise by composing discrete-time control systems and continuous-time physical plants. This paper address the problem of falsifying safety properties of hybrid systems. A *safety* property specifies that the continuous state of a hybrid automaton starting from some initial region does not reach a pre-specified unsafe region. The verification problem asks whether the hybrid system satisfies a given safety property.

Whereas the term "verification" (from the Latin root *verus* for truth) often indicates a pre-supposition that the property in question is most likely satisfied by the system under consideration, we use its opposite "falsification" to presuppose that the property in question is not satisfied by the system. Therefore, falsification seeks to find a counterexample trajectory to demonstrate the violation of a property. Unfortunately, checking if a dynamical system (discrete, continuous or hybrid) satisfies a safety property is undecidable for all but the simplest of cases.

S. Sankaranarayanan is affiliated with faculty of Computer Science, University of Colorado, Boulder, CO srirams@colorado.edu

A. Zutshi is affiliated with department of Electrical, Computer and Energy Engineering (ECEE) Department, University of Colorado, Boulder, CO aditya.zutshi@colorado.edu

J. V. Deshmukh and J. Kapinski are affiliated with the Toyota Technical Center, CA {jyotirmoy.deshmukh, jim.kapinski}@tema.toyota.com

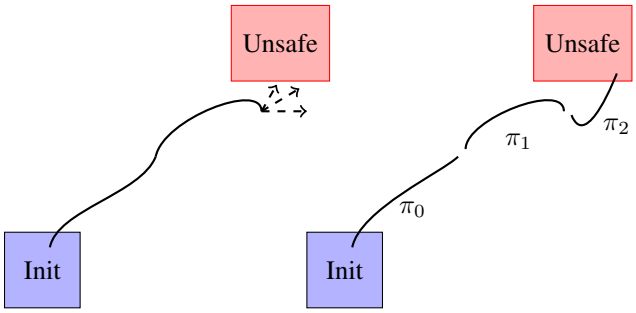

Fig. 1. Illustration of a forward search approach (left), contrasted with our approach using trajectory splicing (right).

A number of verification tools for hybrid automata conservatively estimate the set of reachable behaviors (reach set estimation) for establishing safety properties of hybrid systems [4], [5], [12], [24], [30], [34], [36]. If the unsafe region is reachable, these techniques report a plausible abstract error trajectory. However, due to the over-approximations performed at each step of the reach set estimation, the abstract counterexamples obtained may be spurious. In practice, it is important to provide concrete error trajectories that can be examined by a control engineer. Therefore, in this paper, we focus on the problem of identifying concrete trajectories from a given sequence of modes and transitions. Our approach performs a best-effort search for a concrete trajectory using optimization.

Fig. 1 presents an illustration of our approach. The input to our approach consists of a hybrid automaton along with a sequence of discrete modes and transitions $m_0 \xrightarrow{\tau_1} m_1 \ldots \xrightarrow{\tau_N} m_N$, the first of which ($m_0$) contains the initial region (set of initial conditions), and the last of which (m_N) contains the unsafe region. Our approach attempts to find an actual system trajectory observing this discrete mode sequence. This is achieved by viewing the gaps between the trajectory segments as a cost function and attempting to find segments that minimize this function. As a result, standard numerical optimization techniques are used to iteratively narrow these gaps using gradient and Hessian information from a sensitivity analysis of the system trajectories in each mode. If the optimal cost (sum of gaps between segments) is zero, our approach yields a concrete trajectory of the system exhibiting a violation. Often, in practice, a very small gap (10^{-6} or less) suffices to observe a falsification.

As opposed to other, related techniques, we consider a

search over the space of disconnected trajectory segments. Searching over multiple trajectory segments is advantageous for hybrid trajectories: Our approach exploits continuous sensitivity to initial conditions for each mode. This enables the application of gradient descent techniques. Likewise, the use of trajectory segments separated by different discrete modes allows us to naturally handle the discontinuities that arise through discrete transitions. Secondly, our approach also separates the search for an abstract violation that can be carried out using tools like SpaceEx [24] or Flow* [12] from the problem of concretizing these violations.

Contributions: We setup a general optimization framework for performing concrete counterexample search through trajectory splicing. For hybrid systems the optimization is shown to be non-linear and non-convex, in general. Next, we present a natural approach of partitioning the problem into independent sub-problems. For a given system, this fixes the size of each sub-problem, irrespective of the length of the counterexample.

We have implemented the ideas presented in this paper in a prototype tool in MATLAB$^{\text{TM}}$. We present the results of our approach over a set of benchmarks of varying difficulty, and compare against other approaches including uniform random testing and the tool S-Taliro that uses robustness-guided optimization to search for concrete counterexamples. We find that our approach can find falsifications of properties using a small fraction of the time required to perform uniform random simulations or robustness-guided optimization. In cases where our approach successfully finds concrete violations, it is able to narrow the gaps between trajectory segments to a small value that can be compared to the integration errors that occur during numerical simulations.

A. Related Work

Our approach in this paper is closely related to ideas that arise from hybrid systems verification, optimal control theory (direct multiple shooting), robotic motion planning (probabilistic roadmaps and RRT).

1) Hybrid System Falsification Techniques: Often, hybrid systems fail to conform to their correctness specifications in many unforeseen ways. As a result, the problem of automatically finding falsifications for hybrid systems is of great importance. Approaches to falsification can be broadly categorized into (a) the use of constraint solvers to find concrete violations, (b) adapting motion planning approaches such as RRTs (reviewed elsewhere in this section), and (c) using optimization to systematically search for falsifications.

