Some Lessons from the HyTech Experience*

Thomas A. Henzinger† Jörg Preußig‡ Howard Wong-Toi§

Abstract

We provide an overview of the current status of the tool HyTech, and reflect on some of the lessons learned from our experiences with the tool. HyTech is a symbolic model checker for mixed discrete-continuous systems that are modeled as automata with piecewise-constant polyhedral differential inclusions. The use of a formal input language and automated procedures for state-space traversal lay the foundation for formally verifying properties of hybrid dynamical systems. We describe some recent experiences analyzing three hybrid systems. We point out the successes and limitations of the tool. The analysis procedure has been extended in a number of ways to address some of the tool's shortcomings. We evaluate these extensions, and conclude with some desiderata for verification tools for hybrid systems.

1 Introduction

Digital controllers are appearing in more and more embedded systems, often in either performance-critical or safety-critical situations, e.g., in engine controllers, navigation systems, medical equipment, and automated manufacturing systems. For these systems, it is highly desirable to have guarantees of correctness. Prototyping and simulation are two means of validation, but neither gives full assurance that all test situations have been checked. Furthermore, prototyping is often too expensive. Formal verification involves the construction of mathematically rigorous models, a means of expressing formal specifications, and a mechanism for proving that the models meet their specifications.

In our model-checking paradigm, systems are modeled as collections of concurrent hybrid automata. A hybrid automaton is a finite labeled transition system augmented with continuous variables subject to differential inclusions specific to each discrete mode. Properties of the system are proven by performing fixpoint computations over the automaton state space. These computations cannot be performed exactly for general continuous dynamics. The class of linear hybrid automata enforces certain linearity restrictions such that all relevant state sets are polyhedral, and all necessary operations over them are effectively computable. More general systems must be conservatively approximated by these restricted automata before analysis.

HyTech implements semi-decision procedures for linear hybrid automata, and thus provides automated support for verification [19]. It has been used successfully to analyze numerous systems

*An abbreviated version of this paper will appear in the *Proceedings of the 40th Annual IEEE Conference on Decision and Control* (CDC 2001).

†Department of Electrical Engineering & Computer Sciences, University of California, Berkeley, CA 94720, USA. tah@eecs.berkeley.edu. This research has been supported in part by the DARPA SEC grant F33615-C-98-3614, the MARCO GSRC grant 98-DT-660, the AFOSR MURI grant F49620-00-1-0327, and the NSF Theory grant CCR-9988172.

‡Lexware GmbH & Co. KG, Jechtinger Str. 8, 79111 Freiburg, Germany. joergp@lexware.de. Research performed while at Process Control Laboratories, Department of Chemical Engineering, University of Dortmund, Germany.

§Cadence Berkeley Laboratories, 2001 Addison St., 3rd floor, Berkeley, CA 94704, USA. howard@cadence.com.

of practical interest. This paper provides an overview of the latest developments in the tool. Recent efforts have concentrated on combating problems that arise when manipulating high-dimensional polyhedra using limited-precision arithmetic.

We discuss the lessons learned from three recent case studies: an engine controller, a communication protocol, and an active-structure controller. The systems differ in their levels of abstraction and relative discrete versus continuous complexity. For each system, we highlight the strengths and limitations of the HyTech approach. We discuss how effectively the latest analysis procedures handle these examples. From this experience, we offer advice about which systems are most suitable for HyTech, and suggest some strategies for completing an analysis successfully.

2 The HyTech Methodology

2.1 Analysis of hybrid systems by state-space traversal

We briefly review the basic concepts underlying HyTech [1, 19]. The verification methodology consists of the following steps: (1) identify the concurrent components of the system; (2) identify the communication mechanisms between the components; (3) model each component using hybrid automata; (4) conservatively approximate each hybrid automaton by a linear hybrid automaton; (5) analyze the collection of linear hybrid automata components with the model-checking tool HyTech. The steps are iterated as required if the abstractions prove to be too coarse, or if HyTech is unable to successfully terminate its analysis.

