Tutorial: Software Tools for Hybrid Systems Verification, Transformation, and Synthesis: C2E2, HyST, and TuLiP

Parasara Sridhar Duggirala[1],Chuchu Fan[2], Matthew Potok[2], Bolun Qi[2], Sayan Mitra[2], Mahesh Viswanathan[2], Stanley Bak[3], Sergiy Bogomolov[4], Taylor T. Johnson[5], Luan Viet Nguyen[5], Christian Schilling[6], Andrew Sogokon[5], Hoang-Dung Tran[5], Weiming Xiang[5]

***Abstract*— Hybrid systems have both continuous and discrete dynamics and are useful for modeling a variety of control systems, from air traffic control protocols to robotic maneuvers and beyond. Recently, numerous powerful and scalable tools for analyzing hybrid systems have emerged. Several of these tools implement automated formal methods for mathematically proving a system meets a specification. This tutorial session will present three recent hybrid systems tools: C2E2, HyST, and TuLiP. C2E2 is a simulated-based verification tool for hybrid systems, and uses validated numerical solvers and bloating of simulation traces to verify systems meet specifications. HyST is a hybrid systems model transformation and translation tool, and uses a canonical intermediate representation to support most of the recent verification tools, as well as automated sound abstractions that simplify verification of a given hybrid system. TuLiP is a controller synthesis tool for hybrid systems, where given a temporal logic specification to be satisfied for a system (plant) model, TuLiP will find a controller that meets a given specification.**

I. INTRODUCTION

Hybrid systems have both continuous and discrete dynamics and are useful for modeling a variety of control systems, from air traffic control protocols to robotic maneuvers and beyond. Recently, numerous powerful and scalable tools for analyzing hybrid systems have emerged. Several of these tools implement automated formal methods for mathematically proving a system meets a specification. This tutorial session will present three recent hybrid systems tools: C2E2, HyST, and TuLiP and each tool as well as the tutorial plan is discussed next. A very brief overview of TuLiP is given and more technical details are given in a companion paper [1].

For C2E2, [1] P.S. Duggirala is with University of Connecticut, and [2] C. Fan, M. Potok, B. Qi, S. Mitra, and M. Viswanathan are with the University of Illinois at Urbana-Champaign. For HyST, [3] S. Bak is with the Air Force Research Laboratory, [4] S. Bogomolov is with IST Austria, [5] L.V. Nguyen, T.T. Johnson, A. Sogokon, H.D. Tran, and W. Xiang are with the University of Texas at Arlington, and [6] C. Schilling is with the University of Freiburg. For TuLiP, see [1].

II. C2E2: Simulation-Based Verification of Hybrid Systems

Compare-Execute-Check-Engine (C2E2) is a tool that implements a simulation-based verification algorithm for hybrid systems. The input to C2E2 is a Stateflow model (or a hybrid system in an XML format) with possibly nonlinear ordinary differential equations (ODEs) and a safety specification. For verification, C2E2 compiles the ODEs using a validated numerical solver, generates simulations, and computes an over-approximation of the set of reachable states. If the over-approximation of the reachable states satisfies (or violates) the safety specification, then C2E2 terminates, otherwise it computes a more precise over-approximation and repeats. We would demonstrate the following features of C2E2 (a) the graphical user interface, (b) specifying the safety properties, and (c) verifying the properties and visualizing the reachable set, which helps in building intuition about the behaviors of the hybrid system.

A. Overview

Hybrid systems are proposed as a new framework for modeling Cyber-Physical Systems where environment evolves according to nonlinear ODE and software is given as a finite state machine. One of the common ways to check whether a given hybrid system satisfies a functional specification, is to perform sample simulations. Further, when the environment is given as a nonlinear ODE, since the closed form solution for the initial value problem might not exist, simulations are the only known scalable way for checking specification. However, these simulations do not provide any formal guarantees. Simulation based verification technique is hence an attractive framework as it leverages the existing testing procedures and provides formal guarantees. We present C2E2, a tool that leverages the scalability of numerical simulations to either prove safety specification of CPS descried as Stateflow models or as hybrid systems.