The use of constraint solvers is a natural and promising approach to falsification. Research on constraint solvers has produced efficient Satisfiability Modulo Theory (SMT) solvers for disjunctive linear constraints such as Z3 and solvers for non-linear constraints such as iSAT and dREAL [16], [23], [27]. As such, the problem of finding falsifications for hybrid systems can be reduced to a constraint solving problem using the *Bounded Model Checking* (BMC) approach [13]. However, doing so requires a discretization of the continuous dynamics assuming a fixed time

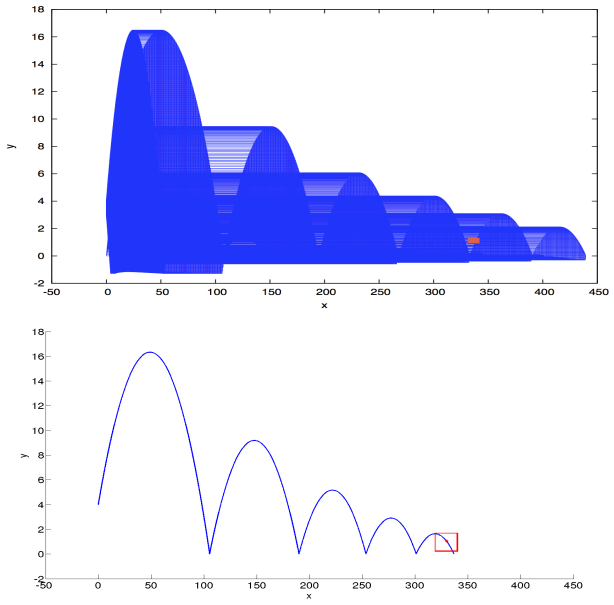

Fig. 2. **(Top)** Flowpipe constructed by the Flow* tool, **(Bottom)** Violation found by our approach.

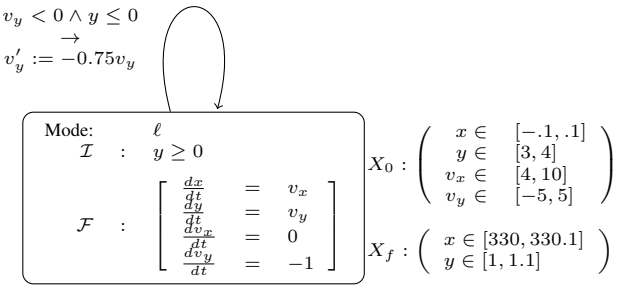

Fig. 3. The hybrid automaton for a bouncing ball with initial set X_0 and unsafe set X_f. The goal is to find whether a trajectory starting from the initial set X_0 can reach the specified unsafe set X_f within $40s$.

step. Furthermore, the resulting constraints are non linear involving trigonometric terms, even if the underlying hybrid systems are linear. Our approach is very much related to constraint solving. However, rather than perform symbolic manipulations to achieve a guaranteed feasible solution that represents a violation, we add an objective function and use numerical optimization techniques based on gradient descent. As a result, our approach results in "optimal" segmented trajectories that may still be disconnected but exhibit gaps that are smaller than a threshold (say 10^{-6}). Nevertheless, we find that such very low cost trajectories are almost always confirmed as real violations through simulation.

The robustness-guided falsification proposed by Fainekos et al. associates each trajectory with a *robustness metric* that measures how close a given trajectory is to a violation and uses robustness as a cost function in a global optimization framework [2]. Robustness metrics for trajectories have been

defined for Metric Temporal Logic (MTL) by Fainekos et al. [20] and for Signal Temporal Logic (STL) by Donze et al [19]. Falsification is achieved by using robustness metrics in conjunction with global optimization techniques such as simulated annealing or the cross-entropy method to search for violations. This approach is implemented in the tool S-Taliro for falsifying MTL properties for Simulink/Stateflow diagrams [3]. Donze et al. also implement a closely related approach that uses robustness and sensitivity analysis to find violations [17]. Like S-Taliro, their approach also searches over the space of single trajectories that form a violation from start to finish. In contrast, our approach minimizes the gaps between trajectory segments. As a result, our approach can exploit the continuous sensitivity to the initial conditions in each mode, while at the same time dealing with discontinuities due to transitions. A recent extension of the robustness-guided falsification approach uses sub-gradient methods to find descent directions for the robustness metric [1]. However, this approach is currently restricted to purely continuous systems.

2) Direct Multiple Shooting: The problem of finding falsifying trajectories is closely related to optimal control. A key distinction lies in the cost functions that are traditionally studied in optimal control problems. Often, the cost functions used exhibits the Bellman optimality principle that allows for a dynamic programming solution. However, work on optimal control has also considered more general *trajectory optimization* problems [25] that define cost functions over the trajectories of a continuous system subject to constraints. A standard approach to trajectory optimization reformulates the problem as a non-linear optimization, and uses numerical optimization techniques [8], [9]. In this regard, the ideas in this paper closely resemble the so-called "direct multiple shooting" approach [11]. The key differences arise from the actual application domains: optimal control problems typically consider non autonomous continuous systems, whereas our work addresses falsification of hybrid systems.

3) Motion Planning and State Space Exploration: As mentioned earlier, techniques from robotic motion planning have been used in the past to find falsifications for hybrid systems [31], [32]. These methods rely on variations of Rapidly-exploring Random Trees (RRT) to iteratively grow from the initial state set towards the final state set (or vice versa and some times employ bi-directional trees). Using a suitable metric for distance and a sampling scheme, simulations are used to find an error trajectory of the system. These approaches have been successful in the case of linear systems, and a few hybrid systems [10], [14], [29], [33], [35]. Hybrid systems pose interesting problems to sequential search because directional information available from the dynamics in each mode need not be useful in the presence of mode switches.

Techniques such as [15], [18], [28] use simulation based approaches to address the problem of uniformly exploring a continuous state space. These techniques typically[1] explore the trajectories in one direction: forward from the initial set, or backwards from the error set.

B. Motivation

We motivate our approach by considering a simple example of bouncing ball hybrid system shown in Fig. 3. Our goal is to find a trajectory that reaches an "unsafe region" X_f as the hyper-rectangle bounded defined by the corners $(330, 1)$ and $(330.1, 1.1)$, starting from the initial set of states, as specified in Fig. 3. Fig. 2 an over-approximation of the set of reachable states computed by the tool Flow* [12]. This over-approximation has a non-empty intersection with X_f, allowing us to conclude a *potential violation*. However, since Flow* over-approximates the actual reachable set of states, we cannot conclude whether a violating trajectory exists, that leads the system from an initial state to an unsafe state. Nevertheless, the reachable set approximation allows us to place bounds on the number of bounces needed (in this case exactly 5 bounces), if at all a violation is to be found.

We used MATLAB$^{\text{TM}}$to perform $100,000$ random simulations requiring roughly 21 minutes on four parallel cores. We found just 3 trajectories violating the property. Likewise, the tool S-Taliro does not find a concrete violation after running for an hour. At best, S-Taliro finds a trajectory with a small positive robustness value using simulation annealing-based optimization, but not a concrete violation.