A *linear hybrid automaton* consists of a finite control graph —whose nodes are called *control modes*, and whose edges are called *control switches*— together with a set X of continuous variables. The continuous dynamics within each control mode are subject to a constant polyhedral differential inclusion, i.e., dynamics of the form $\phi(\dot{x})$, where ϕ is a conjunction of linear constraints over first derivatives of X. Discrete dynamics are modeled by the control switches, each of which has a guard condition and a reset condition over X. A *state* is a pair consisting of a control mode and a vector of variable values. A set of states is *linear* if it is definable using a linear predicate — a finite conjunction of linear constraints — for each control mode.

A common analysis task involves finding the set of states reachable from an initial set. For linear hybrid automata, the set of states that are successors from a linear set, by either a continuous flow or a discrete jump, is linear and effectively computable [1]. The heart of HyTech is a procedure that attempts to compute reachable sets, by repeatedly computing successor sets until no new successor states are discovered. Termination is not guaranteed, since the reachable set may consist of a potentially countably infinite union of linear successor sets. Other built-in routines include methods for extracting parametric constraints for system correctness and the generation of debugging traces.

2.2 Common obstacles to analysis

Analysis with HyTech may fail to terminate for a number of non-independent reasons. First, the size of the reachable state space may exceed available memory. Second, the analysis may take too long. Third, for certain linear hybrid automata, the reachable set is not linear, and HyTech's attempt to visit all reachable states will simply not terminate. Fourth, arithmetic overflow may occur.

We elaborate on the fourth problem above, since it is this obstacle that most often prevents the successful use of HyTech. Consider the automaton in Figure 1, where control begins in mode A with $(d, u) = (1, 0)$. The states in mode B that are forward reachable following $n - 1$ iterations of the loop at mode A are of the form $s_n = (u = 0, d = 2^{-n})$. Exact reachability analysis would

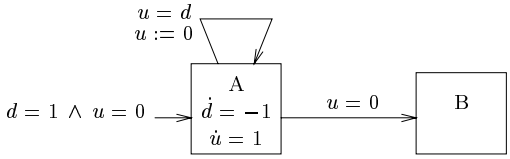

Figure 1: Executions enter mode B with state $(u = 0, d = 2^{-n})$ after n loop iterations at mode A.

involve countably infinite steps of the reachability procedure, since s_n cannot be reached in fewer iterations. In fact, HyTech would halt with arithmetic overflow, since it is unable to store s_n, for large n, with its limited-precision arithmetic. Arithmetic overflow can easily occur even when an exact reachability analysis would terminate. The use of large constants in the definition of the automata can quickly lead to polyhedra whose representation requires high degrees of precision. Unfortunately, in general it is difficult to round coordinates conservatively without destroying the shape of the resultant polyhedra. Instead, HyTech checks the validity of each polyhedral operation and exits with an error message when problems are detected.

3 Alternative Analysis Procedures

We describe some recent attempts to avoid the arithmetic overflow problem by using different underlying mechanisms for storing and/or computing reach sets. While our prototype implementations of these methods have been helpful in some case studies, certainly there is need for further research before producing robust procedures with wider applicability.

3.1 Rectangular automata

A linear hybrid automaton is *rectangular* if all its linear predicates define multi-dimensional rectangles. Rectangular automata occur naturally in many situations, such as the modeling of drifting clocks, or objects with bounded velocities. Furthermore, there are methods to abstract more complicated continuous dynamics into rectangular differential inclusions [20, 33]. We describe three specialized analysis procedures for rectangular automata. The procedures all store sets in $\mathbb{R}^n$ as the convex hull of sets in $\mathbb{R}^{n-1}$ along the orthogonal faces of a rectangular invariant. They vary in the class of $(n-1)$-dimensional sets employed. The procedures provide a reduction in dimensionality, and, in conjunction with the simple rectangular dynamics, easier and more robust means of computing time successors.

