Concolic Testing for Analysis of Hybrid Systems

CS477 Project Report

Umang Mathur Atul Sandur

November 9, 2015

1 Overview

Cyberphysical systems are engineered systems that are built from, and depend upon, the seamless integration of computational algorithms and physical components. These systems have made entered into every aspect of our life, and have become an indenspensible part of modern life. Indeed, the applications of these systems are widespread. Example include engineered pacemakers and robotic arms in healthcare, autopilots in passenger aircrafts and spacecrafts, controllers for chemical plants, self driving cars, railways systems, critical systems like nuclear reactors. Clearly the list is endless.

The framework of Hybrid automata—introduced by Alur, Courcourbetis, Henzinger, and Ho—provides a formal modeling and analysis environment to analyze the interaction between the discrete and the continuous parts of cyber-physical systems. Hybrid automata can be considered as generalizations of finite state automata augmented with a finite set of real-valued variables whose dynamics in each state is governed by a system of ordinary differential equations. Moreover, the discrete transitions of hybrid automata are guarded by constraints over the values of these real-valued variables, and enable discontinuous jumps in the evolution of these variables.

Given the immense integration of these systems in modern applicances, it is natural to ask questions such as whether a given system is free from erros. Concretely, if a given hybrid system reaches some *error* state, it can lead to loss of lives and property. For safety critical systems such as nuclear power plants, it becomes quite important to verify that the system does not crash.

As one can expect, there is a rich literature on verification of hybrid systems to tackle this problem. However, it turns out that the general model of Hybrid automata is quite expressive and not surprisingly, very simple questions like reachability are undecidable. Decidable subclasses for hybrid automata do exist, but most of them rely on a finite bisimulation to check for reachability, and do not scale very well.

As part of this project, we intend to develop systematic heurisitcs for answering questions such as reachability of some state in hybrid systems, using concepts from automated test input generation. The main advantage for using these techniques is scalability, that is, from the literature, these techniques look promising, and seem to scale for decently large systems. Another advantage is that such techniques also give information about concrete input values that lead the system to a target state

2 Hybrid Automata

The framework of Hybrid automata, introduced by Alur, Courcourbetis, Henzinger, and Ho [1], provides a formal modeling and analysis environment to analyze the interaction between the discrete and the continuous parts of cyber-physical systems. Hybrid automata can be considered as generalization of finite state automata augmented with a finite set of real-valued variables whose dynamics in each state is governed by a system of ordinary differential equations. Moreover, the discrete transitions of hybrid automata are guarded by constraints over the values of these real-valued variables, and enable discontinuous jumps in the evolution of these variables. In its full generality, the reachability question for Hybrid automata is undecidable. In fact the question remains undecidable for very simple subclasses of Hybrid automata, when the rate of evolution of variables are restricted to be constant integer values, and the constraints in the system are simple linear constraints.

3 Concolic Testing

Concolic testing [3] automates test input generation by combining concrete and symbolic executions of the system under test. So the concrete part constitutes normal execution of the program while symbolic part

collects symbolic constraints over the symbolic input values at each branch point encountered along concrete execution path. At the end of an execution, we have collected a set of path constraints that exercise an execution path in the system. For the next iteration, one of the constraints is negated and solved using constraint solvers to generate a new test input that directs the system along a different execution path. This process continues until all feasible distinct execution paths have been explored. When some of the symbolic constraints become too complicated for the constraint solver to generate concrete values, they are simplified by replacing some of the symbolic values with concrete values. Concolic testing is thus sound (infers real bugs) and also allows exploring previously unreachable paths in the system.

4 Approach

To begin with, we will model embedded systems using Singular Hybrid Automata (SHA), the reachability question for which is known to be undecidable even for 3 vartiables [2]. SHA can be described as a set of discrete states and a set of continuous variables. Additionally there is a function $f(s, x)$ which is a constant integer for every discrete state s and continuous variable x, that denotes the rate of evolution of x in state s. Moreover every state s and discrete transition between any two such states are annotated using a set of linear constraints. The constraints are referred to as *invariants* and *guards* respectively. A sequence of pairs of the form (s,t) where s is a discrete state and t is positive constant denoting time spent in that state, describes a valid *trace* if all guards and invariants are met throughout the sequence. We say that a state q is reachable, if there is some initial value for the continuous variables and a finite trace $(s_0, t_0), (s_1, t_1), \ldots (s_k, t_k)$ exists such that $s_k = q$.

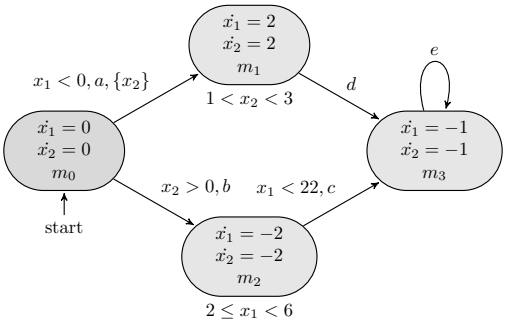

Figure 1: Singular Hybrid Automata with four modes and two variables.

Given initial values of continuous variables and a concrete trace, it is easy to check if it is a valid trace that reaches some state q. However coming up with concrete values for time durations in that trace is non-trivial because the possible values of t_i's are uncountable. Therefore it makes sense to look at symbolic traces where t_i's are symbolic variables.

A very basic idea to solve reachability question using test generation concepts, is to collect constraints on invariants/guards in a given symbolic trace and feed it to a suitable solver to check for feasibility of the constraints. This will give concrete initial values for the input to the system. It is possible to avoid quantifiers from the constraints because the continuous variables evolve in a linear fashion relative to each other. Therefore we expect simple SMT solvers like Yices to be able to handle these constraints. The challenge is in determining correct symbolic traces that lead to a final goal state in hybrid sytems which have loops. This is because one can generate infinite number of symbolic traces as inputs to our algorithm. We will initially focus our attention on bounded step reachability. Another challenging aspect is to be able to deal with non-linear invariants/guards as constraints that most solvers cannot handle. We attempt to attack this problem by guiding symbolic executions using concrete values for symbolic variables. This is primarily where we will use concepts from concolic testing techniques. Further, such non-linear constraints result in quantified constraints which have one or more quanitified variables. This is again difficult for constraint solvers to handle. For hybrid systems where continuous variables evolve in a much complex fashion, such as when the rates of evolution are specified in terms of arbitrary differential equations, there are many more challenges that need to be addressed. We will be looking at such problems as part of future work.

References

[1] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-S. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, pages 209–229, 1992.

[2] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? *Journal of Comp. and Sys. Sciences*, 57:94–124, 1998.

[3] Koushik Sen. Concolic testing. In *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*, ASE '07, pages 571–572, New York, NY, USA, 2007. ACM.