Our approach assumes a fixed number of bounces (5) and searches for a violation with the mode sequence $\langle \ell, \ell, \ell, \ell, \ell \rangle$. We use a set of trajectory segments $\pi_0, \pi_1, \pi_2, \pi_3, \pi_4$ corresponding to this sequence. We setup an optimization that minimizes the sum of the Euclidean distances between the end point of segment π_i and the starting point of segment π_{i+1}.Using random simulations to find an initial seed, the optimization terminates *within* 2 *seconds*, providing a fully concrete violation with the initial states: $x(0) = -0.09996,\ y(0) = 3.999,\ v_x(0) = 9.8672,\ v_y(0) = 4.968.$

II. Preliminaries

We use *hybrid automata* to model systems that combine discrete and continuous behavior, providing a brief description of its syntax and semantics. More details on this model are available elsewhere [26].

Given a system of Lipschitz continuous, ordinary differential equations (ODEs) $\dot{\mathbf{x}} = f(\mathbf{x})$, we denote its solution at time t by its flowmap $\textsc{Flow}(f, \mathbf{x}_0, t)$. In practice, the ODEs encountered seldom have closed-form analytic solutions. Nevertheless, in most cases, we may approximate FLOW using a numerical ODE solver to a desired degree of precision.

Definition 2.1 (Hybrid Automata): Formally, a hybrid automaton $\mathcal{A}$ is a tuple $(\mathcal{L}, X, \mathcal{F}, \mathcal{I}, \mathcal{G}, \mathcal{R}, \Delta, \mathcal{X}_0)$, where:

- $\mathcal{L}$ is a finite set of *discrete modes*.
- $X \subseteq \mathbb{R}^n$ is the n-dimensional continuous state space. The state space of the hybrid system is $\mathcal{X} \subseteq \mathcal{L} \times X$.

[1]Some variants of RRT-based techniques explore the state-space in a directionless manner, like the techniques proposed here.

- $\mathcal{F}$ maps each mode $\ell \in \mathcal{L}$ with an ODE $\dot{\mathbf{x}} = f_\ell(\mathbf{x})$, where $\mathbf{x} \in X$.
- $\mathcal{I}$ maps each mode ℓ with a *mode invariant* $\mathcal{I}(\ell) \subseteq X$.
- $\mathcal{G}$ is a set of predicates over X.
- $\mathcal{R}$ is a set of functions from X to X.
- $\Delta \subseteq \mathcal{L} \times \mathcal{G} \times \mathcal{R} \times \mathcal{L}$, is a finite set of transitions. For each transition $\delta : (\ell, g_\delta, r_\delta, \ell') \in \Delta$, $g_\delta \in \mathcal{G}$ is its *guard predicate*, and $r_\delta \in \mathcal{R}$ its *reset map*.
- $\mathcal{X}_0 \subseteq \mathcal{L} \times X$ is the set of possible initial states.

The hybrid automaton model, and the techniques presented in this paper, generalize to non-autonomous systems allowing external inputs and time-varying dynamics. However, for simplicity, we restrict our attention to autonomous hybrid automata. The full algorithm will be described in our extended version, available upon request.

A state of the hybrid automaton is a pair $(\ell, \mathbf{x}) \in \mathcal{X}$ wherein $\ell \in \mathcal{L}$ and $\mathbf{x} \in \mathcal{I}(\ell)$. The state evolves over time by interleaving two actions: *flows* and *jumps*.

A *flow* from $(\ell, \mathbf{x})$ to $(\ell, \mathbf{y})$ in time $\tau \geq 0$, denoted by $(\ell, \mathbf{x}) \rightsquigarrow_\tau (\ell, \mathbf{y})$, wherein $\mathbf{y} = \textsc{Flow}(\mathcal{F}(\ell), \mathbf{x}, \tau)$, and $(\forall\, t \in [0, \tau))\ \textsc{Flow}(\mathcal{F}(\ell), \mathbf{x}, t) \in \mathcal{I}(\ell)$.

A *jump* from $(\ell, \mathbf{x})$ to $(\ell', \mathbf{x}')$ is due to a discrete transition $\delta : (\ell, g_\delta, r_\delta, \ell') \in \Delta$, denoted $(\ell, \mathbf{x}) \xrightarrow{\delta} (\ell', \mathbf{x}')$. We require that (a) the transition δ leads from ℓ to ℓ', (b) $\mathbf{x}$ satisfies g_δ, (c) $\mathbf{x}' = r_\delta(\mathbf{x})$ and (d) $\mathbf{x}'$ satisfies $\mathcal{I}(\ell')$. Jumps are considered instantaneous, i.e, no time is assumed to elapse during a jump.

In this work we will consider the problem of finding finite, time bounded counter-examples to safety properties. Let ℓ_f be a distinguished *error* mode of the hybrid automaton $\mathcal{A}$ and $X_f \subseteq \mathcal{I}(\ell_f)$.

Definition 2.2 (Concrete Counterexample): A time-bounded concrete counterexample for a safety property (ℓ_f, X_f) is a finite trajectory:

$$(\ell_0, \mathbf{x}_0) \rightsquigarrow_{\tau_0} (\ell_0, \mathbf{y}_0) \xrightarrow{\delta_0} (\ell_1, \mathbf{x}_1) \rightsquigarrow_{\tau_1} (\ell_1, \mathbf{y}_1) \xrightarrow{\delta_1} \cdots \xrightarrow{\delta_N} (\ell_N, \mathbf{x}_N) \rightsquigarrow_{\tau_N} (\ell_N, \mathbf{y}_N)$$

such that (a) $(\ell_0, \mathbf{x}_0)$ belongs to the initial states $\mathcal{X}_0$, (b) $\mathbf{y}_N \in X_f$, and (c) $\sum_{j=0}^{N} \tau_j \leq T$.

III. Trajectory Splicing

In this section, we explain the idea of trajectory splicing and the setup of the basic optimization problem to achieve perfect splicing. Let $d : X \times X \to \mathbb{R}_{\geq 0}$ denote a *metric* over the continuous states. Common metrics include the L_1, L_2 and L_∞ norms.