C2E2 can be used for verifying bounded time safety specification. Given a hybrid system H, the set of initial states Θ, set of unsafe states U, and time bound T, C2E2 can verify whether any behavior of the system starting from the initial set Θ reaches an unsafe state in U within T time units. The main algorithm implemented in C2E2 [2] performs the following 4 steps iteratively: *simulate*, *bloat*, *check*, *refine*. C2E2 first partitions the initial

set Θ into a collection of neighborhoods. Then, it selects a state from each neighborhood and simulates the behavior of hybrid system H from the state. A new procedure, called *on-the-fly discrepancy computation* computes the value ϵ, an upper bound on the distace between the trajectories in the neighborhood selected. Then a safety check is performed to infer whether the trajectories from the neighborhood are safe, or a counterexample that violates the safety is discovered. If neither can be inferred, then C2E2 computes a finer partitioning of the initial set and repeats the process. C2E2 was able to verify realistic case studies of aircraft landing protocols [3] and automotive control systems [4].

B. Features of C2E2

The algorithm implemented in C2E2 has several features that will be demonstrated. These features are provided below.

Discrepancy function computation: For verification of hybrid models using simulations, one of the most important steps is to compute the *bloating factor*, i.e., the amount by which a sample simulation should be bloated in order to compute an overapproximation of the reachable set of states. In C2E2, this bloating factor is compted using what we call a discrepancy function. When the nonlinear dynamics satisfies special properties such as contraction behavior [5] or is incrementally stable [6], this discrepancy function can be provided by the user. For verifying general nonlinear systems, C2E2 implements a new technique that computes a local discrepancy function automatically. To compute these local discrepancy functions on-the-fly [7], the upper bound of the eigenvalues of (the symmetric part of) Jacobian matrices and the upper bound of the matrix perturbations are obtained along the simulation traces using Python linear algebra library[1]. The local discrepancy function module takes as input a given simulation traces and neighborhood, and returns local discrepancy function specially for the given simulation trace. In C2E2, we use the guaranteed simulator CAPD [8] to produce validated simulation traces.

Symbolic Jacobian computation: Computing the discrepancy function automatically requires computing numerical values of the Jacobian matrix of the differential equation. C2E2 therefore implements a new procedure for computing symbolically the Jacobian matrix. For an ODE $\frac{dx}{dt} = f(x)$, where f is a vector valued function, the Jacobian matrix $J(x)$ is the matrix of partial derivatives $J_{ij}(x) = \frac{\partial f_i}{\partial x_j}$. We use the Python Sympy [2] library for computing derivatives of f symbolically. This library handles a general class of functions and as a result our implementation of symbolic Jacobian computation works for all standard polynomial, trigonometric, exponential and logarithmic functions. It worked for complicated models like the powertrain benchmark [9], [10] which has more than 30 nonlinear terms in f. C2E2 compiles the symbolic Jacobian matrices into a Python module, which is then used to evaluate their numerical values.

Global discrepancy for linear ODEs: For linear time invariant hybrid models, the entries in the symbolic Jacobian matrix are constants. Thus, the local discrepancy function will be the same as the global one. C2E2 takes advantage of this fact and evaluates the Jacobian matrix just once and computes a global exponential discrepancy function to be used throughout, instead of on-the-fly local discrepancy. For example, analysis of a 28-dimensional linear model of the helicopter with this approach completes in seconds.

Automatic handling of constant dynamics: Often hybrid models have timers, and other variables that evolve at a constant rate with time. The ODEs for such a system have the form:

$$\begin{cases} \frac{dx}{dt} = f(x) \\ \frac{dy}{dt} = k \end{cases}$$

where k is a constant and y changes at that constant rate with time. Although the simple dynamics of y should make it easier to compute its reach set—at any time t, $y(t) = y(0) + kt$—our discrepancy-based algorithm has problems dealing with such systems. These constant-rate variables introduce all 0 rows and all 0 columns in the Jacobian matrix. This not only increases the dimension of the system, but also introduces extra conservatism in the estimation of the eigenvalues. For example, the Jacobian matrix of such systems has 0 eigenvalue even when the rest of the system is stable. The algorithm in C2E2 mitigates this problem by automatically decomposing the system by handling the constant-rate part independently.

