Computing Bounded Reach Sets from Sampled Simulation Traces *

Zhenqi Huang
University of Illinois at Urbana-Champaign
247 CSL
1308 W. Main Street
Urbana, Illinois, 61801
zhuang25@illinois.edu

Sayan Mitra
University of Illinois at Urbana-Champaign
266 CSL (Mail Code: 228)
1308 W. Main Street
Urbana, Illinois, 61801
mitras@illinois.edu

ABSTRACT

This paper presents an algorithm which uses simulation traces and formal models for computing overapproximations of reach sets of deterministic hybrid systems. The implementation of the algorithm in a tool, *Hybrid Trace Verifier (HTV)*, uses Mathwork's Simulink/Stateflow (SLSF) environment for generating simulation traces and for obtaining formal models. Computation of the overapproximation relies on computing error bounds in the dynamics obtained from the formal model. Verification results from three case studies, namely, a version of the navigation benchmark, an engine control system, and a satellite system suggest that this combined formal analysis and simulation based approach may scale to larger problems.

Categories and Subject Descriptors

D.2.5 [**Software Engineering**]: Testing and Debugging—*Symbolic execution*

Keywords

verification, hybrid automata

1. INTRODUCTION

For a deterministic hybrid automaton, an initial state and a time-bound uniquely determines an execution. Denote $\alpha(x_0, t)$, α for short, as an execution starts at x_0 and evolves for T time. For the purposes of safety verification, we would like to compute the set of states that are reached by this execution α. Exact computation of this set is intractable, and for several special classes of automata approximation algorithms have been proposed (see, for example, [2, 4, 10]). On the other hand, simulation tools, such as Mathwork's Simulink/Stateflow (SLSF), effectively compute simulation traces from the models. In this paper, we explore how such simulation traces can aid verification.

Our main contribution is an algorithm for computing an overapproximation of the reach set of a specific execution from simulation traces. For the purposes of discussion, let us fix an execution α and a particular simulation trace $x_0, x_1, \ldots, x_l$. In using the simulation engine, Matlab's ODE solvers for example, the user can set an *absolute tolerance*. For a given absolute tolerance of $c > 0$, the simulation engine guarantees that if a simulation trace point x_k is on α, then the following simulation point x_{k+1} is within c distance of α. This does not, however, provide any guarantees about the overall accumulated error nor about the error between successive simulation points. In this paper, we show how these errors can be bounded using information from a formal model of the system. In particular, we use continuity and the Lipschitz constants of the functions describing the differential equations and resets. The algorithm is implemented in a tool called *Hybrid Trace Verifier (HTV)*. HTV translates SLSF models to hybrid automata using HyLink [12] and uses the simulation traces in conjunction with the automaton for analysis. We applied HTV to three examples: (1) a navigation benchmark, (2) an autonomous satellite system, and (3) an engine control system with upto 4 continuous state variables and 9 locations. For these relatively small examples HTV successfully generates overapproximations for both linear and non-linear systems and verifies safety properties in seconds. This leads us to contemplate that this method may scale to larger systems.

Related Work. SLSF models are now widely used in model-based real-time system design. Several recent papers address the problem of formally verifying SLSF models from the simulation traces generated by the models. The verification approach presented in [8] generates a symbolic trace from a given simulation trace by carefully instrumenting the model. The symbolic traces are then used to compute a set of initial states that from which all traces visit the same sequence of locations. In this work we do not instrument the model and we do not assume that the trace generated by the simulation engine is arbitrarily accurate. The approach presented in [1] and [5] searches for counterexamples to Metric Temporal Logic (MTL) properties for (possibly non-linear) hybrid systems through minimization of a robustness metric. The global minimization is carried out using Monte-Carlo techniques. In contrast, our analysis approach is deterministic and currently targets only safety properties. In our previ-

*This research is supported by a research grant from John Deere Company and NSF (Contract No CNS-1016791).