Problem Setup. Assume that $\mathcal{A}$ is a hybrid automaton and let $\phi : (\ell_f, X_f)$ represent a safety property of interest. We assume that we are given an abstract counterexample C that is simply a sequence of modes and transitions:

$$C : \langle \ell_0, \delta_0, \ell_1, \delta_1, \cdots, \delta_{N-1}, \ell_N \rangle \tag{1}$$

wherein $\ell_N = \ell_f$. Our goal is to find a concrete counterexample to this property.

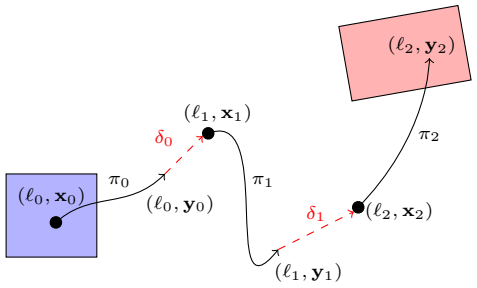

Fig. 4. Illustration of a trajectory segment: initial set is shown shaded in blue and the unsafe set in red.

Trajectory Splicing. We now describe the trajectory splicing approach to this problem. Given the abstract counterexample C, we seek a concrete counterexample

$$(\ell_0, \mathbf{x}_0) \rightsquigarrow_{\tau_0} (\ell_0, \mathbf{y}_0) \xrightarrow{\delta_0} \cdots \xrightarrow{\delta_{N-1}} (\ell_N, \mathbf{x}_N) \rightsquigarrow_{\tau_N} (\ell_N, \mathbf{y}_N) \tag{2}$$

that satisfies the conditions in Def. 2.2. Our approach searches over *segmented trajectories*.

Definition 3.1 (Segmented Trajectories): Given an abstract counterexample $C = \langle \ell_0, \delta_0, \ell_1, \delta_1, \cdots, \delta_{N-1}, \ell_N \rangle$, we define a segmented trajectory S to be a sequence of $N+1$ trajectory segments $\langle \pi_0, \ldots, \pi_N \rangle$:

$$S = \begin{bmatrix} \pi_0 : (\ell_0, \mathbf{x}_0) \rightsquigarrow_{\tau_0} (\ell_0, \mathbf{y}_0) \\ \pi_1 : (\ell_1, \mathbf{x}_1) \rightsquigarrow_{\tau_1} (\ell_1, \mathbf{y}_1) \\ \cdots \\ \pi_N : (\ell_N, \mathbf{x}_N) \rightsquigarrow_{\tau_N} (\ell_N, \mathbf{y}_N) \end{bmatrix} \tag{3}$$

wherein (i) the mode sequence is specified by $C : \langle \ell_0, \ell_1, \ldots, \ell_N \rangle$, (ii) $(\ell_0, \mathbf{x}_0) \in \mathcal{X}_0$, (iii) $(\ell_N, \mathbf{y}_N) \in \mathcal{X}_f$, and (iv) $\mathbf{y}_i \in g_{\delta_i}$ for $i = \{0, \ldots, N-1\}$.

Remark: A segmented trajectory is not, in general, a trajectory of the hybrid automaton. For that to happen, the condition $\mathbf{x}_{j+1} = r_{\delta_j}(\mathbf{y}_j)$ must be satisfied.

The *cost* of a segmented trajectory is defined as the sum of "gaps" between the end points of its consecutive segments.

Definition 3.2 (Cost of a Segmented Trajectory): The cost of a segmented trajectory S is given by the expression:

$$\textsc{Cost}(S) : \sum_{j=0}^{N-1} \underbrace{d(r_{\delta_j}(\mathbf{y}_j), \mathbf{x}_{j+1})}_{\textsc{Cost}(\pi_j, \pi_{j+1})}$$

Note that $d(\mathbf{x}, \mathbf{y}) \geq 0$ for all $\mathbf{x}, \mathbf{y}$. Hence, for any segmented trajectory, $\textsc{Cost}(S) \geq 0$. Clearly, segmented trajectories with zero cost are, in fact, trajectories of $\mathcal{A}$.

Our goal is to setup the search for a zero cost segmented trajectory, as an optimization problem. For a hybrid automaton $\mathcal{A}$, safety property $\varphi : (\ell_f, X_f)$, and sequence of modes and transitions C, let $\mathsf{SegTrajs}(\mathcal{A}, \varphi, C)$ denote the set of segmented trajectories. We wish to find an S with minimum cost using the optimization,

$$\min \textsc{Cost}(S) \ \ s.t.\ S \in \mathsf{SegTrajs}(\mathcal{A}, \varphi, C)$$

$$\min_{\mathbf{x}_0,\tau_0,\ldots,x_N,\tau_N} \sum_{i=0}^{N-1} \textsc{Cost}(\pi_i, \pi_{i+1}) \text{ s.t.} \quad (4)$$

$$\textsc{Cost}(\pi_i, \pi_{i+1}) = d(r_{\delta_i}(\mathbf{y}_i), \mathbf{x}_{i+1}),\ i = \{0 \ldots N-1\} \quad (5)$$

$$\tau_{\min} \leq \tau_i \leq \tau_{\max},\ i = \{0 \ldots N\} \quad (6)$$

$$(\ell_0, \mathbf{x}_0) \in \mathcal{X}_0 \quad (7)$$

$$\mathbf{x}_i, \mathbf{y}_i \in \mathcal{I}(\ell_i),\ i = \{0 \ldots N\} \quad (8)$$

$$\mathbf{y}_i = \textsc{Flow}(\mathcal{F}(\ell_i), \mathbf{x}_i, \tau_i),\ i = \{0 \ldots N\} \quad (9)$$

$$(\forall\ t \in [0, \tau_i))\ \textsc{Flow}(\mathcal{F}(\ell_i), \mathbf{x}_i, t) \in \mathcal{I}(\ell_i),\ i = \{0 \ldots N\} \quad (10)$$

$$(\ell_N, \mathbf{y}_N) \in \mathcal{X}_f \quad (11)$$

$$\mathbf{y}_i \in g_{\delta_i},\ i = \{0 \ldots N-1\} \quad (12)$$

Fig. 5. The optimization problem.

