Demo: Hybrid Systems Model Transformations with HyST*

Stanley Bak
Air Force Research Laboratory
Information Directorate, USA

Sergiy Bogomolov
IST Austria

Taylor T. Johnson
University of Texas at
Arlington, USA

ABSTRACT

Algorithmically analyzing hybrid systems models is challenging in theory and in practice. Numerous sound (and sometimes complete) transformations for simplifying the analysis of hybrid systems models have been developed, and are used to show both theoretical results such as reductions to finite-state automata for certain classes and practical results to ease reachability analysis. HyST is a software framework for implementing transformation passes for hybrid automata, and supports various transformation passes, including hybridization (which simplifies continuous dynamics), continuization (which simplifies discrete dynamics), pseudo-invariants (which adds auxiliary invariants that do not change the reachable states, but ease reachability analysis computations), order-reduction (which reduces the number of state variables [dimensionality]), among others. This demonstration will illustrate these transformations in HyST on canonical hybrid systems examples, and show analysis results with a number of state-of-the-art hybrid systems verification tools such as SpaceEx, Flow*, dReach, and HyComp.

1. INTRODUCTION

A hybrid automaton [1] is an expressive mathematical model useful for describing complex dynamic processes involving both continuous and discrete states and their evolution. Software tools for algorithmically analyzing various classes of hybrid automata have been developed, and recent tools include SpaceEx for affine dynamics [6–8, 11], Flow* for nonlinear dynamics [9], dReach for nonlinear dynamics [12], and HyComp [10] for polynomial dynamics. HyST is a source transformation and translation tool for hybrid automaton models [4]. HyST supports source-to-source model transformation passes in an intermediate representation, which is represented as networks of hybrid automata. The input to HyST is a network of hybrid automata in the SpaceEx format, and the output is a new network of hybrid automata in the various input formats supported by different tools (currently SpaceEx, Flow*, dReach, and HyComp). In addition to syntactic conversions, several recent transformation passes in HyST are useful for simplifying analyses, and its architecture makes these transformations applicable across numerous tools. Compared to the original version presented as a tool paper [4], HyST now includes support for additional model transformation passes, networks of hybrid automata, and additional output formats (HyComp). This demonstration will illustrate HyST's usage to interface tools and the recent transformation passes.

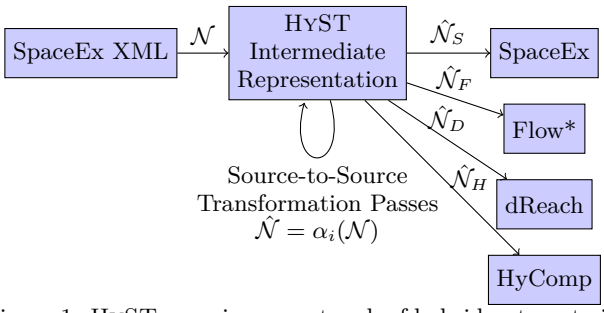

Figure 1: HyST overview: a network of hybrid automata in the SpaceEx format $\mathcal{N}$ is parsed into the intermediate representation, on to which model transformation passes α_i are applied. Various syntactic transformations are also applied before exporting to the supported tools.

2. DEMONSTRATING TRANSFORMATION PASSES WITH HYST

This demonstration of HyST will illustrate its currently supported use cases, with a particular focus on the recently added transformation passes.[1] We note that all of the model transformations are sound or overapproximative, in the sense that the resulting transformed automaton's reachable states contain those of the original automaton. Figure 1 shows the high-level architecture of HyST. The demonstration will consist of showing how to apply passes to hybrid automata, modify examples, and use scripts to automatically execute the supported tools.

2.1 Hybridization

HyST implements a hybridization source-to-source transformation, which creates a simpler hybrid automaton from a more complex one, for example, by overapproximating nonlinear differential equations as linear differential inclusions [3]. Most modern hybridization techniques rely on dynamic (or on-the-fly) hybridization which helps to avoid the costly partitioning of the state-space as convex cells, which if done statically by creating a new hybrid automaton to analyze frequently leads to an exponential blow-up in the dimensionality of the system. However, HyST's source-to-source (i.e., static) hybridization methods exploit benefits of dynamic hybridization methods by guiding the static partitioning through offline simulations, and additionally using time-triggered transitions in addition to state-dependent transitions between partitions of the state-space [3].[2]

*DISTRIBUTION A. Approved for public release; Distribution unlimited. (Approval AFRL PA #88ABW-2016-0181, 28 JAN 2016)

[1] HyST is available online: http://verivital.com/hyst/
[2] http://verivital.com/hyst/pass-hybridization/

2.2 Continuization

A challenge in analyzing hybrid automata with time-dependent switching is that frequently occurring transitions can cause a blow-up in the number of intersection operations needed, which may lead to a blow-up in the overapproximation error for such systems. For some classes of periodically-switched hybrid automata that are reasonable models of popular real-time schedulers (such as rate monotonic scheduling [RMS] and earliest-deadline first [EDF]), HyST implements continuization to avoid these explosions of numbers of transitions and intersection operations [5]. Somewhat as the converse of hybridization, which may take a purely continuous nonlinear system and from it create a hybrid automaton with simpler (e.g., linear) dynamics, continuization takes a hybrid automaton and overapproximates its behavior with a purely continuous system by overapproximating the switching behavior as nondeterministic additive terms.

