HyTech: A Model Checker for Hybrid Systems*

Thomas A. Henzinger** Pei-Hsin Ho*** Howard Wong-Toi†

Abstract. A hybrid system consists of a collection of digital programs that interact with each other and with an analog environment. Examples of hybrid systems include medical equipment, manufacturing controllers, automotive controllers, and robots. The formal analysis of the mixed digital-analog nature of these systems requires a model that incorporates the discrete behavior of computer programs with the continuous behavior of environment variables, such as temperature and pressure. *Hybrid automata* capture both types of behavior by combining finite automata with differential inclusions (*i.e.* differential inequalities). HyTech is a symbolic model checker for *linear hybrid automata*, an expressive, yet automatically analyzable, subclass of hybrid automata. A key feature of HyTech is its ability to perform parametric analysis, *i.e.* to determine the values of design parameters for which a linear hybrid automaton satisfies a temporal requirement.

From finite to hybrid automata. Model checking [11, 39], and in particular symbolic model checking [10] (*i.e.* the manipulation of state sets rather than individual states), has been proven an effective technique for the automatic analysis of complex finite-state systems. The first extensions of discrete models toward mixed discrete-continuous behavior considered real-numbered time [15]. One such model is the *timed automaton*—a finite automaton augmented with a finite number of real-valued clocks, *i.e.* continuous variables whose rate of change is always 1 [3]. Symbolic model checking for timed automata involves the computation of fixpoints over state sets that are represented by a restricted class of linear constraints, namely, boolean combinations of inequalities of the form $x \leq k$ and $x - y \leq k$, for clocks x and y and constants k [31]. Although the state space of a timed automaton is infinite, the fixpoint computation is guaranteed to terminate. *Hybrid automata* are extensions of timed automata that allow continuous variables with more general dynamics than clocks [2, 38, 1].

In contrast to most other formalisms for hybrid systems [18, 8, 5], where the proposed analysis methods are deductive, research on hybrid automata has focused on (semi)algorithmic state-space analysis. This has led to the definition of *linear hybrid automata*, for which the dynamics of the continuous variables are defined by linear differential inclusions of the form $\mathbf{A}\dot{\mathbf{x}} \leq \mathbf{b}$, for a constant matrix $\mathbf{A}$, a vector $\mathbf{x}$ of variables, and a constant vector $\mathbf{b}$.[1] For linear hybrid automata, the successor states of a state set defined by linear constraints are computable, and themselves linearly definable [4]. Thus, symbolic model checking for timed automata can be extended to linear hybrid automata, provided that state sets are represented by boolean combinations of arbitrary linear inequalities. The price to pay for the increased generality is the loss of guaranteed termination. The theory of hybrid automata—in particular, decidability results for subclasses of linear hybrid automata—is presented in [30, 21, 20, 28, 22, 29, 36].

* This research was supported in part by the ONR YIP award N00014-95-1-0520, by the NSF CAREER award CCR-9501708, by the NSF grant CCR-9504469, by the AFOSR contract F49620-93-1-0056, by the ARO MURI grant DAAH-04-96-1-0341, by the ARPA grant NAG2-892, and by the SRC contract 95-DC-324.036.

** EECS Department, University of California, Berkeley; tah@eecs.berkeley.edu.

*** Strategic CAD Labs, Intel Corporation, Hillsboro, Oregon; pho@ichips.intel.com.

† Cadence Berkeley Labs, Berkeley, California; howard@cadence.com.

[1] This definition of linearity differs from the definition commonly used in systems theory.