$$\min_{\mathbf{x}_i,\tau_i} \textsc{Cost}(\pi_{i-1}, \pi_i) + \textsc{Cost}(\pi_i, \pi_{i+1}) \text{ subject to:} \quad (13)$$

$$\textsc{Cost}(\pi_{i-1}, \pi_i) = d(r_{\delta_{i-1}}(\mathbf{y}_{i-1}), \mathbf{x}_i) \quad (14)$$

$$\textsc{Cost}(\pi_i, \pi_{i+1}) = d(r_{\delta_i}(\mathbf{y}_i), \mathbf{x}_{i+1}) \quad (15)$$

$$\tau_{\min} \leq \tau_i \leq \tau_{\max} \quad (16)$$

$$\mathbf{x}_i, \mathbf{y}_i \in \mathcal{I}(\ell_i) \quad (17)$$

$$\mathbf{y}_i = \textsc{Flow}(\mathcal{F}(\ell_i), \mathbf{x}_i, \tau_i) \quad (18)$$

$$(\forall\ t \in [0, \tau_i))\ \textsc{Flow}(\mathcal{F}(\ell_i), \mathbf{x}_i, t) \in \mathcal{I}(\ell_i) \quad (19)$$

$$\mathbf{y}_i \in g_{\delta_i} \quad (20)$$

Fig. 6. Optimization instance P_j of trajectory segment π_j for $1 < j < N$. Note that problems P_1, P_N will involve a slightly modified setup.

If the resulting cost is 0 then a violation exists for the property along the given sequence C. Otherwise, the result is strictly positive indicating that no such trajectory exists.

Optimizing Segmented Trajectory Cost. We now consider the problem of optimizing the cost function over set of segmented trajectories. As noted earlier, a segmented trajectory for a given C is entirely determined by the initial segment points $\mathbf{x}_0, \ldots, \mathbf{x}_N$ and the dwell times $\tau_0, \ldots, \tau_N$. The overall optimization setup is as shown in Fig. 5 . The objective function represents the total cost of the segmented trajectory S. Constraint (6) states that the dwell times are non-negative and bounded by τ_{min} and τ_{max}. Constraint (7) enforces that the initial point of the trajectory is in the initial set, (8) states that $\mathbf{x}_i, \mathbf{y}_i$ are within the mode invariants of the respective mode, (9) enforces that $\mathbf{y}_i$ is the continuous state obtained starting from $\mathbf{x}_i$ in time τ_i, (10) states that all intermediate states of the trajectory segments lie inside the mode invariant of ℓ_i, (11) describes that the safety property is violated by the end point, the constraint (12) states that $\mathbf{y}_i$ satisfies the guard of transition δ_i and finally (5) defines the cost $\textsc{Cost}(\pi_i, \pi_{i+1})$ as the gaps between the segments.

Fig. 5 shows a complex optimization problem, which in general is non-linear and non-convex. However, if initialized properly, it can be solved using numerical solvers such as the `fmincon` optimization function available in MATLAB™. To do so, we may numerically approximate the FLOW function using standard ODE solvers. Also, standard sensitivity analysis approaches can be used to compute derivatives $\nabla \mathbf{y}_i$ (w.r.t. $\mathbf{x}_i$) and $\partial \mathbf{y}_i / \partial \tau_i$. This enables the use of gradient-descent and quasi-Newton methods to solve the problem. Furthermore, such a technique places relatively few restrictions on the nature of the automaton $\mathcal{A}$ and the property. We have implemented this basic technique and augmented it to provide derivatives for the objective function and constraints.

The non-linear optimization problem can be directly given to an off the shelf NLP solver such as the `fmincon` function in MATLAB™. Doing so requires a few modifications, however. Significantly, the constraint (10), which enforces the mode invariant, requires special treatment as described at the end of this section.

While directly posing the problem to an off-the-shelf solver is convenient, it may sometimes be inefficient to do so. The size of the optimization problem in terms of the number of decision variables and constraints, depends linearly on the number of state variables in $\mathcal{A}$ and the length of the counter-example sequence. Dependence on the sequence length can be problematic, if the specified sequence is long. We now describe an iterative scheme that decomposes the optimization problem into multiple smaller ones to reduce the number of decision variables and constraints. We show that if the iterative, decomposed scheme converges, it does so to a KKT point of the original optimization problem.

A. Decomposed Scheme

We partition the optimization problem into smaller sub-problems $P_0, \ldots, P_N$ corresponding to the segments $\pi_0, \ldots, \pi_N$. Specifically, problem P_j is allowed to modify the starting point $\mathbf{x}_j$ and the dwell time τ_j of segment π_j, while fixing the decisions for every other segment to their current value. In effect, we optimize the cost of each segment one at a time rather than all together. This reduces the size of each sub-problem P_i to involve just $(\mathbf{x}_i, \tau_i)$, and is thus independent of the length of the counter-example N.

Fig. 6 shows the formulation of P_i involving the trajectory segment π_i in mode ℓ_i. The formulation of each sub-problem P_i is quite similar to that in Fig. 5 with the exception of constraints (7) and (11), which are present only for the first and the last segments π_0, π_N respectively.

Definition 3.3 (Fixed Point Solution): A solution $(x_0^*, \tau_0^*, \ldots, x_N^*, \tau_N^*)$ is said to be a *fixed point solution* for the system of decomposed optimization problems $(P_0, \ldots, P_N)$ iff for each $j \in [0, N]$, (x_j^*, τ_j^*) is a local optimal solution (KKT point) for P_j, after setting $(x_l, \tau_l) = (\mathbf{x}_l^*, \tau_l^*)$ for all $l \neq j$.

To find a fixed point solution, we solve the decomposed problems $P_0, \ldots, P_N$ in some order, until we converge to a solution. In other words, the solutions for the individual optimization instances P_j remain invariant upon further iterations. We now show an important property that relates each KKT point for the original problem in Fig. 5 and to fixed points of the decomposed problem.

We can write the optimization problem in Fig. 5 as

$$\begin{array}{ll} \min & f(\mathbf{x}_1,\tau_1,\mathbf{x}_2)+f(\mathbf{x}_2,\tau_2,\mathbf{x}_3)+\cdots+f(\mathbf{x}_{N-1},\tau_N,\mathbf{x}_N) \\ \text{s.t.} & u_{i,j}(\mathbf{x}_j,\tau_j)\leq 0,\ 1\leq j\leq N, u_{i,j}\in \mathsf{Ineq}_j \\ & v_{k,j}(\mathbf{x}_j,\tau_j)=0,\ 1\leq j\leq N, v_{k,j}\in \mathsf{Eq}_k \end{array} \tag{21}$$

The key observation is that *every single equality/inequality in the original problem (Fig. 5) involves decision variables* $(\mathbf{x}_j,\tau_j)$ *from the same segment* $j\in[0,N]$. Therefore, let Ineq_j and Eq_j represent the set of LHS of inequalities and equalities that involve the decision variables $(\mathbf{x}_j,\tau_j)$ for the j^{th} segment. Also, we conclude by analyzing the constraints in the original problem that the functions $u_{i,j}, v_{k,j}$ involved in these constraints are differentiable.