2.3 Pseudo-Invariants

The pseudo-invariants transformation passes introduces auxiliary invariants in modes of the hybrid automaton, such that these pseudo-invariants do not change the set of reachable states after the transformation. While the reachable states do not change, the reason for adding such pseudo-invariants is for reducing overapproximation error in the reachability algorithms, which often can exploit such additional invariants to reduce the set of computed reachable states [2].

2.4 Order-Reduction

Order-reduction is a common approach in systems and control to simplifying analysis of systems, and roughly creates a reduced-order system with fewer state variables (decreased dimensionality) such that the reduced-order system has behaviors similar to those of the original, or full-order, system [17]. To be used as a sound abstraction for verification, key arguments must be made with respect to the similarity of behaviors between the original and reduced-order system, and approaches relying on approximate bisimulation relations [13] and deriving error bounds from numerical simulations [14] have been explored. HyST implements order-reduction methods for linear systems based on balanced-truncation, which have allowed us to verify safety of systems with up to a thousand state variables (dimensions) [17]. The implementation of these order-reduction methods relies on a bridge between HyST and Matlab, and allows us to use built-in order-reduction methods in Matlab, as well as derive error bound overapproximations.[3]

3. BENCHMARKS

HyST has been evaluated and comes with a wide range of benchmarks of various classes of hybrid automata, including: timed automata, rectangular hybrid automata, hybrid automata with linear/affine differential equations, and nonlinear hybrid automata. Several benchmark packages in part leveraging HyST have been released, and all the models are in the SpaceEx XML format. The benchmarks released include: DC-to-DC power electronics converters [15], nonlinear systems frequently used as benchmarks in the numerical analysis community [16], and larger-scale linear systems frequently used in controls and order-reduction [17].

[3] http://verivital.com/hyst/pass-order-reduction/

4. CONCLUSION

Overall, this demonstration will illustrate the current features in HyST, from model transformation passes to integration with state-of-the-art hybrid systems verification tools. In the future, we hope to continue to engage with the community and integrate additional tools and new transformation passes within HyST.

5. REFERENCES

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[2] S. Bak. Reducing the wrapping effect in flowpipe construction using pseudo-invariants. In *4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*, CyPhy '14, pages 40–43. ACM, 2014.

[3] S. Bak, S. Bogomolov, T. A. Henzinger, T. T. Johnson, and P. Prakash. Scalable static hybridization methods for analysis of nonlinear systems. In *Proc. of the 19th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC)*. ACM, 2016.

[4] S. Bak, S. Bogomolov, and T. T. Johnson. HyST: A source transformation and translation tool for hybrid automaton models. In *Proc. of the 18th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC)*. ACM, 2015.

[5] S. Bak and T. T. Johnson. Periodically-scheduled controller analysis using hybrid systems reachability and continuization. In *36th IEEE Real-Time Systems Symposium (RTSS)*, San Antonio, Texas, Dec. 2015. IEEE Computer Society.

[6] S. Bogomolov, A. Donzé, G. Frehse, R. Grosu, T. T. Johnson, H. Ladan, A. Podelski, and M. Wehrle. Abstraction-based guided search for hybrid systems. In *International SPIN Symposium on Model Checking of Software 2013*, LNCS. Springer, 2013.

[7] S. Bogomolov, A. Donzé, G. Frehse, R. Grosu, T. T. Johnson, H. Ladan, A. Podelski, and M. Wehrle. Guided search for hybrid systems based on coarse-grained space abstractions. *International Journal on Software Tools for Technology Transfer*, pages 1–19, 2015.

[8] S. Bogomolov, G. Frehse, M. Greitschus, R. Grosu, C. S. Pasareanu, A. Podelski, and T. Strump. Assume-guarantee abstraction refinement meets hybrid systems. In *10th International Haifa Verification Conference (HVC 2014)*, volume 8855 of *LNCS*, pages 116–131. Springer, 2014.

[9] X. Chen, E. Abraham, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In N. Sharygina and H. Veith, editors, *Computer Aided Verification*, volume 8044 of *LNCS*, pages 258–263. Springer Berlin Heidelberg, 2013.

[10] A. Cimatti, A. Griggio, S. Mover, and S. Tonetta. HyComp: An SMT-based model checker for hybrid systems. In C. Baier and C. Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 9035 of *Lecture Notes in Computer Science*, pages 52–67. Springer Berlin Heidelberg, 2015.

[11] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable verification of hybrid systems. In *Computer Aided Verification (CAV)*, LNCS. Springer, 2011.

[12] S. Gao, S. Kong, and E. Clarke. Satisfiability modulo ODEs. In *International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, Oct. 2013.

[13] A. Girard and G. J. Pappas. Approximate bisimulation relations for constrained linear systems. *Automatica*, 43(8):1307 – 1317, 2007.

[14] Z. Han and B. Krogh. Reachability analysis of hybrid control systems using reduced-order models. In *American Control Conference*, volume 2, pages 1183–1189, 2004.

[15] L. V. Nguyen and T. T. Johnson. Benchmark: Dc-to-dc switched-mode power converters (buck converters, boost converters, and buck-boost converters). In *Applied Verification for Continuous and Hybrid Systems Workshop (ARCH 2014)*, Berlin, Germany, Apr. 2014.

[16] H.-D. Tran, L. V. Nguyen, and T. T. Johnson. Benchmark: A nonlinear reachability analysis test set from numerical analysis. In *Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Seattle, Washington, Apr. 2015.

[17] H.-D. Tran, L. Viet Nguyen, W. Xiang, and T. T. Johnson. Order-Reduction Abstractions for Safety Verification of High-Dimensional Linear Systems. *ArXiv e-prints*, Feb. 2016.