The procedure RectExact uses $(n-1)$-dimensional polyhedra along the faces, giving an exact representation of the reachable states [34]. Overflow errors can be avoided by associating rounding directions with each point. The rounding directions encode whether the point is responsible for extremal (upper or lower bounding) behavior during continuous evolution: if a point were derived by maximal rate in x, then the x coordinate should be rounded up. RectExact uses excessive memory unless the number of points used in the polyhedral representations is reduced. We use the public domain tool Qhull, which guarantees minimization over three-dimensional polyhedra, thereby making RectExact robust over $\mathbb{R}^4$.

The procedure RectApprox avoids general polyhedra altogether, and uses only rectangles for $(n-1)$-dimensional sets [32]. The algorithm is extremely fast, and numerically robust, since it is clear in which direction to overapproximate each bounding value. Compared to RectExact, it is far faster, and uses less memory. However, while it is exact for systems with two continuous vari-

ables, in higher dimensions, its usefulness is limited since it must overapproximate non-rectangular successor sets using rectangles.

One of the strengths of HyTech is its ability to extract automatically correctness conditions for systems defined with parametric values. The procedure RectParam is a natural generalization of RectApprox for handling *parametric rectangular automata*. In these automata, bounds may be expressed as linear terms over the parameters, e.g., one may specify $x \leq k_0 + k_1\alpha_1 + k_2\alpha_2$, for rational constants $k_0 \ldots k_2$ and parameters α_1 and α_2. RectParam stores sets of states using $(n-1)$-dimensional parametrically-bounded rectangles. The procedure is only exact for $n = 2$, and we expect it will be too approximate in higher dimensions.

3.2 Expanding the dynamics: interval methods

HyperTech, a successor tool to HyTech, performs analysis over a far wider range of continuous dynamics than HyTech allows, thereby eliminating the need for abstraction in many cases [21]. HyperTech uses interval methods to compute rectangles that are guaranteed to enclose all time successors originating from a given rectangle [31]. The continuous dynamics permitted by HyperTech are given by differential equations of the form $dx_i/dt = f(x_1, \ldots, x_n)$, where f is a composition of polynomials, exponentials, and trigonometric functions. In particular, multi-modal linear systems can be handled directly. The convenience of directly modeling more complex continuous dynamics and the robustness of the results make this approach extremely appealing. The tool is effective if one can determine the appropriate time-steps that will allow for accurate yet computationally efficient analysis.

3.3 Accelerating termination

Many control systems involve the repeated execution of control loops. In some cases, the loops are nested, or involve busy-waiting phases where the value of some variable is checked on a regular basis until some enabling condition occurs. Explicit reachability analysis of such systems can lead to either an extraordinary amount of repetitive analysis, or even non-termination due to infinite loops. Extrapolation operators can accelerate reachability analysis by guessing that other states will be reachable based on those already reached. We experimented with the following operator. Assume P and Q are closed polyhedra. We extrapolate P using Q by (1) taking the hull H of P and Q, and then (2) removing from H every defining constraint for which the closure of its negation does not intersect P. This operator is tighter than (and therefore less aggressive than) both the widening operator of [14] and the extrapolation operator of [16], which is not monotone in its second argument. The implementation in HyTech enables one to specify where to perform the extrapolation. Intelligent choice of control modes is crucial to achieve overapproximation that is sufficient for forcing or accelerating termination, and yet accurate enough to produce useful results.

4 Recent Examples

For each project below, we describe both where HyTech is effective, and where its limitations are tested. Details of the verification of the engine controller and the active-structure controller can be found in [39] and [40], respectively.

4.1 Engine controller in cut-off mode

4.1.1 System description and specification

We consider control of an automotive engine once the driver has released the accelerator pedal [5]. The control objective is to reach injection cut-off while minimizing oscillations in acceleration, as these create passenger discomfort. The system consists of three modules: the driveline, the cylinders, and the controller. The driveline model is a fifth-order linear system. Each of the car's four cylinders cycle through four phases in lock-step: intake, compression, expansion, and exhaust. The controller makes its injection decisions at the beginning of the exhaust phase, based on state predictions three and four phases into the future. We verify that the hybrid controller keeps the system quantifiably close to an optimal switching line while guaranteeing convergence.