Let $(\mathbf{x}^*,\tau^*)$: $(\mathbf{x}_0^*,\tau_0^*,\ldots,\mathbf{x}_i^*,\tau_i^*,\ldots,\mathbf{x}_{n-1}^*,\tau_{n-1}^*)$ be a KKT point (local optimum) for the joint optimization problem (Fig. 5, as denoted by (21)). Therefore, we note the existence of dual Lagrange multipliers, $\lambda_{i,j}^*,\mu_{i,j}^*$ corresponding to $u_{ij}\in\mathsf{Ineq}_j, v_{kj}\in\mathsf{Eq}_j$, respectively. We write the KKT conditions as:

$$\begin{pmatrix} \nabla_j f_j(\mathbf{x}_j^*,\tau_j^*,\mathbf{x}_{j+1}^*)+ \\ \nabla_j f_{j-1}(\mathbf{x}_{j-1}^*,\tau_{j-1}^*,\mathbf{x}_j^*)+ \\ \sum_{u_{i,j}\in\mathsf{Ineq}_j}\lambda_{i,j}^*\nabla_j u_{i,j}(\mathbf{x}_j^*,\tau_j^*)+ \\ \sum_{v_{k,j}\in\mathsf{Eq}_j}\mu_{i,j}^*\nabla_j v_{i,j}(\mathbf{x}_j^*,\tau_j^*) \end{pmatrix}=0,$$

$$\begin{array}{rll} u_{i,j}(\mathbf{x}_j^*,\tau_j^*)\leq 0, & u_{i,j}\in\mathsf{Ineq}_j \\ v_{k,j}(\mathbf{x}_j^*,\tau_j^*)=0 & v_{k,j}\in\mathsf{Eq}_j \\ \lambda_{i,j}^*\geq 0 & u_{i,j}\in\mathsf{Ineq}_j \\ \lambda_{i,j}^* u_{i,j}(\mathbf{x}_j^*,\tau_j^*)=0 & u_{i,j}\in\mathsf{Ineq}_j \\ \ldots & 0\leq j\leq N \end{array} \tag{22}$$

Here ∇_j refers to the gradient w.r.t $(\mathbf{x}_j,\tau_j)$.

We now separately consider the solution s_j : $(\mathbf{x}_j^*,\tau_j^*)$ for the decomposed problem P_j (see Fig. 6) having fixed the solutions for all other segments π_l for $j\neq l$ to $\mathbf{x}_l=\mathbf{x}_l^*,\tau_l=\tau_l^*$. Following the form in Eq. (21), we obtain P_j as follows ($j=0,N$ have a slightly different form):

$$\begin{array}{ll} \min & f_{j-1}(\mathbf{x}_{j-1}^*,\tau_{j-1}^*,\mathbf{x}_j)+f_j(\mathbf{x}_j,\tau_j,\mathbf{x}_{j+1}^*) \\ \text{s.t.} & u_{i,j}(\mathbf{x}_j,\tau_j)\leq 0,\ u_{i,j}\in\mathsf{Ineq}_j \\ & v_{k,j}(\mathbf{x}_j,\tau_j)=0,\ v_{k,j}\in\mathsf{Eq}_k \end{array} \tag{23}$$

We can now write down the KKT constraints for this problem and note that (a) the KKT conditions from Eq. (22) can be decomposed into N set of contraints by fixing $j=0,\ldots,N$ and yields the KKT conditions for each P_j, and (b) the KKT conditions expressing a fixed point solution for $(P_0,\ldots,P_N)$ are conjoined to yield a KKT condition in Eq. (22).

Theorem 3.1 (Equivalence of KKT Points): $(\mathbf{x}^*,\tau^*)$ is a KKT point for the problem Fig. 5 iff $(\mathbf{x}^*,\tau^*)$ is a fixed point for the decomposed problem in Fig. 6

In summary, the decomposed problem when solved iteratively for its constituent sub-problems converges to a (local) minimum of the original problem, if the optimization of each of its sub-problem converges.

Remarks. Instead of splitting for every trajectory segment, several segments can be group together as a sub-problem; the tradeoff being the problem size and the number of iterations needed for convergence. Also, such a formulation can be easily parallelized.

B. Handling Mode Invariants

Finally, we consider the problem of dealing with constraint (10). The constraint can (a) either be entirely dropped, (b) relaxed, or (c) handled directly on the fly. Dropping the constraint may result in solutions that are unacceptable; this can be remedied by multiple re-initialization of the problem until an acceptable solution is found. Though this approach is the simplest, it may be unsatisfactory, depending on the problem. Relaxing the constraint by enforcing only the end points of a trajectory segment to satisfy the mode invariant is another option. This relaxation can be tightened arbitrarily by further discretizing the trajectory space (splitting a trajectory segment into multiple segments) and letting more than one trajectory segment to be inside a mode, while enforcing the end points to satisfy the respective mode invariants. The constraint can also be directly handled by using a combination of a simulator and a zero crossing detector as the constraint function. It should be noted that such a direct formulation can be highly non-linear, discontinuous and lead to poor performance.

IV. Experimental Results

We present a prototype implementation and a preliminary evaluation of the various ideas presented in this paper.

A. Implementation

The techniques presented were implemented using MATLAB™. Given the description of a hybrid system $\mathcal{A}$, a safety property φ, and an abstract counter-example C as a sequence of modes/transitions, the falsification problem is reformulated as an optimization problem. The abstract counter-example C may be obtained by running the flowpipe construction tool such as SpaceEx [24], or alternatively by enumerating all likely mode/transition sequences up to a given length. Our implementation uses the latter combined with some user guidance to narrow the search space. For each mode sequence, an optimization problem is initialized through random simulation in each mode; which may not be feasible to begin with. Our approach utilizes the generic `fmincon` optimization function available in MATLAB™.