For example, the Cardiac cell model in [11] uses a timer $\frac{d(\text{timer})}{dt} = 1$ to transit between the location where stimulate is on and the location where it is off. Systems with such constant dynamics are detected and decomposed automatically. That is, C2E2 will first compute the reach set of $\frac{dx}{dt} = f(x)$ using our standard technique, then bloat $y(t)$ by $\delta_y kt$ for $\frac{dy}{dt} = k$, where δ_y is the size of initial set for variable y.

Coordinate transformation: Coordinate transformation can help produce less conservative overapproximations of the discrepancy functions. Coordinate transformations are done automatically in C2E2 in the following manner: first, Jacobian matrix is transformed to the real Jordan form by a similarity transformation, and then the similarity transformation matrix is used to perform the linear coordinate transformation. Such transformation decreases the conservatism of exponential bound, but comes with the price of a constant multiplicative factor in $\beta(t)$. C2E2 allows the users to set a parameter that helps explore this trade off.

Usability and Visualization Features: C2E2 has both GUI and a command line interface. The GUI aids in reducing the learning curve associated with using formal

[1] http://docs.scipy.org/doc/numpy/reference/routines.linalg.html

[2] http://www.sympy.org/en/index.html

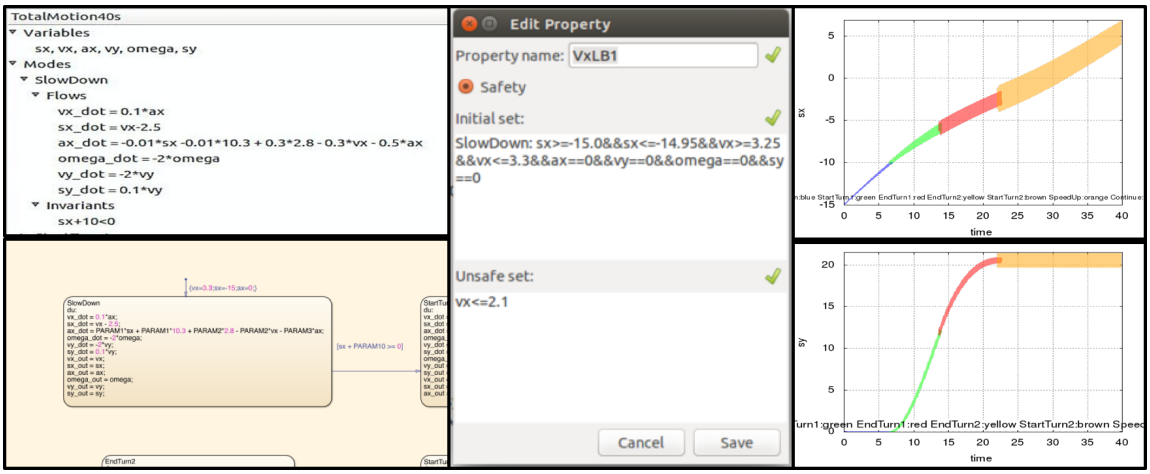

Fig. 1: Left to right: figures showing a snippet of partial adaptive cruise control model in C2E2 front end and Stateflow^{TM}, property dialog, plots of reachable set for adaptive cruise control model.

verification tools and the command line interface can be used by advanced users to invoke C2E2 from other tools and develop new verification techniques. To improve the usability and ease of installation, C2E2 comes with an installation script and a set of testing scripts. The tests check the reach sets computed on a new installation against the corresponding reference versions computed in our lab machine. C2E2 also has an in-house visualization feature (built using gnuplot) that helps in visualizing the set of reachable states of the system and also provide intuition about the system behaviors. The visualizer also shows the unsafe regions and counter-example segments. Examples inputs and outputs are documented in the website[3].