4.1.2 Analysis and discussion

In the first attempts at verifying the property above, the state space was partitioned into blocks, with the linear dynamics overapproximated by a rectangular differential inclusion within each block. The standard version of HyTech encountered arithmetic overflow for both the original five-dimensional system, and a three-dimensional abstraction. The different non-trivial rate constraints at each block of the partition, coupled with the constraints defining the blocks, led to accumulated precision, and ultimately overflow, in the rational coefficients defining the polyhedra for reachable states. This problem appears to be inevitable when the analysis must accurately track continuous trajectories with varying tangents. The procedure RectApprox of subsection 3.1, successfully terminated, but was too coarse: the overapproximation of the reach set violated the property. Successful analysis was only possible after abandoning attempts at phase-portrait approximation, and instead performing a manual discrete-based abstraction to reduce the three-dimensional system to a two-dimensional one. The tool CheckMate uses an alternative approach that avoids these abstractions by directly accepting linear dynamics [36].

4.2 Biphase mark protocol

We describe the semi-automated parametric analysis of a biphase mark protocol [30]. While other authors have already formally analyzed this protocol [30, 25, 38], here we use automated tools to generate correctness conditions. Further details will appear in a forthcoming publication.

4.2.1 System description and specification

The biphase mark protocol is a commonly used method for encoding bit sequences sent between hardware devices. The passage of time is divided into cells of length n cycles each. During the first m cycles of a cell—known as the *mark subcell*—the sender toggles the wire value from the end of the last cell. Then over the latter part of the cell—known as the *code subcell*—the sender either leaves the wire unchanged to send a 0 bit, or toggles it again to send a 1 bit. The receiver samples the wire until it detects a new cell, then waits s cycles before sampling the wire again. The intention is that the mark subcell is long enough for the receiver to detect the start of the cell, and that the code subcell is long enough for the receiver to read the data bit correctly. Correctness is complicated by the following facts: the transmitted signal is not perfectly square; the sender and receiver clocks drift independently within known bounds; when the receiver reads the wire, the value obtained may correspond to some time during the preceding cycle, not necessarily the cycle boundary.

4.2.2 Analysis and discussion

The sender, the wire, and the receiver are naturally expressed as rectangular automata. Hence the specialized procedures of subsection 3.1 prove especially useful. While the standard HyTech could analyze the protocol for some fixed low-precision parameter settings, once again arithmetic overflows occurred when either the rates had high precision or during parametric analysis. When n, m, and s were treated as parameters, our initial attempts at analysis revealed that the receiver can check the wire an arbitrary number of times for the start of a cell, since the parameter n may be much larger than s. This behavior led to a non-terminating reachability analysis. We tackled this problem using two different techniques. In the first approach, in the case of perfect clocks, we modified the analysis by using the extrapolation operator of subsection 3.3 to force termination, enabling parametric verification. In the second approach, we obtained stronger results, but only after additional manual effort. We abstracted the receiver into a component that detected the start of a cell within a given timing delay, without repeated sampling of the wire. Exact reachability analysis over the abstracted receiver terminated, and enabled the synthesis of a correctness condition over n, m, and s for clocks of fixed bounded drift.

A further limitation of this study is that the system cannot be viewed in its full parametric form — either the rates are fixed, or the critical cycle counts are fixed — in order to keep the model within the class of linear hybrid automata.

4.3 Active-structure controller

4.3.1 System description and specification

A fault-tolerant structural-control system is formally defined and verified in [12]. The control system uses a pulse control algorithm that performs three basic tasks to limit vibration of the structure: sampling the state of the structure, updating its model of the structure, and applying a pulse. It uses a voting mechanism with three statically replicated sensors. A back-up actuator is dynamically invoked should the current actuation fail. Inter-pulse applications must occur within periodic time bounds.