HSCC12, April 17–19, 2012, Beijing, China.

ous work, we developed HyLink [11, 12] which is a tool for translating SLSF models to hybrid automata. Guaranteed ODE solvers have been developed to compute error bounds on the solutions of differential equations. For example, the C++ library in [3] solves an initial value problem and computes a discrete sequence of error terms which upper bound the distances to the true solution at each sampled point. Our tool on the other hand is integrated with the modeling environment and computes a tube containing the true execution explicitly. Furthermore, our approach handles hybrid models directly.

2. PRELIMINARIES

In this paper, we discuss a class of deterministic hybrid automata; see [9, 13] for a detailed exposition. Let V as a set of variables. Each variable $v \in V$ is associated with a type, $type(v)$, which defines the set of values v can take. A valuation $\mathbf{v}$ maps each $v \in V$ to a value in $type(v)$. By $\mathbf{v}.x$ and $\mathbf{v}.loc$ we refer to the values of the variables x and loc at $\mathbf{v}$. We denote $val(V)$ as the set of all valuations of V. For a set of variables V a trajectory τ is a function $\tau : [0, t] \mapsto val(V)$, where t is a non-negative real. We define $\tau.fstate = \tau(0), \tau.lstate = \tau(t), \tau.dur = t$. By vector or matrix norm $||\cdot||$ we refer to infinity norm.

Definition 1. *A Hybrid Automaton $\mathcal{A}$ is a tuple $(V, \mathcal{L}, Q, q_0, Grd, Inv, \mathcal{T})$ where:*

1. *$V = X \cup \{loc\}$ is a set of variables, where loc is a discrete variable of finite type $\mathcal{L}$ called the set of* locations*, and each $x \in X$ is a continuous real-valued variable. $Q \subseteq val(V)$ is a set of states.*
2. *$Grd : \mathcal{L} \times \mathcal{L} \to 2^{val(X)}$, is a function that maps each pair $i, j \in \mathcal{L}$ to a set in $val(X)$. A transition $\mathbf{v} \to \mathbf{v}'$ can occur iff $\mathbf{v}.X \in Grd(\mathbf{v}.loc, \mathbf{v}'.loc)$ and $\mathbf{v}.X = \mathbf{v}'.X$, and we write this transition as $\mathbf{v} \to \mathbf{v}'$. That is, the transitions do not change the continuous state.*
3. *$Inv : \mathcal{L} \mapsto 2^{val(X)}$ is a function that maps each location to a set of states, called invariant set.*
4. *$\mathcal{T}$ is a set of trajectories for V such that along each $\tau \in \mathcal{T}$ (i) loc remains constant and (ii) the continuous variables X evolve according to a differential equation: $\dot{X} = f_{\tau(0).loc}(X)$.*

Semantics. An execution captures a particular run of the hybrid automaton. In this paper, we consider hybrid automata that are *deterministic*. That is, from any state there is at most one transition or a unique trajectory of non-zero duration. Formally, an *execution* starting from a state $\mathbf{v}_0$ is a finite sequence of trajectories $\tau_0\tau_1 \cdots \tau_n$, where for each i, $\tau_i.lstate \to \tau_{i+1}.fstate$ and $\tau(0) = \mathbf{v}_0$. We define the duration of an execution as $\sum_i \tau_i.dur$. For an execution of duration at least t starting from $\mathbf{v}_0$, we denote the valuation of the variables at time t by $\alpha(\mathbf{v}_0, t)$. A state $\mathbf{v} \in Q$ is said to be reachable from $\mathbf{v}_0$ if there exists t such that $\alpha(\mathbf{v}_0, t) = \mathbf{v}$.

We will make the following additional assumptions throughout this paper.