C. Tutorial Plan

The tutorial in C2E2 would illustrate the following features: 1) syntax and semantics of input hybrid systems models, 2) semantics and verification of the safety specification, and 3) using the grapical user interface and the terminal interface for verifying safety properties of hybrid models with various configurations. Additioanlly, new experimental features such as falsification using sample simulations would be demonstrated.

III. HyST: A Hybrid Source Transformation and Translation Tool

Algorithmically analyzing hybrid systems models is challenging in theory and in practice [12], [13]. 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 [14] and practical results to ease reachability analysis [15]–[17]. 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 tutorial 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.

A. Overview

A hybrid automaton [18] 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 [19]–[22], Flow* for nonlinear dynamics [23], dReach for nonlinear dynamics [24], and HyComp [25] for polynomial dynamics. HyST is a source transformation and translation tool for hybrid automaton models [26]. 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 [26], HyST now includes support for additional model transformation passes, networks of hybrid automata, and additional output formats (HyComp). This tutorial will illustrate

[3] http://publish.illinois.edu/c2e2-tool/example/

```
1    <?xml version="1.0" encoding="iso-8859-1"?>
2  <sspaceex xmlns="http://www-verimag.imag.fr/xml-namespaces/sspaceex"
       version="0.2" math="SpaceEx">
3    <component id="main">
4      <param name="x" type="real" local="false" d1="1" d2="1" dynamics
         ="any" />
5      <param name="y" type="real" local="false" d1="1" d2="1" dynamics
         ="any" />
6      <location id="1" name="running" x="164.0" y="194.0" width="146.0"
        height="140.0">
7        <flow>
8         x' == y &amp;
9         y' == (1-x*x)*y-x
10       </flow>
11     </location>
12   </component>
13   <component id="sys">
14     <param name="x" type="real" local="false" d1="1" d2="1" dynamics
         ="any" controlled="true" />
15     <param name="y" type="real" local="false" d1="1" d2="1" dynamics
         ="any" controlled="true" />
16     <bind component="main" as="main_1" x="228.0" y="118.0">
17       <map key="x">x</map>
18       <map key="y">y</map>
19     </bind>
20   </component>
21 </sspaceex>
```

Fig. 2: Van der Pol oscillator in SpaceEx format.

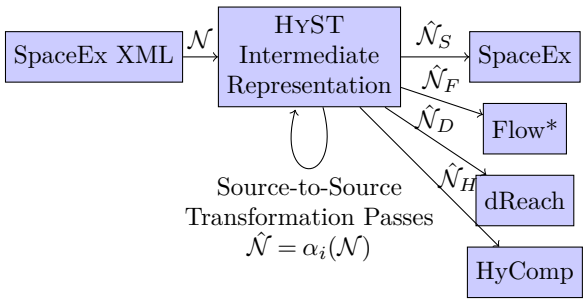

Fig. 3: HyST overview: a network of hybrid automata in the SpaceEx format $\mathcal{N}$ is parsed into the intermediate representation, on to which model transformation passes α_i are applied. Various syntactic transformations are also applied before exporting to the supported tools.

HyST's usage to interface tools and the recent transformation passes.

B. 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.[4] 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 3 shows the high-level architecture of HyST. The tutorial will consist of showing how to apply passes to hybrid automata, modify examples, and use scripts to automatically execute the supported tools.

C. Hybridization

Hybridization is a technique whereby one seeks to approximate a continuous system with non-linear dynamics by a hybrid system in which the continuous dynamics is in some sense simpler [27], [28]. 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 (more details may be found in [29]). 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 [29].[5]

D. 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 [30]. 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.

E. 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 [31].

[4] HyST is available online: http://verivital.com/hyst/

[5] http://verivital.com/hyst/pass-hybridization/

F. Order-Reduction

Order-reduction is a common approach in systems and control to simplifying the analysis of systems. Roughly speaking, it creates a reduced-order system with fewer state variables (decreased dimensionality) such that the reduced-order system exhibits behaviors similar to those of the original, or full-order, system [32]. 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 [33] and deriving error bounds from numerical simulations [34] 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) [32]. 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 over-approximations.[6]

G. 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 [35], phase-locked loop (PLL) and full-wave rectifier (FWR) circuits [36], nonlinear systems frequently used as benchmarks in the numerical analysis community [37], purely continuous nonlinear systems that have been verified to be safe [38], larger-scale linear systems frequently used in controls and order-reduction [32], [39].

