HYST: A Source-to-Source Transformation Framework for Hybrid Automata

Stanley Bak[1], Sergiy Bogomolov[2,3], and Taylor T. Johnson[4]

[1] Air Force Research Laboratory Rome, NY, USA
[2] IST Austria, Austria
[3] University of Freiburg, Germany
[4] University of Texas at Arlington Arlington, TX, USA

Abstract

A number of powerful hybrid systems model checkers have recently emerged. Those tools share a common goal of establishing an automatic verification process of hybrid systems, yet use quite different approaches. The two most prominent approaches are flow-pipe construction and decision procedures. Although both classes of approaches show promise, they have unique strengths and weaknesses. Therefore, it is natural to investigate the possible synergies of those tools. Unfortunately, although all the tools support roughly the same hybrid systems semantics, their model description languages differ dramatically. This makes it difficult to quickly evaluate a specific hybrid automaton model using the different tools and, even more importantly, makes the application cost of multiple tools in a tool chain prohibitively high due to the low level of inter-tool interaction possibilities. In this paper, we present HYST, a *source-to-source translation tool*, currently taking input in the SpaceEx model format, and translating to the formats of HyCreate, Flow*, or dReach. Besides providing a translation between different model description languages, our tool supports the concept of model-to-model *transformation passes* that both ease the translation and potentially improve reachability results for the supported tools. In addition, the transformation passes provide a natural way to embed multiple tools into a single *tool chain*. We envision HYST being a research vehicle which will help to bootstrap novel approaches on the interface of multiple tools and approaches. Our evaluation demonstrates HYST is capable of automatically translating benchmarks in several classes (including affine and nonlinear hybrid automata) to the input formats of several tools. Additionally, we illustrate reachability improvement with a general model transformation pass based on pseudo-invariants, currently implemented in HYST.

1 Introduction

Hybrid systems are mathematical models that combine discrete and continuous dynamics. This formalism has a large expressive power and therefore can describe the behaviour of a large range of real-world systems, e.g., from the domain of embedded systems [19] and biological systems [10, 12]. A number of powerful and scalable model checkers have recently emerged [3, 8, 13, 14, 18]. They cover a number of hybrid system classes, e.g., affine vs. non-linear continuous dynamics and monolithic vs. automata networks. Furthermore, the analysis algorithms are built around different ideas and state representations, such as flow-pipe construction or decision procedures for differential equations. These design decisions make the tools particularly efficient in some settings, such as only for some classes of continuous dynamics. This makes the analysis of complex heterogeneous systems comprising of *multiple* components particularly challenging as the tool suitable for the analysis of the *most complex* component will be applied to the *whole* system. In this paper, we lay the foundations for an *inter-tool* analysis framework which relies

on the power of multiple approaches organised in a single tool chain to attack the system complexity. In particular, we present an automatic source-to-source model converter from the SpaceEx input format to the Flow*, HyCreate, and dReach formats. At present, direct comparisons between model checkers cannot be done out-of-the-box as the input languages are syntactically different. However, a manual comparison is possible because, although the input languages of the considered model checkers differ *syntactically*, they rely on the same behavioural *semantics*. In this way, a user of verification tools can quickly generate a model file for a number of tools in order to find a tool that best fits the system under consideration. Furthermore, a developer of hybrid systems model checkers can use HYST to both compare the performance of newly developed algorithms with other up-to-date analysis tools, as well as to quickly check for correctness against a common set of models and as part of a regression test suite. Finally, HYST supports the concept of model-to-model *transformation passes* which provide a way to adjust hybrid models towards the applied system analysis methodology. In the paper, we discuss a model transformation pass based on pseudo-invariants to show possible reachability analysis improvements. In the long term, we envision the Hyst being a natural way to connect multiple tools in a tool chain and in this way providing an efficient way to exploit possible collaborative verification among different tools.

This report is an extended abstract of an earlier paper presented recently at HSCC 2015 [5].