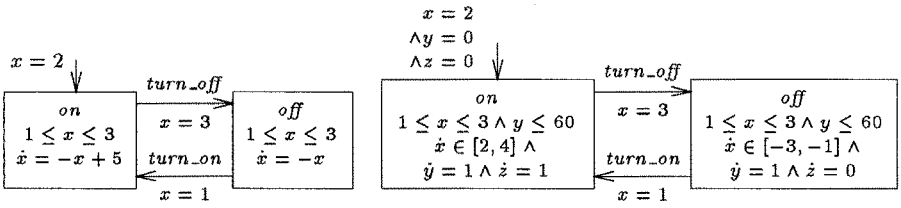

Fig. 1. Thermostat automaton

A simple example. A hybrid automaton is a nondeterministic finite transition graph whose nodes are labeled with differential inclusions. The hybrid automaton to the left in Figure 1 models a simple thermostat. The temperature x is initially 2 degrees, and rising at the rate of $-x+5$ degrees per minute. When the temperature reaches 3 degrees, the heater is turned off, and the temperature then falls at the rate of $-x$ degrees per minute. While the automaton control resides in a given node, the behavior of the continuous variables satisfies the node's differential inclusions. Nodes are also labeled with invariant conditions on the values of the variables. For example, the invariant of the node *on* is $1 \leq x \leq 3$, implying that the automaton control must leave the node before the temperature exceeds 3. Transitions between nodes may be guarded by constraints on the variables (*e.g.* the guard on the transition labeled *turn_off* is $x = 3$), and may incorporate reassignment of the variables, such as resetting a clock to 0. Shared event labels allow transitions in one hybrid automaton to be synchronized with transitions in another (this does not occur in the example).

For algorithmic analysis, we restrict our model to linear hybrid automata, described above. This formalism does not allow general differential inclusions, but is still quite expressive. For example, the model captures stopwatches (variables whose rates may be either 0 or 1), skewed clocks (variables whose rates are constant, possibly different from 1), and variables whose rates have constant bounds, such as clocks whose rates drift within some interval $[1-\epsilon, 1+\epsilon]$. Nonlinear hybrid automata may be analyzed either by translating them into linear hybrid automata where possible, or by conservatively approximating them with linear hybrid automata [23, 32].

For example, in order to analyze the proportion of time that the thermostat is on, we use the linear hybrid automaton to the right in Figure 1, which is derived from the original nonlinear hybrid automaton as follows. First, we overapproximate the nonlinear behavior of the temperature by placing lower and upper bounds on its rate within each node (*e.g.* in node *on*, the invariant $1 \leq x \leq 3$ implies that the rate $-x+5$ is bounded within the interval $[2, 4]$). Next, we introduce a clock y that measures the elapsed time, and a stopwatch z that measures the accumulated time spent in node *on*. We wish to check that the thermostat is on for less than $\frac{2}{3}$ of the first hour of operation. To ensure termination of the computation, we add the conjunct $y \leq 60$ to the invariants. HyTech then fully automatically verifies that no state satisfying $y = 60 \wedge z \geq \frac{2}{3}y$ is reachable from the initial state.

The tool. Early versions of HyTech were built using Mathematica [24, 34]. Building on the observation that a linear constraint over n variables can be represented by the union of n-dimensional polyhedra, the current, more efficient, generation of HyTech [27, 26] manipulates linear constraints via calls to a library for polyhedral operations [19]. A HyTech input file consists of two parts: a textual description of a collection of hybrid automata, and a sequence of analysis commands. The component automata are composed into a product representing the entire system. The analysis language provides access to a variety of operations for polyhedral manipulation within

a flexible framework for writing state-space exploration programs. For added convenience, there are built-in macros for reachability analysis, temporal operators, abstract interpretation [25], error-trace generation, and parametric analysis [13, 6].

In parametric analysis, the system is described using *design parameters*—symbolic constants with unknown, fixed values. Standard reachability analysis followed by existential quantification can be used to determine necessary and sufficient constraints on the parameters under which system violations cannot occur. Thus, rather than merely verifying (or falsifying) systems, HyTech can be used to extract quantitative information, further aiding the design process. Common uses for parametric analysis include determining minimum and maximum bounds on variables, and finding cutoff points for the placement of sensors or the values of timers. For example, to compute an upper bound on the time the thermostat is on during the first hour of operation, we introduce a parameter α and use HyTech to determine the values of α for which there is a reachable state satisfying $y = 60 \wedge z \geq \alpha$. HyTech returns the constraint $\alpha \leq 36$, implying that the thermostat is on for no more than 36 minutes during the first hour.

Applications. HyTech has been used in a number of case studies—primarily control-based applications—including a distributed robot controller [24], a robot system in manufacturing [24], the Philips audio control protocol [35], an active structure controller [26], a generalized railroad controller [26], a nonlinear temperature controller [23], a predator-prey ecology [32], an aircraft landing-gear system [37], a steam-boiler controller [33], and an automotive controller [40]. Corbett [12] has verified robot controllers written in a subset of Ada by automatically translating them into linear hybrid automata for analysis with HyTech. We are currently experimenting with the modeling and analysis of timed circuits.