H. Tutorial Plan

Overall, the tutorial on HyST will illustrate the current features already implemented 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 in HyST.

IV. Control Design for Hybrid Systems with TuLiP: The Temporal Logic Planning Toolbox

This tutorial describes TuLiP, the Temporal Logic Planning toolbox. The companion tutorial paper [1] describes TuLiP in more technical detail. TuLiP is a collection of tools for designing controllers for hybrid systems from specifications in temporal logic. The tools support a workflow that starts from a description of desired behavior in temporal logic, and a description of the system to be controlled. The system can be represented as a discrete transition system, or a hybrid dynamical system with a mixed discrete and continuous state space. The system description can include both discrete and continuous uncontrollable variables that represent disturbances, communication signals, and other environmental factors that affect the system dynamics and controller decisions.

For solving the control design problem, the logic specification is refined, by conjoining it with a discrete description of system dynamics in logic, which is an abstraction of the underlying continuous dynamics, in the case of hybrid systems. For piecewise affine dynamical systems, this abstraction is constructed automatically, guided by the geometry of the dynamics and under logical constraints from the specification. The resulting logic formulae describe admissible discrete behaviors that capture both controlled and environment variables. To find a controller, the toolbox solves a game of infinite duration. Existence of a discrete (winning) strategy for the controlled variables in this game is a proof certificate for the existence of a controller for the original problem that guarantees the satisfaction of the specification. This discrete strategy, refined with continuous controllers when needed, yields a feedback controller for the original hybrid system. The toolbox frontend is written in Python, with backends in C, Python and other languages.

The tutorial starts with an overview of the theory behind TuLiP, and of its software architecture, organized into specification frontends and backends that implement algorithms for abstraction, solving games, and interfaces to other tools. Then, the main elements for writing a specification for input to TuLiP are introduced. These include logic formulae, labeled transition systems, and hybrid dynamical systems, with linear or piecewise affine continuous dynamics. The working principles of the algorithms for predicate abstraction and discrete game solving using nested fixpoints will be explained, by showing an input specification through the various transformations that compile it to a symbolic representation that scales well to large games. The tutorial concludes with design examples that demonstrate the toolbox's capabilities.

V. Summary

Overall, this tutorial session will feature three recent hybrid systems tools for specifying, verifying, and synthesizing controller software interacting with physical plants. This paper presented some details on two of the tools, C2E2 and HyST, and the companion paper [1] describes TuLiP in detail.

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

References

[1] S. Dathathri, I. Filippidis, S. C. Livingston, R. M. Murray, and N. Ozay, "Control design for hybrid systems with tulip: The temporal logic planning toolbox (tutorial paper)," in *Multi-conference on systems and control*, 2016.

[2] P. S. Duggirala, S. Mitra, and M. Viswanathan, "Verification of annotated models from executions," in *EMSOFT*, 2013.

[3] P. S. Duggirala, L. Wang, S. Mitra, M. Viswanathan, and C. Muñoz, "Temporal precedence checking for switched models and its application to a parallel landing protocol," in *FM 2014: Formal Methods - 19th International Symposium, Proceedings*, 2014, pp. 215–229.

[4] P. S. Duggirala, C. Fan, S. Mitra, and M. Viswanathan, "Meeting a powertrain verification challenge," in *Computer Aided Verification*. Springer, 2015, pp. 536–543.

[5] W. Lohmiller and J. J. E. Slotine, "On contraction analysis for non-linear systems," *Automatica*, 1998.

[6] D. Angeli, "A lyapunov approach to incremental stability properties," *IEEE Trans. Automat. Contr.*, 2000.