Related Work. In the last decade, several research groups have worked on approaches to unify the syntax of hybrid model checkers. Sangiovanni-Vincentelli et al suggest the hybrid systems interchange format (HSIF) [6]. A further attempt to provide a common input language focused on the model composition has been undertaken within the FP7 Multiform project [23]. The project resulted in the Compositional Interchange Format (CIF). Earlier efforts for interchange formats were initiated for Charon [2]. The above outlined projects have in common an idea to collect all the features available in different hybrid model checkers and provide an input language which essentially subsumes the languages of every particular tool. Although having a common interchange language supported by all the tools would be an ultimate solution, this approach hinges on the willingness of the tool developers to support such an input format. Furthermore, the incorporation of a common format into an established tool by third party developers would be difficult due to the time overhead needed to get acquainted with the code of each particular hybrid model checker.

Alternative approaches include using other frequently-used languages as standard input formats. Agrawal et al. [1] suggest an algorithm to translate Simulink and Stateflow models (SSM) into the equivalent HSIF models. In a slightly different setting, Schrammel et al. [22] consider the translation problem for complex SSMs where involved treatment of zero-crossings (enabling conditions for guards) is needed. Chen et al. [7] provide a translation from Stateflow to CSP [17]. Mathworks, unfortunately, does not provide any rigid operational semantics for its tools. This makes the model translation process error-prone and ambiguous, whereas we are mostly concerned with the formal verification of a given model. Other recent languages include the HYbrid systems with Discrete Interaction (HyDI) language, which is an extension of the SMV input language [9]. Recent converter initiatives include a converter from Ptolemy II to SpaceEx [21], and the HyLink converter from SSMs to hybrid automata [20].

2 Converter Architecture

The conversion architecture used in HYST is shown in Figure 1. The tools takes as input a source file, parses it to an intermediate representation (IR), then prints the resulting output source format desired. Using the SpaceEx format has a number of practical advantages: several

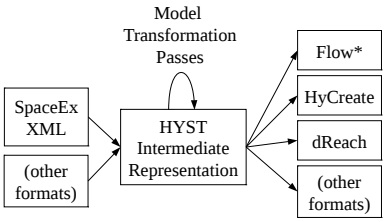


Figure 1: HYST conversion architecture. SpaceEx XML models specifying hybrid automata are translated to an intermediate representation (IR) inside HYST, where model transformation passes are performed. Then, the hybrid automaton is output in different syntax for input to other tools, including HyCreate, Flow*, and dReach. It is not shown, but the SpaceEx configuration file is also used, where, e.g., initial states and potentially bad states are specified.

available examples already exist in the format, there is a visual model editor for these files, SpaceEx can import from the CIF format [15] and output SpaceEx XML, and there is preliminary support in SpaceEx for hybrid automata flattening. As an input format, the grammar specifying flows, guards, invariants, and resets of the automata does not have any restrictions on being affine, so we use the input format of SpaceEx for HYST and allow for nonlinear expressions. We note that in its current version, however, SpaceEx cannot analyze nonlinear examples as the algorithms it uses do require affine expressions. In terms of output formats, the supported tools are Flow*, HyCreate (both reachability and simulation), and dReach, and output to other tools is part of our planned future work.

Internally, the IR is currently a set of data structures in Java which encode the modes, transitions, continuous variables, flow differential equations, guards, and invariants. The intermediate representation may be modified prior to output for a specific tool's format through model transformation passes. Passes can be viewed as model-to-model conversions. Some passes can aid in the exporting process. For example, dReach requires that identity resets are explicitly defined, whereas other tools do not. A model transformation pass is therefore run before writing a dReach source file which adds identity resets to transitions which did not define them. Another model transformation pass can be used to check and rename variables that are disallowed keywords for specific tools. Finally, model transformation passes can be used to modify the reachability computation itself. For example, the method of pseudo-invariants [4] has previously been shown to improve the accuracy and speed of the reachability computation for multiple tools. Implementing it as a model transformation pass allows all supported tools to be able to use the technique. An example using this pass is shown later in Section 3. Other candidate passes we plan on implementing include over-approximating abstraction techniques, such as hybridization. Once implemented in HYST, these techniques would not need to be reimplemented for each tool to use the approach, saving implementation effort and reducing the likelihood of mistakes.

3 Results

In the introduction, three classes of users were considered for HYST: (1) Users of verification tools, (2) developers of tools, and (3) researchers who develop general techniques that may be applicable to a wide variety of tools.