Assumption 1 (Dwell time). *There exists a minimum dwell time $\Delta > 0$ such that along any execution, no two discrete transitions occur within Δ time of each other.*

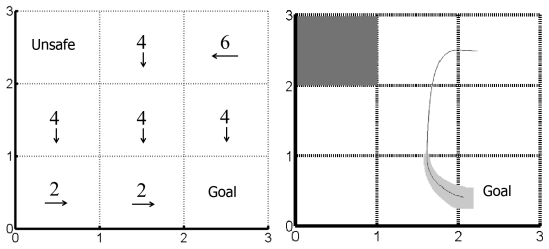

Figure 1: Overapproximation of reach set for the navigation system.

Assumption 2 (Lipschitz). *For each location $i \in \mathcal{L}$, the function f_i (the right hand side of the differential equation) is Lipschitz with a Lipschitz constant $L_i > 0$.*

Assumption 3 (Bounded derivative difference). *For $i, j \in \mathcal{L}$, there exists an upper bound $D_{i,j}$ such that for all $y \in Grd(i, j)$, $||f_i(y) - f_j(y)|| \leq D_{i,j}$.*

Assumption 1 is satisfied by most well-designed systems that do not have Zeno executions. This ensures that along any execution, at a given time there is a unique valuation of the variables. Assumption 2 is satisfied if f_i has a bounded derivative over $Inv(i)$. If the guards are compact then Assumption 2 implies Assumption 3.

Example 1 We consider a version of the 3×3 navigation benchmark [7]. This is a deterministic hybrid automaton $\mathcal{A}$ with 4 continuous variables $X = (x, y, vx, vy)$ and 9 locations $\mathcal{L} = \{(i, j) | i, j \in \{1, 2, 3\}\}$. The system evolves according to differential equation $\dot{X} = AX - Bu(i, j)$, where

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1.2 & 0.1 \\ 0 & 0 & 0.1 & -1.2 \end{bmatrix} \text{ and } B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1.2 & 0.1 \\ 0.1 & -1.2 \end{bmatrix}.$$

The input u is assigned to be a constant in each modes $u(i, j) = [\sin((\pi c(i, j))/4), \cos((\pi c(i, j))/4)]^T$, where $c(i, j) \in \{0, 1, \cdots, 7\}$ determines a direction vector of location (i, j).

The different regions and an execution is visualized in Figure 1. In this example, the Lipschitz constants of all the locations equal $L_i = ||A|| = 1.3$. For two locations the difference in the derivatives is bounded by $D = 2||B||$.

Next, we define c-sampled traces which are related to the simulation traces we can get from SLSF models.

Definition 2. *For a hybrid automaton $\mathcal{A}$, a start state $\mathbf{v}_0$, a step error constant $c > 0$, and a time-bound $t_l > 0$, a c-sampled trace β is a finite (V, T)-sequence $\beta = (\mathbf{v}_0, t_0), (\mathbf{v}_1, t_1), \cdots, (\mathbf{v}_l, t_l)$, for each $k \in \{0, \cdots, l-1\}$,*

$$||\alpha(\mathbf{v}_k, t_{k+1} - t_k).X - \mathbf{v}_{k+1}.X|| \leq c.$$

That is, the $(k+1)^{st}$ sample point in the trace is at most c distance away from the execution starting from the sampled trace at k.

3. OVERAPPROXIMATION OF REACH SET

From a given c-sampled trace β we want to compute the set of all the states that are reachable by any execution that

can generate β. Our approach involves two steps: first, we find the accumulated simulation error corresponding to every step, and then we propagate a tube around each sampled point in β that is guaranteed to contain all executions that may have generated β.

3.1 Estimating Accumulated Error

We discuss how fast the error possibly diverges from the real trace given a c-sampled trace. For the remainder of the paper, we fix the initial state $\mathbf{v}_0$ and the sampled trace β of length $l+1$. Let $\{\epsilon_k\}_0^l$ be a collection of error bounds for β that satisfy the following: for $k \in \{0, \ldots, l\}$, $||\alpha(\mathbf{v}_0, t_k).X - \mathbf{v}_k.X|| \leq \epsilon_k$. That is, the true state of the execution starting from $\mathbf{v}_0$ at time t_k is within ϵ_k distance of the k^{th} sampled point $\mathbf{v}_k.X$. Using the following lemma, $\{\epsilon_k\}$ are computed iteratively (starting from a given ϵ_0) provided transitions do not occur in any of the $[t_k, t_{k+1}]$ intervals.