[7] C. Fan and S. Mitra, "Bounded verification with on-the-fly discrepancy computation," *13th International Symposium on Automated Technology for Verification and Analysis*, 2015.

[8] *Computer Assisted Proofs in Dynamic Groups (CAPD)*, http://capd.ii.uj.edu.pl/index.php.

[9] X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. Butts, "Powertrain control verification benchmark," in *Proceedings of the 17th international conference on Hybrid systems: computation and control*. ACM, 2014, pp. 253–262.

[10] ——, "Benchmarks for model transformations and conformance checking," in *1st International Workshop on Applied Verification for Continuous and Hybrid Systems (ARCH)*, 2014.

[11] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok, "C2e2: A verification tool for stateflow models," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2015, pp. 68–82.

[12] A. Fehnker and B. H. Krogh, "Hybrid system verification is not a sinecure," in *Automated Technology for Verification and Analysis*. Springer, 2004, pp. 263–277.

[13] H. Guéguen and J. Zaytoon, "On the formal verification of hybrid systems," *Control Engineering Practice*, vol. 12, no. 10, pp. 1253 – 1267, 2004, analysis and Design of Hybrid Systems.

[14] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 971–984, 2000.

[15] A. Tiwari, "Abstractions for hybrid systems," *Formal Methods in System Design*, vol. 32, no. 1, pp. 57–83, 2008.

[16] ——, "Hybridsal relational abstracter," in *Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings*, 2012, pp. 725–731.

[17] S. Sankaranarayanan, T. Dang, and F. Ivančić, "Symbolic model checking of hybrid systems using template polyhedra," in *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, 2008, pp. 188–202.

[18] 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*, vol. 138, no. 1, pp. 3–34, 1995.

[19] 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)*, ser. LNCS. Springer, 2011.

[20] 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*, pp. 1–19, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10009-015-0393-y

[21] 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)*, ser. LNCS, vol. 8855. Springer, 2014, pp. 116–131.

[22] 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*, ser. LNCS. Springer, 2013.

[23] X. Chen, E. Abraham, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, N. Sharygina and H. Veith, Eds. Springer Berlin Heidelberg, 2013, vol. 8044, pp. 258–263.

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

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

[26] 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.

[27] E. Asarin, T. Dang, and A. Girard, "Hybridization methods for the analysis of nonlinear systems," *Acta Inf.*, vol. 43, no. 7, pp. 451–476, 2007.

[28] T. Dang, O. Maler, and R. Testylier, "Accurate hybridization of nonlinear systems," in *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2010, Stockholm, Sweden, April 12-15, 2010*, 2010, pp. 11–20.

[29] 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, Apr. 2016.

[30] 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: IEEE Computer Society, Dec. 2015.

[31] S. Bak, "Reducing the wrapping effect in flowpipe construction using pseudo-invariants," in *Proceedings of the 4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*, ser. CyPhy '14. New York, NY, USA: ACM, 2014, pp. 40–43.

[32] 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.

[33] A. Girard and G. J. Pappas, "Approximate bisimulation relations for constrained linear systems," *Automatica*, vol. 43, no. 8, pp. 1307 – 1317, 2007.

[34] Z. Han and B. Krogh, "Reachability analysis of hybrid control systems using reduced-order models," in *American Control Conference, 2004. Proceedings of the 2004*, vol. 2, June 2004, pp. 1183–1189.

[35] 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.

[36] O. A. Beg, A. Davoudi, and T. T. Johnson, "Charge pump phase-locked loops and full wave rectifiers for reachability analysis (benchmark proposal)," in *Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Vienna, Austria, Apr. 2016.

[37] 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.

[38] A. Sogokon, K. Ghorbal, and T. T. Johnson, "Non-linear continuous systems for safety verification (benchmark proposal)," in *Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Vienna, Austria, Apr. 2016.

[39] H.-D. Tran, L. V. Nguyen, and T. T. Johnson, "Large-scale linear systems from order-reduction (benchmark proposal)," in *Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Vienna, Austria, Apr. 2016.