Users of Verification Tools Users can use HYST as a quick way to create workable model files for all of the supported tools. We used HYST to convert numerous hybrid system models

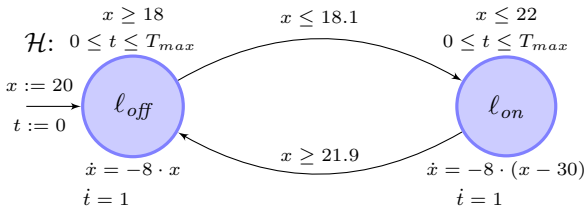
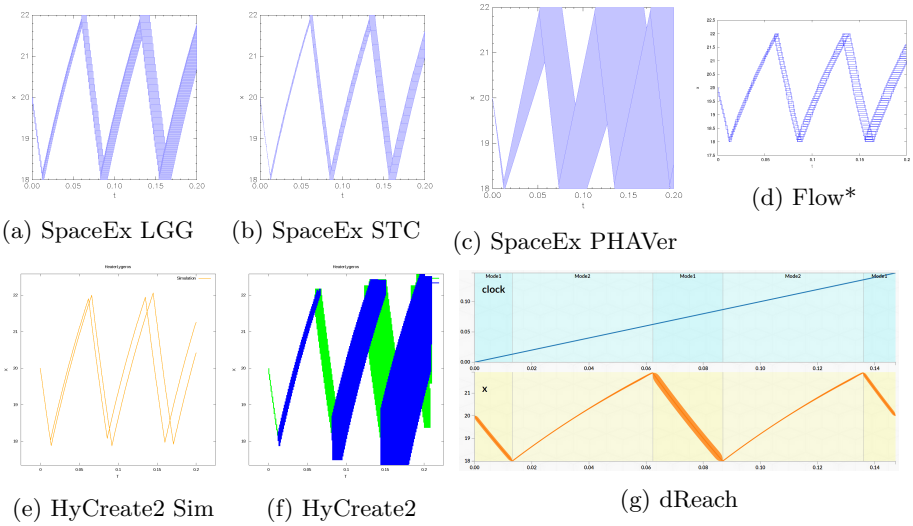


Figure 2: Thermostat/heater example hybrid automaton.



(a) SpaceEx LGG (b) SpaceEx STC (c) SpaceEx PHAVer (d) Flow*

(e) HyCreate2 Sim (f) HyCreate2 (g) dReach

Figure 3: Default reach set visualizations for thermostat/heater hybrid automata example from Figure 2 for the various output model files and tools.

from different classes, including typical affine and nonlinear examples. For the purposes of illustration, we use the example of a heater system interacting with the temperature in a room measured by a thermostat, as shown in Figure 2 [16]. Although a simple system, it is illustrative of most of the features found in hybrid automata including invariants, guards, flows, and non-determinism. More complicated benchmarks have been converted using HYST, however this system is sufficient for reach-set illustration.

This system was converted by HYST into the input format of the various tools. Each tool was then run on the files, producing a reach set as the output. The graphical results for each of the tools are shown in Figure 3. For all tools except dReach, these figures correspond to reachable states. For dReach, the image corresponds to a witness counterexample execution that leads to a bad (goal) set of states. These models serve as initial starting points for users who want to analyze a model, as they can immediately see the rough performance of the various tools. As described earlier, the tools contain tool-specific parameters which, after generation using HYST, can be further tweaked by the user.

We currently have about a dozen benchmarks which can be run through the translator and executed by the various tools. These examples range from biological systems, neuron models, power converters, and typical systems for evaluations of nonlinear reachability methods. The

full set of examples and their converted models are available on the HYST website[1]. The conversion process for each model takes less than a second and is negligible when compared with the reachability computation runtime.

Tool Developers For tool developers such output can also be illuminating. One unexpected finding was that the visualization output for a buck-boost converter differed between HyCreate2 and Flow* for the same model, shown in Figure 4. If the produced reach sets do not intersect, this is indicative of a bug in one of the tools (or in the translation process). In this case, the issue was not caused by the reachability algorithm but rather by the visualization output of Flow* being strictly six digits after the decimal point, which is not sufficient for the time scales considered for this fast-switching system. This was confirmed by rescaling time in the model file (using milliseconds as the X-axis instead of seconds), which corrected the visualization of the reach set.