The gradients of the objective function and constraints are supplied to `fmincon` in order to improve the speed of convergence. Given K mode sequences $C_1,\ldots,C_K$, we run the optimization instances for each C_i in parallel. For all practical purposes, a violation is concluded within a small tolerance $\epsilon=10^{-3}$ for the cost. The found counterexamples are confirmed by simulating from the initial condition $\mathbf{x}_0$ with the obtained mode dwell times(τ_i).

B. Benchmarks

We test our approach on the NAV benchmarks that are commonly used to evaluate hybrid system verification tools [21]. These benchmarks model a particle traveling on a 2 dimensional grid of cells. The continuous dynamics of each cell are described by set of affine ODEs. A transition is taken when the particle crosses over from one cell to its neighbor. All instances of the benchmark designate an initial

Fig. 7. 10,000 Simulations and the target cells P, Q, R and S.

set of states and an error cell. To test our approach we focus on the "NAV-30" instance which has $25 \times 25 = 625$ cells or discrete modes and roughly 2500 transitions. To make the falsification more challenging, we modify initial set of states to $X_0 : \ x \in [4, 5], y \in [21, 22], v_x \in [-1, 1], v_y \in [-1, 1]$.

We carefully choose safety properties P, Q, R and S, wherein each safety property is simply a cell in the grid. The cells P, Q, R and S are detailed in Table I. We characterize the difficulty of violating each property by a "degree of difficulty" (DoD) metric by running $N = 100,000$ random simulations and recording the number of error trajectories (plotted in Fig. 7). We note that P was violated by 3, Q by 2, R by 1 and S by 105 simulation traces, respectively. The challenge lies in whether our optimization scheme can discover the violations given some likely mode sequences.

TABLE I
EXPERIMENTAL RESULTS

Property	DOD	Time[1]		
		S-Taliro	Direct	Decomp. (Iters.)
P: $x \in [6, 7], y \in [7, 8]$	3	482	0.3	290 (848)
Q: $x \in [7, 8], y \in [9, 10]$	5	2443	0.5	113 (341)
R: $x \in [1, 2], y \in [6, 7]$	1	474	3	285 (704)
S: $x \in [22, 23], y \in [11, 12]$	110	564	8	1721 (2680)

[1]All times were measured in seconds on a laptop with Intel Core i7-2820Q 2.30GHz processor (x86_64 arch) and 8GB RAM running Ubuntu 11.04 Linux 3.2.0-34. Each falsification was verified using simulations. Time for 100,000 simulations running in parallel was 200 mins.

Table I summarizes the results of running our approaches and the comparison with S-Taliro [3]. We compare two optimization schemes on the benchmarks: the direct formulation using `fmincon` (direct) function, and the decomposed per mode formulation `fmincon` (decomposed). Within `fmincon`, SQP is selected as the algorithm of choice. Each falsification instance considers multiple mode sequences. The table reports only the results over sequences that led to error trajectories. Other mode sequences yielded final costs in the range $[0.5, 1]$, markedly higher than the successful sequences. In most instances, a small number of gradient descent iterations sufficed to distinguish a plausible mode sequence from a spurious one, suggesting a a fast heuristic for detecting such sequences.

The results also suggest that the direct optimization using an off-the-shelf solver (`fmincon` with SQP) outperforms the decomposed method. This is mainly due to the large number of iterations required by the decomposed method to achieve convergence. We posit that the efficacy of the decomposed scheme may be evident on counter-examples with long mode sequences, and merits further investigation. . We are currently working to confirm this by trying our approach on a wider variety of systems. Finally, better methods for reducing the number of iterations, and thus accelerting the convergence of the decomposed method remain to be investigated.

We also use S-Taliro for comparison and tabulate its results alongside. The S-Taliro column in Table I describes the least time taken for falsification for two runs (with the simulated Annealing algorithm and 10,000 simulations).The obtained trajectory is verified in all cases using a simulator. As discussed above, the table also includes a DOD column for every property, that is the number of randomly selected samples succeeding in falsifying the property against a total of 100,000 samples. Clearly, the smaller this number, the more difficult it is to test the property using random testing.

C. Case Study: Insulin-Glucose Control

In this section we study an *artificial pancreas* controller that seeks to maintain safe levels of plasma blood glucose in type I diabetic patients. The insulin-glucose dynamics are modeled using the Bergman minimal model with 3 state variables G, I, X [6], denoting the plasma glucose, plasma insulin and remote chamber insulin concentrations, respectively. We use two hybrid automata models for the closed loop system, for our study. The controller design was originally proposed by Fisher [22]. The two models used in our evaluation differ in their treatment of glucose absorption dynamics. *Model-A* uses linear hybrid glucose gut absorption dynamics, and is taken from our previous work [12]. This is a 6 mode hybrid automaton. *Model-B* has two discrete modes, uses an exponential glucose absorption sub-model, and is taken from Fisher's original paper [22].

Using both models, we search for scenarios of hyperglycemia, where the plasma blood glucose levels increases beyond 22 mmol/L and hypoglycemia, where it dips below -0.9 mmol/L. Each model consists of upto 7 patient parameters. The parameters for model-A are described by Chen et al. [12]. However, we extend the ranges of some parameters for our study here Model-B is evaluated over parameter sets describing 18 different subjects, as described by Bergman [7]. Our model included an uncertainty interval for each of the patient parameters. We report results, for cases where we could find a falsification.The models used and the full results are available upon request.

Results. For Model-B, Table II lists the initial values and parameters which lead to hyperglycemia ($G \geq 22$ mmol/L) for five subjects and hypoglycemia for one subject The

TABLE II
CASE STUDY

Prop.	ID	time[2]	DOD	$\{G0, I0\}, \{B, k\}$
$G \geq 22$	6	0.5	0	{19.73,56.67},{0.69,0.03}
	14	8.0	0	{14.93,170.98},{0.7,0.03}
	16	0.6	6	{17.79,198.41},{0.68,0.038}
	17	0.6	293	{16.13,20.02},{0.63,0.037}
	18	0.6	84	{16.93,0,13.28},{0.63,0.034}
$G \leq -0.9$	1	1	0	{15.17,399.6},{0.2,0.07}

[2]Time taken by `fmincon`. Time to compute DoD (on 4 parallel cores) took approximately 1 hour. In all cases $X_0 = 0$.

table also reports te degree of difficulty (DoD) for each of these falsification problems. In two cases, 10,000 random simulations were unable to find the violation in question, whereas our technique does.

