STRONG: A Trajectory-Based Verification Toolbox for Hybrid Systems

Yi Deng[1], Akshay Rajhans[2], and A. Agung Julius[1]

[1] ECSE Department, Rensselaer Polytechnic Institute
[2] ECE Department, Carnegie Mellon University

Abstract. We present STRONG, a MATLAB toolbox for hybrid system verification. The toolbox addresses the problem of reachability/safety verification for bounded time. It simulates a finite number of trajectories and computes robust neighborhoods around their initial states such that any trajectory starting from these robust neighborhoods follows the same sequence of locations as the simulated trajectory does and avoids the unsafe set if the simulated trajectory does. Numerical simulation and computation of robust neighborhoods for linear dynamics scale well with the size of the problem. Moreover, the computation can be readily parallelized because the nominal trajectories can be simulated independently of each other. This paper showcases key features and functionalities of the toolbox using some examples.

1 Introduction

The problem of safety verification using reachability analysis, i.e., finding out whether the trajectories of a system reach a goal set and/or avoid an unsafe set, has received a lot of attention particularly in the hybrid systems community. The different approaches from the literature can be roughly classified into two types: state-space exploration techniques and construction of certificate-based guarantees. Despite recent progress, the applicability of these formal techniques still remains limited due the state-explosion problem and due to challenges in coming up with the right certificates necessary. On the other hand, in practice, simulation remains a widely-used approach for analyzing systems despite it being incomplete and informal. To bridge the divide between simulation and verification, tools that combine simulation with some formal analysis are recently being developed [3, 2]. In the similar spirit, we have been developing STRONG (System Testing using RObust Neighborhood Generation)[3], a Matlab toolbox for trajectory-based reachability/safety verification of hybrid systems.

Our approach combines simulation and formal verification. By simulating trajectories from a finite number of initial points within a compact set of initial conditions, we can obtain reachability and safety properties for the entire set of initial conditions [5].

[3] The toolbox and the supporting examples can be downloaded at `http://dengy3.myrpi.org/strong.html`. Preliminary work on the tool was done at University of Pennsylvania as a part of the masters thesis [6].

2 Features and functionalities

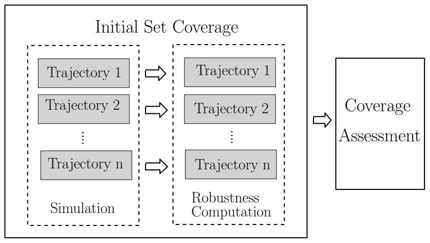
Model Consistency Checking. The toolbox has the ability to detect and correct certain kinds of ill-posedness in the model. It tracks the validity of state flow to detect common mistakes such as the reset state after a discrete transition falling out of the invariant of the new location. It can also detect a particular case of Zeno behaviors (infinite jumps in finite time between two neighboring locations), and correct the model by replacing such transitions with sliding modes.

Trajectory Simulation. Simulating a trajectory for a given initial condition is one of the main functionalities of the toolbox and forms a basis for the verification. The toolbox uses MATLAB's ode45 solver as a default for numerical integration. For every trajectory, the tool gathers all the information including the continuous evolution, transition events (e.g. unsafe), and event times. Trajectories of linear as well as nonlinear dynamics can be simulated.

Robustness Computation. The trajectory robustness and upper/lower time difference bounds between a simulated trajectory and its neighbors can be computed automatically for each continuous segment within every discrete location visited by the simulated trajectory. For linear dynamics, the computation involves solving a Lyapunov equation, for which we use standard convex optimization tools. Automatic verification of nonlinear systems is still under development. A trajectory and its robust ball of initial states can be visualized for any two specified dimensions.

Initial Set Coverage. Using the robustness analysis of a single trajectory, we can ascertain that the portion of the initial set covered by a robust ball around the chosen initial state leads to trajectories with the same safety and reachability properties as the simulated one. The final goal is to cover a given compact initial set as much as possible using simulated trajectories and their robust neighborhoods. Doing this in an effective way involves smartly choosing initial states for the simulated trajectories and assessing current coverage. Currently, the coverage strategy implemented is to generate random points as initial states, and an unbiased estimator [1] is used to evaluate the percentage of the covered initial set, which has a precision independent of the dimension of the state space.

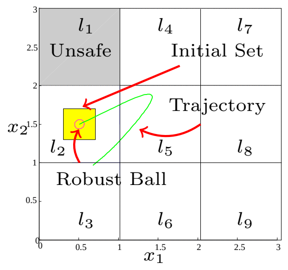
The procedure of property verification for an initial set is parallelized in the tool. As shown in the adjacent figure, trajectory simulations and robustness computation, which form the majority of the computation, can be performed independently and the initial set can be covered in a highly parallel manner.



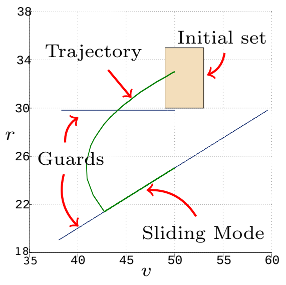
3 Examples

Demos of all examples can be found in the toolbox. Readers are recommended to view them for specific usage of commands. Here we present the examples briefly, and summarize the results as well as the performance of the tool in a table. For details, we refer the reader to the tool user guide available on the tool webpage.