Researchers Many research results are applicable to general hybrid automata models, and not only a specific tool. Reimplementing these results in each tool would require significant effort, and be an error prone process. Piecemeal implementation of such results is also problematic because it is not clear if a tool is superior to another one because of an optimization performed on the model, or the underlying algorithm, or due to subtle differences in the implementation of the technique. By implementing generic model transformations in HYST, effort and errors can be reduced, and a more fine-grained comparison of tools becomes possible.

For example, the time-scaling performed on the buck-boost system in Figure 4 was done using a time-scaling model transformation pass. Using a command-line flag to HYST, the user can select the pass to perform and time scale desired, and the output model will be modified accordingly.

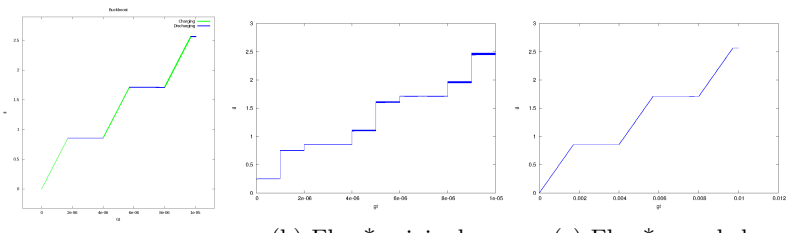
Another model transformation pass implemented in HYST is the method of pseudo-invariants (PI) [4]. This method splits an individual mode of a hybrid automaton into several using a set of provided conditions (called pseudo-invariants). The modes after splitting have identical dynamics to the mode they came from, and the transformed automaton is bisimilar to the original automaton. However, these artificial discrete transitions allow accumulating the set of states being tracked for tools that use flow-pipe construction methods. This can, in certain cases, serve to increase computation accuracy and reduce computation time.

To demonstrate this pass, we used the 2-d nonlinear model of a FitzHugh-Nagumo Neuron, using the dynamics and initial states given by Dang et al. [11]. This system was converted to both Flow* and HyCreate2 (which support nonlinear dynamics). Without pseudo-invariants, neither tool can complete a single cycle within the state-space of the system due to accumulated error. By passing the appropriate flag and parameters to HYST, a modified model is produced, resulting in visibly improved accuracy in the computed reach set for both tools. Other model-transformations passes implemented in Hyst include expression simplification and zeno behavior regularization.

4 Conclusion

In this paper, we present a source-to-source conversion tool called HYST for hybrid automata. The tool is capable of quickly converting a model to a number of hybrid system model checking

[1]HYST and examples are available at: http://www.verivital.com/hyst/



(a) HyCreate2 (b) Flow* original (c) Flow* rescaled

Figure 4: The reach set computed by HyCreate2 and Flow* appears to differ. By rescaling time in the Flow* model, this is confirmed as a result of the number of decimal digits Flow* outputs, rather than a bug in the tool.

tools. Additionally, it supports model transformation passes, which serve to both ease conversion, and allow generic application of model-transformation research results. As future work, we plan to extend HYST to more case studies and more tools, and we welcome contributions of tool authors interested to integrate in the HYST framework.

Acknowledgment

The material presented in this paper is based upon work supported by the Air Force Research Laboratory's Information Directorate (AFRL/RI) through the Visiting Faculty Research Program (VFRP) under contract number FA8750-13-2-0115 and the Air Force Office of Scientific Research (AFOSR). Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the AFRL/RI or AFOSR. This work was also partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS, http://www.avacs.org/), by the European Research Council (ERC) under grant 267989 (QUAREM) and by the Austrian Science Fund (FWF) under grants S11402-N23 (RiSE) and Z211-N23 (Wittgenstein Award).

DISTRIBUTION A. Approved for public release; Distribution unlimited. (Approval AFRL PA #88ABW-2015-0468, 09 FEB 2015)

References

[1] Aditya Agrawal, Gyula Simon, and Gabor Karsai. Semantic translation of simulink/stateflow models to hybrid automata using graph transformations. *Electronic Notes in Theoretical Computer Science*, 109:43–56, 2004.

[2] Rajeev Alur, Radu Grosu, Yerang Hur, Vijay Kumar, and Insup Lee. Modular specification of hybrid systems in Charon. In Nancy Lynch and BruceH. Krogh, editors, *Hybrid Systems: Computation and Control*, volume 1790 of *LNCS*, pages 6–19. Springer Berlin Heidelberg, 2000.

