
Interactive Verification of Hybrid Systems

Xian Li, Klaus Schneider

2 pages
Interactive Verification of Hybrid Systems

Xian Li, Klaus Schneider

Embedded Systems Group
Department of Computer Science
University of Kaiserslautern, Germany
Email: {xian.li,klaus.schneider}@cs.uni-kl.de

Abstract: We present our approach towards the interactive verification of hybrid systems which takes advantage of combining human intelligence with existing decision procedures and tools. In particular, we target the parametric verification problem of hybrid systems, where values of parameters of the system have to be determined to meet the given specifications. To this end, we allow linear and non-linear arithmetic with real, integer, and boolean variables. A prototypical interactive theorem prover AIFProver is developed as an interactive verification environment providing an effective interface between designers and backend verification techniques and tools for hybrid systems.

Keywords: Interactive Verification, Parametric Verification, Hybrid Systems

The parametric verification problem of hybrid systems consists in deriving constraints on the values of parameters to meet the given specifications. Numerous approaches and tools targeted for hybrid system verification have been proposed in the literature for verifying either concrete systems with given parameter instantiations, or a subset of hybrid systems in the theory of reals only. So, the parametric verification problem has not yet found good solutions. We work on an interactive verification approach combining human intelligence with the existing verification methods for parametric hybrid systems, where we allow linear and non-linear arithmetic with real, integer, and boolean variables. To support the proposed interactive verification approach, our Averest system (http://www.averest.org), a toolset for HW/SW co-design and formal verification, has been extended by AIFProver, a prototypical interactive theorem prover for discrete and hybrid systems [GS13, LBS13].

The idea of interactive verification of parametric hybrid programs is illustrated in Fig. 1. The Averest compiler computes for a given hybrid program an equivalent set of guarded actions [BS10]. Proof goals are given as pairs \((G, L)\) that contain a set of guarded actions \(G\) and the set of currently active control-flow variables \(L\) [GS13]. The AIFProver consists of proof rules that split these proof goals into subgoals or simply solve these proof goals directly. The proposed approach allows us not only for establishing interoperability among a wide range of different modeling formalisms and tools, but also for exploring a verification approach that takes advantage of the verification engineers.

The basic proof rules have been implemented in AIFProver as F# functions so that these can be applied either in an interactive F# session or in a proof script (i.e., a F# script). In addition to the integrated SAT solver and a BDD package, there are interfaces to SAL [dOR+04] for verification of discrete systems, and to KeYmaera [Pla10] for verification of continuous systems. Also, we explore a new method by integrating the SAT solver with BONMIN [BBC+08] to solve a special
Interactive Verification of Hybrid Systems

SMT problem addressing boolean combinations of propositional-logic atoms and theory atoms from undecidable non-linear arithmetic theories over integers and reals.

In the interactive verification approach, our hybrid modelling language serves as a bridge between different verification techniques and tools. Therefore, we are evaluating other SMT solvers and mathematical computation tools to extend the backend tool interface. Meanwhile, we are evaluating and extending the set of basic proof rules by benchmark examples. Except for the tool supported techniques, we plan to integrate non-tool supported techniques for the interactive verification approach as well. We are exploring and implementing abstraction and counterexample guided abstraction refinement (CEGAR) techniques to obtain some practically relevant results. Also, we consider the simulation-based verification method. Simulating some system instances explicitly could give some useful hints for for the interactive verification itself.

Bibliography


