AVERIST: Algorithmic Verifier of Stability

Miriam García Soto*
IMDEA Software Institute & Universidad Politécnica de Madrid
Pozuelo de Alarcón, Madrid, Spain
miriam.garcia@imdea.org

Pavithra Prabhakar†
Kansas State University
Manhattan, KS, USA
pprabhakar@ksu.edu

Abstract

AVERIST[9] is a software tool which implements an algorithmic approach to verify stability of linear hybrid systems [6, 8]. In particular, it analyzes stability of linear switched systems. We illustrate the AVERIST performance through four easy examples, two polyhedral switched systems and two linear switched systems, where we explore stability, instability, arbitrary switching and state based switching.

CCS Concepts •**Theory of computation →Abstraction;** *Logic and verification;* •**Computer systems organization →Embedded and cyber-physical systems;**

Keywords Hybrid systems, Stability verification

ACM Reference format:
Miriam García Soto and Pavithra Prabhakar. 2017. AVERIST: Algorithmic Verifier of Stability. In *Proceedings of 20th ACM International Conference on Hybrid Systems: Computation and Control, Pittsburgh, PA, USA, April 18–21, 2017 (HSCC '17),* 2 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 Introduction

Hybrid systems are dynamical systems with a mixed discrete and continuous behaviour. Hybrid automaton is a formalism which captures such mixed evolution. They are useful to model modern control systems, as autonomous vehicles and smart grids. A fundamental property expected out of any control system is stability. The stability property assures that small perturbations in the input to the systems just result in small perturbations of the eventual behavior of the system. The state of the art for stability verification relies on deductive methods. We propose an algorithmic approach.

2 Architecture and design

AVERIST implements a new counterexample guided abstraction refinement (CEGAR) framework for analyzing the hybrid systems with polyhedral inclusion dynamics that are generated as a result of the hybridization. The tool performs the following main functions:

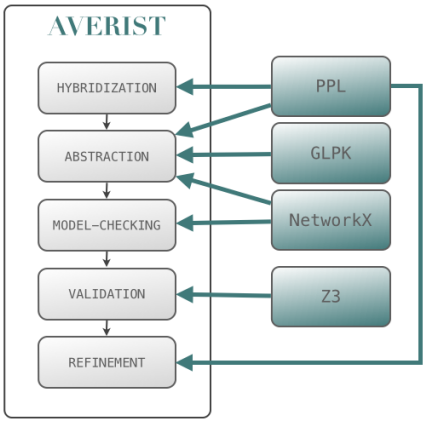


Figure 1. AVERIST architecture

Hybridization. This function essentially constructs a hybrid system with polyhedral inclusion dynamics (PHS) from a linear hybrid system (LHS). This is achieved by means of a state-space partition and a linear dynamics over-approximation. The state-space partition is constructed using a set of linear predicates; and the over-approximation consists of a polyhedral set which collects the vector field of the LHS restricted to a region (see [7] for details).

Quantitative Predicate Abstraction. This function takes as input a polyhedral hybrid system and outputs a finite weighted graph that over-approximates the behavior of the PHS. The nodes in the graph correspond to certain facets of the boundaries of the regions in a state-space partition, and the edges correspond to the existence of an execution between the facets corresponding to the nodes. In addition, each edge is tagged with a weight that provides an upper bound on the scaling factor associated with the executions between the facets.

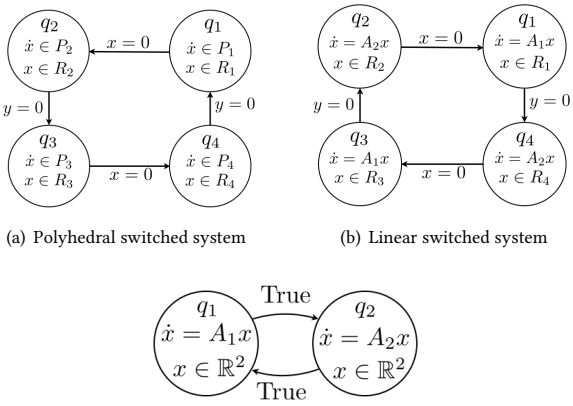
Model-checking. This function takes a weighted graph corresponding to a quantitative predicate abstraction (QPA) as input and checks structural conditions corresponding to the existence of certain kinds of cycles to either deduce stability, or output a counterexample showing a potential reason for instability. In particular, if there is no cycle in the weighted graph with weight greater than one, then the initial hybrid system is stable. On the contrary, a cycle with weight greater than one represents an abstract counterexample.

Validation. This function takes as input an abstract counterexample from the weighted graph analysis of a QPA, and checks if it corresponds to an actual execution that exhibits instability. In particular, the analysis determines if the abstract counterexample is spurious or on the contrary corresponds to an infinite divergent execution in the concrete hybrid system.