4.3.2 Analysis and discussion

This control system is presented at a level where the only continuous aspect being modeled is time. Thus the system is an attractive target for real-time system verifiers and HyTech's parametric analysis procedures. Here, the prime limitation of HyTech is state-space explosion due to the number of discrete states, rather than arithmetic overflow. To tackle this bottleneck, we again manually produced abstractions of the system components, this time with the goal of reducing the number of control modes. We used HyTech to prove that a control agent and a sampling agent are abstractions of their respective components, and that when composed they satisfy the timing specification. This manual effort is tedious and error-prone, but the methodology should scale well.

5 Lessons Learned

We take a step back six years after the first public release of HyTech to reflect on some of the lessons learned. While some points below are HyTech-specific, others generalize to the broader field of verification of hybrid systems.

5.1 Success stories from the HyTech experience

HyTech has demonstrated that exhaustive state-space exploration methods, originally developed for discrete systems, can play a role in hybrid systems analysis.

State-space traversal for verification of hybrid systems. A number of successful case studies of non-trivial systems have demonstrated the feasibility of rigorously proving correctness properties of high-level models of mixed discrete-continuous systems using automated methods [23, 10, 17, 37, 35, 28, 4, 27, 39, 6, 7, 40, 24, 26, 11]. The ability to generate timed error traces for faulty systems is extremely useful for debugging.

Polyhedral abstraction of the continuous state space. By representing continuous state sets using polyhedra, HyTech makes a specific compromise between a purely discrete (but analyzable) abstraction of a system and a more general (but intractable) representation of sets. Thus, on the one hand, the restriction to linear hybrid automata as an input model leads to effectively computable analysis procedures. But on the other hand, for many systems, one cannot find polyhedral abstractions accurate enough without meeting computational or arithmetic bottlenecks. There are a number of alternative approaches to representing hybrid state spaces and/or performing successor computations for more expressive dynamics, e.g., polygonal projections [13], flow-pipes [9], ellipsoids [8], griddy polyhedra [3], level sets [29]. It is too early to tell how successful these approaches will be, but we believe the polyhedral abstractions of HyTech will continue to play a role in specialized but common situations, such as rectangular systems.

Parametric analysis. Most systems are defined using parameters. They are only intended to work correctly under certain parametric conditions. Furthermore, a top-down design methodology can benefit from verifying high-level parametric specifications, and then enjoying the freedom of parameter selection when implementing at lower levels of refinement. HyTech has shown that it is possible in some cases to perform parametric analysis.

Expressiveness of reachability properties. Although more general specifications can require more complex fixpoint computation to verify, it has been our experience that of the properties that can be handled by model checking, most can be adequately expressed as reachability properties, possibly by coupling the system model with a monitor component that checks for property violations.

5.2 Characteristics of systems suitable for HyTech

The HyTech procedures, and their rectangular extensions, appear to work best for systems with many of the following characteristics.

Small number of variables. To tackle the curse of dimensionality, the system should be abstracted to as few continuous variables as possible. Furthermore, discrete variables should be encoded into the control modes.

Small number of parameters. Analysis with a single parameter is often successful, but systems with complex relationships between multiple parameters and timing constants can quickly lead to arithmetic overflow.

Constant dynamics. Arithmetic errors are less likely to occur when the rate conditions are the same over most of the control modes. Rate conditions that differ in neighboring control modes tend to result in polyhedra with more faces — due to the different extremal rates being propagated.

Hence, systems consisting of concurrent components, each with its own clock with its own drifting rates, are good candidates for analysis.

Slowly-varying dynamics. Trajectories with fast changes in tangent are difficult to approximate accurately using phase-portrait approximation. Close tracking will require fine partitioning of the state space that is computationally expensive.