Lemma 1. *If $\forall t \in [t_k, t_{k+1}]$, $\alpha(\mathbf{v}_0, t).loc = i$ for some $i \in \mathcal{L}$, then*

$$\epsilon_{k+1} \leq \epsilon_k e^{L_i(t_{k+1}-t_k)} + c.$$

In the special case of a linear autonomous hybrid system we have the following corollary.

Corollary 1. *If $\forall t \in [t_k, t_{k+1}]$, $\alpha(, \mathbf{v}_0, t).loc = i$ and the continuous states evolve follows $\dot{X} = A_i X + B_i$, Then*

$$\epsilon_{k+1} \leq \epsilon_k ||e^{A_i(t_{k+1}-t_k)}|| + c.$$

On the other hand, if a transition occurs in the interval $[t_k, t_{k+1}]$, the following error bound holds.

Lemma 2. *If there exists $\eta \in [t_k, t_{k+1}]$ such that a single transition from location i to j occurs in $\alpha(\mathbf{v}_0)$ at time η, then there exists a constant $M > 0$ such that*

$$\epsilon_{k+1} \leq (\epsilon_k + M/L_j)e^{L(t_{k+1}-t_k)} - M/L_j + c,$$

where $L = max\{L_i, L_j\}$.

Setting $M = \sup_{x \in Inv(i) \cup Inv(j)} ||f_i(x) - f_j(x)||$ satisfies the above condition, but a smaller range for x based on reachability analysis from i can provide tighter bounds.

Lemmas 1 and 2 provides a sequence of error bounds $\{\epsilon_k\}$ such that $\forall\ k \in \{0, \cdots, l\}$, $||\alpha(\mathbf{v}_0, t_k).X - \mathbf{v}_k.X|| \leq \epsilon_k$. We define the sequence γ from β by attaching these error bounds. $\gamma = (v_0, t_0, \epsilon_0)(v_1, t_1, \epsilon_1) \cdots (v_l, t_l, \epsilon_l)$.

3.2 Reach Set Between Sampled Points

With a given sampled trace with error bounds (γ) we compute a tube containing all reachable states any execution that may generate the sampled trace β. This tube is the union of all tube segments between every two consecutive sample points. To build the tube containing the real execution,we execute Algorithm 1.

In this algorithm, the k^{th} tube segment is overapproximated by a ball B' centered at $\mathbf{v}_k.X$ with the radius ϵ'. The initial estimate of ϵ' is ϵ_k and in each iteration this estimate is bloated by a constant factor $b > 1$ until the terminating condition holds. The terminating condition requires that ϵ' is less than the original error estimate ϵ_i added to the maximum possible increase in the error ($m(t_{i+1} - t_i)$. Assumption 2 guarantees that if the sample period satisfies $L(t_{k+1} - t_k) < 1$ then this loop is going to terminate. The union of all B's, the set R, contains the real execution and is returned by the algorithm.

```
R ← ∅ ;
for i ← 0 to l − 1 do
    ε' ← ε_i;
    do
        ε' ← b ∗ ε';
        B' ← B(v_i.X, ε');
        m ← sup_{x∈B'} ||f(x)||;
    while ε' < m ∗ (t_{i+1} − t_i) + ε_i;
    R ← R ∪ B';
end
return R;
```

Algorithm 1: Overestimation of Reach Set

4. IMPLEMENTATION AND EXPERIMENTAL EVALUATION

We developed a tool, *Hybrid Trace Verifier (*HTV *)*, that implements the above algorithms for over-approximating the reach set based on a sampled trace. First, from a SLSF model a sampled trace is generated. In addition, a hybrid automaton translation representing the SLSF model is obtained using the HyLink tool [12]. HTV can handle both linear and non-linear hybrid automata. We applied HTV to verify safety properties of the navigation benchmark [6], a satellites system, and an engine control system. Performance of the tool is shown in Table 1; discussions of these results appear below.