*Miriam García Soto is partially supported by BES-2013-065076 grant from the Spanish Ministry of Economy and Competitiveness.
†Pavithra Prabhakar is partially supported by EU FP7 Marie Curie Career Integration Grant no. 631622 and NSF CAREER award no. 1552668.

HSCC '17, Pittsburgh, PA, USA

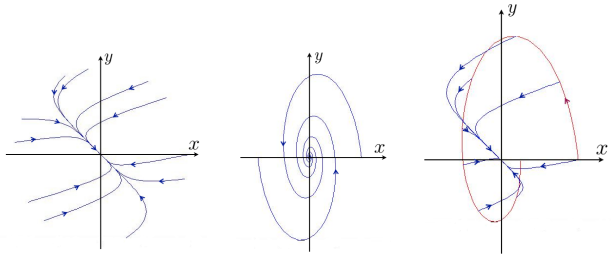
DOI: 10.1145/nnnnnnn.nnnnnnn



(a) Polyhedral switched system (b) Linear switched system

(c) Linear arbitrary switched system

Figure 2. Hybrid automata



(a) $\dot{x} = A_1x$ (b) $\dot{x} = A_2x$ (c) Arbitrary switching

Figure 3. Linear systems

Refinement. This function takes as input an abstract counterexample of a QPA that has been determined spurious in the validation module and identifies predicates to be added to the state-space partition of the QPA which will eliminate the abstract counterexample.

The software architecture is shown in Figure 1. The Parma Polyhedra Library (PPL) [4] is used to manipulate polyhedral sets, a reachability analyzer to determine the existence of edges and the GNU Linear Programming Kit GLPK [1] solver to compute the weights associated with such edges. All these steps are oriented to construct the abstract weighted graph. The graph is constructed and model-checked by using the NetworkX Python package [2]. The Z3 software [5] is a high-performance theorem prover and it is used to check for satisfiability of formulas encoding the existence of a concrete counterexample for stability. The PPL and NetworkX packages utilities are included in the free open-source mathematics software system sage [3] and the tool is run in it through a terminal session.

3 Demo

We will show AVERIST performance on two polyhedral switched systems, one stable and the other one unstable, and on two linear switched systems, one with arbitrary switching and the other one with state based switching. These examples are defined as hybrid automata. We will show how to instantiate them by using a mark-up language. The instances of the systems will be processed by AVERIST, which will generate output files containing relevant information about the run of the tool and the stability analysis. In addition to the hybrid automaton input, certain parameters need to be set. The role of input parameters in the process will be explained. The obtained output will include detailed information for each CEGAR iteration of the full procedure.

Experiment 1. It will consider a 3-dimensional stable switched system with constant dynamics, as shown in Figure2(a). This system will illustrate a stability proof after three CEGAR iterations, spurious counterexamples and predicate refinement.

Experiment 2. It will consider a 3-dimensional unstable switched system with constant dynamics, as shown in Figure2(a). This instance will illustrate an instability proof by validating the output abstract counterexample. The validation is performed by creating a formula and asking about satisfiability to the Z3 theorem prover. We will show the satisfiability formula.

Experiment 3. It will consider a 2-dimensional stable system with two arbitrary dynamics, as shown in Figure2(c). Sample executions of the linear dynamics are depicted in Figures 3(a) and 3(b). This system can switch between the two dynamics at any state. Sample executions of the arbitrary switching dynamics are shown in Figure 3(c). This experiment will be used to illustrate different ways of choosing the initial predicates for state-space partition in the hybridization and QPA procedures. We will choose predicates manually by creating an input file to determine them, and we will choose automatically by providing certain input parameters.

Experiment 4. It will consider a 2-dimensional stable switched system as shown in Figure 2(b). We will illustrate hybridization on it, by showing the over-approximated polyhedral switched system.

4 Conclusion

The proposed demo shows an automatic approach for stability verification of hybrid systems. Input data can be easily defined by people with no experience in hybrid systems and running AVERIST does not require a formal knowledge on control systems and stability analysis.

References

[1] Glpk: https://www.gnu.org/software/glpk/.
[2] NetworkX: https://networkx.github.io/.
[3] sage: http://www.sagemath.org/.
[4] R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.
[5] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. pages 337–340, Berlin, Heidelberg, 2008. Springer-Verlag.
[6] Pavithra Prabhakar and Miriam García Soto. Hybridization for stability analysis of switched linear systems. HSCC '16, pages 71–80, New York, NY, USA, 2016. ACM.
[7] Pavithra Prabhakar and Miriam García Soto. Hybridization for stability analysis of switched linear systems. HSCC '16, pages 71–80, 2016.
[8] Pavithra Prabhakar and Miriam García Soto. An algorithmic approach to stability verification of polyhedral switched systems. In *ACC 2014*.
[9] Pavithra Prabhakar and Miriam García Soto. AVERIST: Algorithmic verifier of stability. In *NSV Workshop 2015*.