Low precision. The likelihood of arithmetic overflow is less when automata are defined with rates and constants of low relative precision. This rule-of-thumb suggests rounding constants up or down to lower their LCM/GCD ratio. The goal is to keep the polyhedra from developing large coefficients in their vertices or slopes. Lower LCMs can also lead to earlier termination over iterated loops.

HyTech is better suited to high-level system descriptions, where the continuous variables have either simple dynamics, or can be adequately abstracted to ones with simple dynamics, e.g., rate-bounded systems. The biphase mark protocol and parametric analysis of the active-structure controller are good examples. Our experience modeling the cut-off controller highlighted the difficulty HyTech has with even third-order linear dynamics, let alone nonlinear dynamics. Although many factors influence the practical complexity of a system, as a rough guide, when the continuous dynamics are simple, the current version of HyTech can often handle systems with about five continuous variables and five concurrent components of about five to ten modes each.

5.3 Further advice for using HyTech

In practice, analysis with HyTech involves repeated attempts at modeling and verification. Typically, the tool does not complete analysis of the first system model one tries, and the model must be abstracted, either on a per component basis, or by merging several components into a single one. It is a fine art to choose a level of abstraction that is simple enough for HyTech to complete and yet accurate enough for properties to be proven. As well as modifying the model according to the guidelines above (see also Section 9 of the user guide [18]), one can try the following: checking the states reached after a few iterations for modeling and computational problems; trying both forward and backward analysis; using extrapolation operators if the analysis appears to be following loops in the discrete control structure; and checking for invariants of the system that one hopes the abstraction would respect. These modeling and verification iterations are best performed jointly by someone having expertise in the operation of the tool as well as someone familiar with the system itself.

Many systems cannot be readily modeled and analyzed in HyTech's polyhedral abstraction, e.g., to track a closed loop in the continuous state space by approximating its dynamics can lead to an outwardly spiraling set of reachable states. In such cases, other means of verification should be explored. We believe there is a need for verifiers tuned for particular subclasses of hybrid systems, e.g., multi-modal linear dynamical systems, a more general class of systems that may still be restrictive enough for accurate tracking of continuous dynamics.

5.4 Shortcomings of HyTech

As pointed out throughout this paper, the area with most room for improvement is the robustness of HyTech's analysis engine. We now list some additional features we would have found useful when using HyTech. Formalisms with these goals in mind include Masaccio [15] and Charon [2].

Richer input language: HyTech uses a simple automaton input language. Many of our HyTech system descriptions would have benefited from (a) support for better and cleaner commu-

nication mechanisms between processes, (b) a hierarchical description language including encapsulation of variables, (c) templates for defining and then creating instances of processes.

Interactive mode. Enhanced debugging features could include an interactive mode with ties to a visual input language and the visualization of diagnostic error traces.

Methods for decomposing verification tasks into simpler subtasks. In our case studies, the curse of dimensionality appears all too often, especially when one considers either distributed control implementations or complex continuous dynamics. A comprehensive hybrid systems verifier should support hierarchical and compositional reasoning and analysis, such as refinement checks and assume-guarantee reasoning [22].

Support for hybrid abstraction. While it is already possible to automatically produce timing abstractions for certain subclasses of automata, it would be helpful to develop mechanisms, either automatic, or semi-automatic, to help guide the search for a wider variety of hybrid abstractions, such as dimension-reducing abstractions.

Acknowledgements

We acknowledge the following contributors to the HyTech project—Grzegorz Czajkowski, Pei-Hsin Ho, Benjamin Horowitz, Peter Kopke, and Rupak Majumdar. We thank Frits Vaandrager for many useful comments on modeling the biphase mark protocol.

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*, 138:3–34, 1995.

[2] R. Alur, R. Grosu, Y. Hur, V. Kumar, and I. Lee. Modular specifications of hybrid systems in CHARON. In *HSCC 2000: Hybrid Systems—Computation and Control*, LNCS 1790, pp. 6–19. Springer, 2000.