For Model-A, we found a possible hyperglycemia scenario. The search for this scenario terminated in 2.7s with DoD = 294. In this instance, the 10,000 random simulations took 7470 mins, many times slower than our approach.

V. Conclusion and Future Work

To conclude, we have presented the general framework of trajectory splicing, specialized it to linear hybrid automata and demonstrated a promising preliminary evaluation of our approach against existing falsification tools such as S-Taliro. Our future work, will further explore the space of optimization approaches for this problem over a larger set of linear as well as non-linear benchmarks. Extensions to handle non-autonomous systems are also planned.

VI. Acknowledgments

This work was supported in part by the US National Science Foundation (NSF) under award numbers CNS-0953941, CNS-1016994 and CPS-1035845. All opinions expressed are those of the authors and not necessarily of the NSF.

References

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

[2] H. Abbas, G. Fainekos, S. Sankaranarayanan, F. Ivancic, and A. Gupta. Probabilistic temporal logic falsification of cyber-physical systems. *Trans. on Embedded Computing Systems (TECS)*, 12(2s):95, 2012.

[3] Y. Annapureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. *Proc. TACAS*, pages 254–257, 2011.

[4] G. Behrmann, A. David, and K. G. Larsen. A tutorial on UPPAAL. In *SFM-RT 200 (Revised Lectures)*, volume 3185 of *LNCS*, pages 200–237. Springer Verlag, 2004.

[5] L. Benvenuti, D. Bresolin, A. Casagrande, P. Collins, A. Ferrari, E. Mazzi, A. Sangiovanni-Vincentelli, and R. Villa. Reachability computation for hybrid systems with Ariadne. In *Proc. the 17th IFAC World Congress*, 2008.

[6] R. N. Bergman. Toward physiological understanding of glucose tolerance: minimal-model approach. *Diabetes*, 38(12):1512–1527, 1989.

[7] R. N. Bergman, L. S. Phillips, and C. Cobelli. Physiologic evaluation of factors controlling glucose tolerance in man: measurement of insulin sensitivity and beta-cell glucose sensitivity from the response to intravenous glucose. *Journal of Clinical Investigation*, 68(6):1456, 1981.

[8] J. T. Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

[9] J. T. Betts. *Practical methods for optimal control and estimation using nonlinear programming*, volume 19. Siam, 2010.

[10] A. Bhatia and E. Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. *Proc. of HSCC*, page 451–471, 2004.

[11] H. G. Bock and K.-J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. 1983.

[12] X. Chen, E. Abrahám, and S. Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. In *Real-Time Systems Symposium (RTSS)*, pages 183–192. IEEE, 2012.

[13] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.

[14] T. Dang, A. Donzé, O. Maler, and N. Shalev. Sensitive state-space exploration. In *Proc. CDC'08*, pages 4049–4054. IEEE, 2008.

[15] T. Dang and T. Nahhal. Coverage-guided test generation for continuous and hybrid systems. *Formal Methods in System Design*, 34(2):183–213, 2009.

[16] L. M. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.

[17] A. Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Proc. CAV*, pages 167–170, 2010.

[18] A. Donzé and O. Maler. Systematic simulation using sensitivity analysis. *Proc. of HSCC*, pages 174–189, 2007.

[19] A. Donzé and O. Maler. Robust satisfaction of temporal logic over real-valued signals. In *Proc. FORMATS*, pages 92–106, 2010.

[20] G. E. Fainekos, A. Girard, and G. J. Pappas. Temporal logic verification using simulation. In *FORMATS*, volume 4202 of *LNCS*, pages 171–186. Springer, 2006.

[21] A. Fehnker and F. Ivanĉić. Benchmarks for hybrid systems verification. In *Proc. of HSCC*, volume 2993, pages 326–341, 2004.

[22] M. E. Fisher. A semiclosed-loop algorithm for the control of blood glucose levels in diabetics. *Biomedical Engineering, IEEE Transactions on*, 38(1):57–61, 1991.

[23] M. Fränzle, C. Herde, S. Ratschan, T. Schubert, and T. Teige. Efficient solving of large non-linear arithmetic constraint systems with complex Boolean structure. *Journal on Satisfiability (JSAT)*, 1:209–236, 2007.

[24] G. Frehse, C. Le 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 *Proc. CAV*, LNCS. Springer, 2011.

[25] C. R. Hargraves and S. Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics*, 10(4):338–342, 1987.

[26] T. A. Henzinger. The theory of hybrid automata. In *LICS'96*, pages 278–292. IEEE, 1996.

[27] C. Herde, A. Eggers, and T. Franzle, M. Teige. Analysis of hybrid systems using HySAT. In *Proc. ICONS 08.*, pages 13–18. IEEE, 2008.

[28] J. Kapinski, B. H. Krogh, O. Maler, and O. Stursberg. On systematic simulation of open continuous systems. In *HSCC*, pages 283–297, 2003.

[29] J. Kim, J. M. Esposito, and V. Kumar. An RRT-based algorithm for testing and validating multi-robot controllers. Technical report, DTIC Document, 2005.

[30] A. A. Kurzhanskiy and P. Varaiya. Ellipsoidal toolbox. Technical Report UCB/EECS-2006-46, EECS Department, University of California, Berkeley, May 2006.

[31] S. M. LaValle. Rapidly-exploring random trees a new tool for path planning. Technical Report TR 98-11, CS Dept., Iowa State University, 1998.

[32] S. M. LaValle and J. J. Kuffner Jr. Randomized kinodynamic planning. In *Proc. ICRA*, volume 1, pages 473–479. IEEE, 1999.

[33] T. Nahhal and T. Dang. Test coverage for continuous and hybrid systems. In *Computer Aided Verification*, page 449–462, 2007.

[34] N. S. Nedialkov and M. von Mohrenschildt. Rigorous simulation of hybrid dynamic systems with symbolic and interval methods. In *Proc. of ACC*, pages 140–146. IEEE, 2002.

[35] E. Plaku, L. Kavraki, and M. Vardi. Falsification of LTL safety properties in hybrid systems. *Proc. TACAS*, page 368–382, 2009.

[36] A. Tiwari. Hybridsal relational abstracter. In *Computer Aided Verification*, CAV'12, pages 725–731. Springer-Verlag, 2012.