Navigation Benchmark. Consider the navigation benchmark problem from [4, 5]. The system state vector x is comprised of position variables (x_1, x_2) and velocity variables (v_1, v_2). As shown in the plot, a simulated trajectory reaches four locations $(\ell_2, \ell_5, \ell_2, \ell_3)$ and no unsafe state ever reached. Any initial state within the robust ball leads to a safe trajectory that will reach $(\ell_2, \ell_5, \ell_2, \ell_3)$ with upper and lower bounds on each transition time.

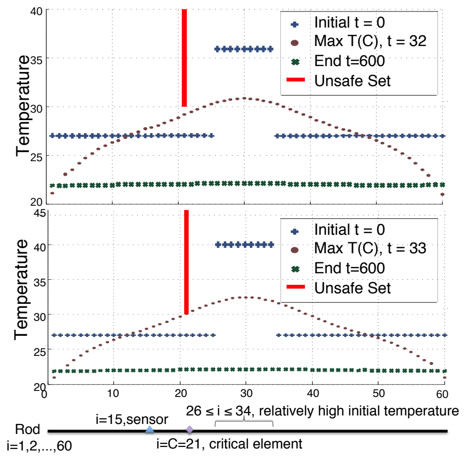


Automotive Cruise Control. In the automotive cruise control example from [7], v is the vehicle's velocity and r is the distance to another vehicle. The original dynamics has chattering, so we invoke our model consistency checking feature to automatically incorporate sliding modes. After that, normal verification can be performed. As shown in the plot, part of the simulated trajectory is along a guard, where we have inserted a location with sliding dynamics.



A High-dimensional Example. To demonstrate the scalability of our tool with the system dimension, we use a finite-element model to describe the heat-flow phenomenon along a rod as a 60th-order differential equation. A critical element is to be protected from being under- or over-heated ($\neg(10 \leq T(C) \leq 30)$) by injecting a hot/cold flux into the ends of the rod.

We simulate and verify the 60-dimensional system under two different initial conditions. For the first case shown in the adjacent figure, although several elements start from a relatively high temperature, safety can been maintained as the maximum $T(C)$, which occurs at time $t = 32$ is less than the unsafe threshold. On the other hand, in the second case depicted, the higher (worse) initial temperature of some elements results in unsafe temperature $T(C)$ at time $t = 33$. Properties of two different initial sets are verified as in the summary table.



problem	Navigation Benchmark				Cruise Control			Heat Flow			
end time	3				20			600			
dimension	4				2			60			
initial set	$0.3 \leq x_1 \leq 0.7$ $1.3 \leq x_2 \leq 1.7$ $-2.1 \leq v_1 \leq -1.9$ $-2.1 \leq v_2 \leq -1.9$			$0.3 \leq x_1 \leq 0.7$ $1.3 \leq x_2 \leq 1.7$ $1.9 \leq v_1 \leq 2.1$ $0.9 \leq v_2 \leq 1.1$	$49 \leq x_1 \leq 53$ $30 \leq x_2 \leq 35$			For $26 \leq i \leq 34$, $35.8 \leq x_i \leq 36.2$; otherwise, $x_i = 27$.			For $26 \leq i \leq 34$, $39.8 \leq x_i \leq 40.2$; otherwise, $x_i = 27$.
results[1]	t 0.7s	# traj. 1	cvg 100%	unsafe[2]	t 75.4s	# traj. 1500	cvg 100%	t 60.1s	# traj. 50	cvg 100%	unsafe[2]

[1] t = computation time on a 3.40 GHz Inter Xeon CPU, 16GB RAM, 4 cores; # traj. = number of trajectories tested; cvg = coverage assessment ($\pm 2\%$, $Pr > 99\%$).
[2] An unsafe trajectory is detected. The initial set cannot have uniform reachability and safety property.

To summarize, the STRONG toolbox is developed for bounded time reachability and safety verification of hybrid systems. Based on the idea of robust test generation and coverage, the tool computes a mathematically proven bound on the trajectory divergence and provides formal verification for the covered initial states. The tool does not use gridding; high-dimensional problems can be handled, and systems that are robustly safe can be verified with potentially very few trajectories. Further speed up can be achieved by using parallelization on multi-core machines. Directions for future work include supporting temporal logic specifications and handling stochastic system models.

Acknowledgments

YD and AAJ would like to acknowledge the support of NSF CAREER grant CNS-0953976. AR would like to acknowledge the support of NSF grants CNS-1035800 and CCF-0926181.

References

1. S. Afshari. Coverage assessment criteria for approximate bisimulation theory and introduction of computer games in hybrid systems safety/reachability design. Master's thesis, Rensselaer Polytechnic Institute, NY, 2010.
2. 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, 2011.
3. A. Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *CAV*, pages 167–170, 2010.
4. A. Fehnker and F. Ivancic. Benchmarks for hybrid systems verification. In *Hybrid Systems: Computation and Control*, pages 326–341. Springer, 2004.
5. 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*, pages 329–342. Springer, 2007.
6. A. Rajhans. Development of robust testing toolbox for hybrid systems. Master's thesis, School of Engineering and Applied Science, Univ. of Pennsylvania, 2007.
7. O. Stursberg, A. Fehnker, Z. Han, and B. H. Krogh. Verification of a cruise control system using counterexample-guided search. In *Control Engineering Practice*, volume 12, pages 1269–1278. October 2004.