[3] E. Asarin, T. Dang, O. Maler, and O. Bournez. Approximate reachability analysis of piecewise-linear dynamical systems. In *HSCC 2000: Hybrid Systems—Computation and Control*, LNCS 1790, pp. 20–31. Springer, 2000.

[4] G.S. Avrunin, J.C. Corbett, and L.K. Dillon. Analyzing partially-implemented real-time systems. *IEEE Trans. on Software Engineering*, 24(8):602–614, 1998.

[5] A. Balluchi, M.D. Benedetto, C. Pinello, C. Rossi, and A. Sangiovanni-Vincentelli. Cut-off in engine control: a hybrid system approach. In *Proc. of the 36th IEEE Conf. on Decision and Control*, 1997.

[6] B. Bérard and L. Fribourg. Automated verification of a parametric real-time program: the ABR conformance protocol. In *CAV 99: Computer Aided Verification*, LNCS 1633, pp. 96–107. Springer, 1999.

[7] B. Bérard and L. Fribourg. Reachability analysis of (timed) Petri nets using real arithmetic. In *CONCUR 99: Concurrency Theory*, LNCS 1664, pp. 178–193, Springer, 1999.

[8] O. Botchkarev and S. Tripakis. Verification of hybrid systems with linear differential inclusions using ellipsoidal approximations. In *HSCC 2000: Hybrid Systems—Computation and Control*, LNCS 1790, pp. 73–88. Springer, 2000.

[9] A. Chutinan and B.H. Krogh. Verification of polyhedral-invariant hybrid automaton using polygonal flow pipe approximation. In *HSCC 99: Hybrid Systems—Computation and Control*, LNCS 1569, pp. 76–90. Springer, 1999.

[10] J.C. Corbett. Timing analysis of Ada tasking programs. *IEEE Trans. on Software Engineering*, 22(7):461–483, 1996.

[11] G. Delzanno. Automatic verification of parameterized cache coherence protocols. In *CAV 2000: Computer aided verification*, LNCS 1855, pp. 53–68, Springer, 2000.

[12] W.M. Elseaidy, R. Cleaveland, and J.W. Baugh Jr. Modeling and verifying active structure control systems. *Sci. of Computer Programming*, 29(1-2):99–122, July 1997.

[13] M.R. Greenstreet. Verifying safety properties of differential equations. In *CAV 96: Computer Aided Verification*, LNCS 1102, pp. 277–287. Springer, 1996.

[14] N. Halbwachs. Delay analysis in synchronous programs. In *CAV 93: Computer-aided Verification*, LNCS 697, pp. 333–346. Springer, 1993.

[15] T.A. Henzinger. Masaccio: a formal model for embedded components. In *TCS 00: Theoretical Computer Science*, LNCS 1872, pp. 549–563. Springer, 2000.

[16] T.A. Henzinger and P.-H. Ho. A note on abstract-interpretation strategies for hybrid automata. In *Hybrid Systems II*, LNCS 999, pp. 252–264. Springer, 1995.

[17] T.A. Henzinger and H. Wong-Toi. Using HyTech to synthesize control parameters for a steam boiler. In *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, LNCS 1165, pp. 265–282. Springer, 1996.

[18] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HyTech. In *TACAS 95: Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 1019, pp. 41–71. Springer, 1995.

[19] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1):110–122, 1997.

[20] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Trans. on Automatic Control*, 43(4):540–554, 1998.

[21] T.A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. Beyond HyTech: Hybrid systems analysis using interval numerical methods. In *HSCC 2000: Hybrid Systems—Computation and Control*, LNCS 1790, pp. 130–144. Springer, 2000.

[22] T.A. Henzinger, M. Minea, and V. Prabhu. Assume-guarantee reasoning for hierarchical hybrid systems. In *HSCC 2001: Hybrid Systems—Computation and Control*, LNCS 2034, pp. 275–290. Springer, 2001.

[23] P.-H. Ho and H. Wong-Toi. Automated analysis of an audio control protocol. In *CAV 95: Computer-Aided Verification*, LNCS 939, pp. 381–394. Springer, 1995.