An instance of navigation benchmark is given earlier in Example 1. The reach set computed by HTV for this instance of the navigation benchmark is shown in Figure 1. The top left region is unsafe, the real execution is shown by the dark line, and the gray region shown the overapproximation of the reach set.

Our second example is a hybrid system model of two satellites—one active and the other passive. The active satellite changes its orbit and the passive satellite remains on the same orbit. The unsafe set corresponds to the set of positions in which the two satellites are too close. The continuous state of the system $X = (\nu_1, \nu_2)$ captures the angular position of the two satellites. The discrete state $\mathcal{L} = \{1, 2\}$ captures the two possible orbits of the active satellite. The guards model the relative angular positions at which the orbital transition must occur. The differential equation for the passive satellite is $\dot{\nu_1} = \frac{1.0077}{(1+0.3\cos(\nu_1))^2}$. The differential equations for the the active satellite is $\dot{\nu_2} = \frac{0.9933}{1+0.3\cos(\nu_2)^2}$ in orbit 1 and $\dot{\nu_2} = \frac{1.0446}{(1+0.25\cos(\nu_2-0.44))^2}$ in orbit 2.

The computed reach set is shown in Figure 2. The figure on the left is the reach set in Cartesian coordinate. The darker area contains the execution of passive satellite and the lighter area contains the execution of active satellite. The figure on the right is the phase portrait. The darker tiny rectangle denotes the unsafe region near the intersection of the two orbits. We observe that in the absence of discrete transitions the error bound increases according to Lemma 1. A non-zero initial error bound will expand exponentially. After a transition at step i, according to Lemma 2, an $O(\delta)$ term adds to ϵ_i which increases the growth rate of the error bound. The passive satellite does not have transitions and therefore its overapproximation for the reach set is tighter than that of the active satellite. Nevertheless, the

Table 1: Tool performance

Benchmark	#variables,locations	Real time	#sample points	Sim time	Reach time	#transitions	max error
Navigation I	4,4	2s	8030	8s	15s	2	2.0e-3
Navigation II	4,9	4s	16036	15s	35s	4	0.275
Satellites	2,2	6.5s	52046	15s	30s	1	0.3475
Engine control	4,2	5s	2026	5s	10s	2	13.27

overapproximation of the combined reach set does not intersect with the unsafe states in this example and safety is verified.

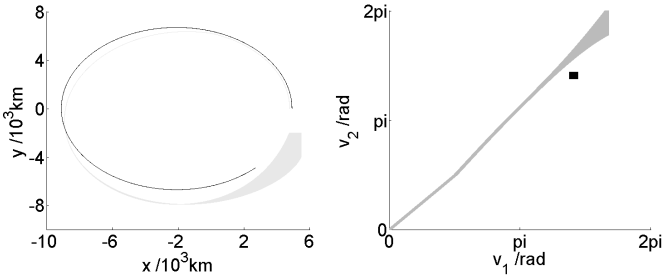

Figure 2: Overapproximation of reach set for the satellite system.

Our final example is a hybrid model of a linear engine control system with 4 continuous variables $X = (n_f, n_c, x_1, x_2)$ and 2 locations $\{1, 2\}$. The dynamics in each location is $\dot{X} = A_i X + u_i$, $i \in \{1, 2\}$. Where,

$$A_1 = \begin{bmatrix} -3.961 & 0.7344 & 672.7 & 0 \\ -3.704 & -1.774 & 1437 & 0 \\ -0.004285 & 0 & 0 & 0 \\ -0.01497 & 0.007887 & 5.543 & -5.425 \end{bmatrix}, u_1 = \begin{bmatrix} 1973 \\ 4257 \\ 2.354 \\ 11.92 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} -2.145 & 0.4919 & 0 & 672.7 \\ 0.1758 & -4.394 & 0 & 1437 \\ -0.0301 & -0.02322 & -12.74 & 12.74 \\ 0 & -0.002217 & 0 & 0 \end{bmatrix}, u_2 = \begin{bmatrix} 699 \\ 1493 \\ -22.14 \\ 1.264 \end{bmatrix}$$