[3] Stanley Bak. HyCreate: A tool for overapproximating reachability of hybrid automata.

[4] Stanley 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 2014)*, pages 40–43, 2014.

[5] Stanley Bak, Sergiy Bogomolov, and Taylor 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.

[6] Luca Carloni, Maria D Di Benedetto, Alessandro Pinto, and Alberto Sangiovanni-Vincentelli. Modeling techniques, programming languages, design toolsets and interchange formats for hybrid systems, 2004.

[7] Chunqing Chen, Jun Sun, Yang Liu, JinSong Dong, and Manchun Zheng. Formal modeling and validation of stateflow diagrams. *International Journal on Software Tools for Technology Transfer*, 14(6):653–671, 2012.

[8] Xin Chen, Erika Abraham, and Sriram Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. *2013 IEEE 34th Real-Time Systems Symposium*, 0:183–192, 2012.

[9] A Cimatti, S. Mover, and S. Tonetta. Hydi: A language for symbolic hybrid systems with discrete interaction. In *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*, pages 275–278, August 2011.

[10] Thao Dang, Colas Le Guernic, and Oded Maler. Computing reachable states for nonlinear biological models. In *Computational Methods in Systems Biology*, pages 126–141. Springer, 2009.

[11] Thao Dang and Romain Testylier. Reachability analysis for polynomial dynamical systems using the Bernstein expansion, 2012.

[12] Tommaso Dreossi and Thao Dang. Parameter synthesis for polynomial biological models. In *Proceedings of the 17th international conference on Hybrid systems: computation and control*, pages 233–242. ACM, 2014.

[13] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable verification of hybrid systems. In *Computer Aided Verification (CAV)*, LNCS. Springer, 2011.

[14] Sicun Gao, Soonho Kong, and Edmund Clarke. Satisfiability modulo ODEs. In *International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, October 2013.

[15] Manish Goyal. Translation between cif and spaceex/phaver. Master's thesis, Verimag Research Lab, 2011.

[16] T. A. Henzinger. The theory of hybrid automata. In *IEEE Symposium on Logic in Computer Science (LICS)*, page 278, Washington, DC, USA, 1996. IEEE Computer Society.

[17] Charles Antony Richard Hoare. *Communicating sequential processes*, volume 178. Prentice-hall Englewood Cliffs, 1985.

[18] Taylor T. Johnson. *Uniform Verification of Safety for Parameterized Networks of Hybrid Automata*. PhD thesis, University of Illinois at Urbana-Champaign, Electrical and Computer Engineering, Urbana, IL 61801, 2013.

[19] C. Livadas, J. Lygeros, and N. A. Lynch. High-level modelling and analysis of tcas. In *IEEE Real-Time Systems Symposium*, pages 115–125, 1999.

[20] Karthik Manamcheri, Sayan Mitra, Stanley Bak, and Marco Caccamo. A step towards verification and synthesis from Simulink/Stateflow models. In *Proc. of the 14th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC)*, pages 317–318. ACM, 2011.

[21] Shiwei Ran, Jinzhi Lin, Ying Wu, Jianzhong Zhang, and Yuwei Xu. Converting ptolemy ii models to spaceex for applied verification. In Xian-he Sun, Wenyu Qu, Ivan Stojmenovic, Wanlei Zhou, Zhiyang Li, Hua Guo, Geyong Min, Tingting Yang, Yulei Wu, and Lei Liu, editors, *Algorithms and Architectures for Parallel Processing*, volume 8630 of *LNCS*, pages 669–683. Springer International Publishing, 2014.

[22] Peter Schrammel and Bertrand Jeannet. From hybrid data-flow languages to hybrid automata: A complete translation. In *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '12, pages 167–176, New York, NY, USA, 2012. ACM.

[23] D.A. van Beek, M.A. Reniers, R.R.H. Schiffelers, and J.E. Rooda. Foundations of a compositional interchange format for hybrid systems. In Alberto Bemporad, Antonio Bicchi, and Giorgio Buttazzo, editors, *Hybrid Systems: Computation and Control*, volume 4416 of *LNCS*, pages 587–600. Springer Berlin Heidelberg, 2007.