[24] P.-A. Hsiung, F. Wang, and Y.-S. Kuo. Scheduling system verification. In *TACAS 99: Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 1579, pp. 19–33. Springer, 1999.

[25] D.V. Hung. Modeling and verification of biphase mark protocols in duration calculus using PVS. In *Proc. of 1998 Intl. Conf. on Application of Concurrency to System Design*, pp. 88–98. IEEE Computer Society Press, 1997.

[26] S. Ivanov and D. Griffioen. Verification of a biphase mark protocol. Tech. Report CSI-R9915, Computing Science Institute, Univ. of Nijmegen, The Netherlands, 1999.

[27] S. Kowalewski, S. Engell, R. Huuck, Y. Lakhnech, B. Lukoschus, and L. Urbina. Using model-checking for timed automata to parameterize logic control programs. In *Proc. of the 8th European Symp. on Computer Computer Aided Process Engineering (ESCAPE 8)*, 1998.

[28] S. Kowalewski, O. Stursberg, M. Fritz, H. Graf, I. Hoffmann, J. Preußig, M. Remelhe, S. Simon, and H. Treseler. A case study in tool-aided analysis of discretely controlled continuous systems: the two tanks problem. In *Hybrid Systems V*, LNCS 1567, pp. 163–185. Springer, 1999.

[29] I. Mitchell and C. Tomlin. Level set methods for computation in hybrid systems. In *HSCC 2000: Hybrid Systems—Computation and Control*, LNCS 1790, pp. 310–323. Springer, 2000.

[30] J.S. Moore. A formal model of asynchronous communication and its use in mechanically verifying a biphase mark protocol. *Formal Aspects of Computing*, 6(1):60–91, 1994.

[31] R. Moore. *Interval Analysis*. Prentice-Hall, 1966.

[32] J. Preußig, S. Kowalewski, H. Wong-Toi, and T.A. Henzinger. An algorithm for the approximative analysis of rectangular automata. In *FTRTFT 98: Formal Techniques for Real-time and Fault-tolerant Systems*, LNCS 1486, pp. 228–40. Springer, 1998.

[33] J. Preußig, O. Stursberg, and S. Kowalewski. Reachability analysis of a class of switched continuous systems by integrating rectangular approximation and rectangular analysis. In *HSCC 99: Hybrid Systems—Computation and Control*, LNCS 1569, pp. 209–222. Springer, 1999.

[34] J. Preußig and H. Wong-Toi. A procedure for reachability analysis of rectangular automata. In *Proc. of American Control Conference*, pp. 1674–1678, 2000.

[35] C.W. Seibel, J.-M. Farines, and J.E.R. Cury. Towards using hybrid automata for the mission planning of unmanned aerial vehicles. In *Hybrid Systems V*, LNCS 1567, pp. 324–340. Springer, 1997.

[36] B.I. Silva and B.H. Krogh. Formal verification of hybrid systems using *CheckMate*: a case study. In *Proc. of the American Control Conference*, pp. 1679–1683, 2000.

[37] T. Stauner, O. Müller, and M. Fuchs. Using HyTech to verify an automotive control system. In *HART 97: Hybrid and Real-Time Systems*, LNCS 1201, pp. 139–153. Springer, 1997.

[38] F. Vaandrager. Analysis of a biphase mark protocol with uppaal, 2001. See http://www.cs.kun.nl/~fvaan/PAPERS/BMP/slides.pdf.

[39] T. Villa, H. Wong-Toi, A. Balluchi, J. Preußig, A. Sangiovanni-Vincentelli, and Y. Watanabe. Formal verification of an automotive engine controller in cutoff mode. In *Proc. of 37th IEEE Conf. on Decision and Control*, pp. 4271–4276, 1998.

[40] H. Wong-Toi. Modular verification of a fault-tolerant active control system: an example. In *CACSD 99: Proc. of IEEE Conf. on Computer-aided Control System Design*, pp. 103–108, 1999.