The invariants of locations are $Inv(1) : w \leq 0$, $Inv(2) : w > 0$. Where $w = -0.0027 n_f + 0.001823 n_c + x1 - x2 + 1.0468$. The required safety property is that n_c remains below a threshold.

In this case the A_i matrices are Hurwitz. In each location, different executions converge to each other. According to Corollary 1, the error bound shrinks when $\epsilon_k > \frac{c}{1-||e^{A(t_{k+1}-t_k)}||}$.

The overall performance of HTV is shown in Table 1. The second column gives the number of continuous variables and locations. The following columns give the duration of simulation trace (Real time), the number of sampled points (l), the actual time taken for the generation of the simulation trace (Sim time), the run time of HTV for computing the overapproximation of reach set (Reach time), the number of transitions, and the maximum error in $\{\epsilon_k\}$'s. We observe that HTV 's running time reduces with (1) smaller Lipschitz constants, (2) fewer transitions, and (3) for stable systems. Furthermore, the memory requirement for HTV grows only linearly with the number of sample points and the dimension of the system. This is because HTV stores the sampled trace and the polyhedral overapproximation of the reach set between a pair of consecutive sample points. These observations suggest that this combined formal analysis and simulation based approach may scale to larger problems.

5. REFERENCES

[1] Y. Annapureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *TACAS*, 2011.

[2] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In *HSCC*, volume 1790, pages 20–31, 2000.

[3] O. Bouissou and M. Martel. Grklib: a guaranteed runge kutta library. In *IMACS*, 2006.

[4] T. Dang and O. Maler. Reachability analysis via face lifting. In *HSCC*. Springer-Verlag, 1998.

[5] G. E. Fainekos and G. J. Pappas. Robustness of temporal logic specifications for continuous-time signals. *TCS*, 410:4262–4291, September 2009.

[6] A. Fehnker and F. Ivancic. Benchmarks for hybrid systems verification. In R. Alur and G. J. Pappas, editors, *HSCC*, volume 2993 of *LNCS*, pages 326–341. Springer, 2004.

[7] A. A. Julius, G. E. Fainekos, M. Anand, I. Lee, and G. J. Pappas. Robust test generation and coverage for hybrid systems. In A. Bemporad, A. Bichi, and G. Buttazzo, editors, *HSCC*, volume 4416 of *LNCS*, pages 329–342. Springer, 2007.

[8] A. Kanade, R. Alur, F. Ivancic, S. Ramesh, S. Sankaranarayanan, and K. Shashidhar. Generating and analyzing symbolic traces of simulink/stateflow models. In *CAV*, 2009.

[9] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager. *The Theory of Timed I/O Automata*. Synthesis Lectures on Computer Science. Morgan Claypool, November 2005. Also available as Technical Report MIT-LCS-TR-917.

[10] K.-D. Kim, S. Mitra, and P. R. Kumar. Bounded epsilon-reachability of linear hybrid automata with a deterministic and transversal discrete transition condition. In *CDC*, 2010.

[11] K. Manamcheri. Translation of simulink/stateflow models to hybrid automata. Master's thesis, University of Illinois at Urbana-Champaign, 2011.

[12] K. Manamcheri, S. Mitra, S. Bak, and M. Caccamo. A step towards verification and synthesis from simulink/stateflow models. In *HSCC*, 2011.

[13] S. Mitra. *A Verification Framework for Hybrid Systems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA 02139, September 2007.