Availability. HyTech has been ported to the following platforms: DEC workstations running Ultrix and Digital Unix, HP workstations running HP-UX, Sun workstations running SunOS and Solaris, and x86 PCs running Linux. HyTech's home page `http://www.eecs.berkeley.edu/~tah/HyTech` includes the source code, executables, an online demo, a user guide, a graphical front end (courtesy of members of the Uppaal project [9]), numerous examples, online versions of papers, and pointers to additional literature. Requests may also be sent to `hytech@eecs.berkeley.edu`.

Related tools. Polka analyzes linear hybrid systems with an emphasis on abstract-interpretation strategies [19]. Shift provides a simulation environment for generalized hybrid automata, but does not perform state-space analysis [16]. Symbolic model checkers for the more restrictive timed-automaton model include Kronos [14], timed Cospan [7], Uppaal [9], and Veriti [17].

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, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems I*, LNCS 736, pp. 209–229. Springer, 1993.
3. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
4. R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. Software Engineering*, 22:181–201, 1996.
5. R. Alur, T.A. Henzinger, and E.D. Sontag, eds. *Hybrid Systems III: Verification and Control.* LNCS 1066. Springer, 1996.
6. R. Alur, T.A. Henzinger, and M.Y. Vardi. Parametric real-time reasoning. In *Proc. 25th ACM Symp. Theory of Computing*, pp. 592–601, 1993.
7. R. Alur and R.P. Kurshan. Timing analysis in Cospan. In *Hybrid Systems III*, LNCS 1066, pp. 220–231. Springer, 1996.
8. P. Antsaklis, A. Nerode, W. Kohn, and S. Sastry, eds. *Hybrid Systems II.* LNCS 999. Springer, 1995.

9. J. Bengtsson, K.G. Larsen, F. Larsson, P. Pettersson, and W. Yi. UppAal: a tool-suite for automatic verification of real-time systems. In *Hybrid Systems III*, LNCS 1066, pp. 232–243. Springer, 1996.
10. J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98:142–170, 1992.
11. E.M. Clarke and E.A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, LNCS 131. Springer, 1981.
12. J. C. Corbett. Timing analysis of Ada tasking programs. *IEEE Trans. Software Engineering*, 22:461–483, 1996.
13. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Proc. 5th ACM Symp. Principles of Programming Languages*, pp. 84–97, 1978.
14. C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool Kronos. In *Hybrid Systems III*, LNCS 1066, pp. 208–219. Springer, 1996.
15. J.W. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, eds. *Real Time: Theory in Practice*. LNCS 600. Springer, 1992.
16. A. Deshpande, A. Göllü, and L. Semenzato. The Shift programming language and run-time system for dynamic networks of hybrid automata. PATH report, http://www-path.eecs.berkeley.edu/shift/doc/ieeshift.ps.gz, 1996.
17. D.L. Dill and H. Wong-Toi. Verification of real-time systems by successive over- and under-approximation. In *Computer-aided Verification*, LNCS 939, pp. 409–422. Springer, 1995.
18. R.L. Grossman, A. Nerode, A.P. Ravn, and H. Rischel, eds. *Hybrid Systems I*. LNCS 736. Springer, 1993.
19. N. Halbwachs, P. Raymond, and Y.-E. Proy. Verification of linear hybrid systems by means of convex approximation. In *Static Analysis Symp.*, LNCS 864, pp. 223–237. Springer, 1994.
20. M.R. Henzinger, T.A. Henzinger, and P.W. Kopke. Computing simulations on finite and infinite graphs. In *Proc. 36rd IEEE Symp. Foundations of Computer Science*, pp. 453–462, 1995.
21. T.A. Henzinger. Hybrid automata with finite bisimulations. In *ICALP: Automata, Languages, and Programming*, LNCS 944, pp. 324–335. Springer, 1995.
22. T.A. Henzinger. The theory of hybrid automata. In *Proc. 11th IEEE Symp. Logic in Computer Science*, pp. 278–292, 1996.
23. T.A. Henzinger and P.-H. Ho. Algorithmic analysis of nonlinear hybrid systems. In *Computer-aided Verification*, LNCS 939, pp. 225–238. Springer, 1995.
24. T.A. Henzinger and P.-H. Ho. HyTech: The Cornell Hybrid Technology Tool. In *Hybrid Systems II*, LNCS 999, pp. 265–293. Springer, 1995.
25. 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.
26. T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: the next generation. In *Proc. 16th IEEE Real-time Systems Symp.*, pp. 56–65, 1995.
27. T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HyTech. In *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS 1019, pp. 41–71. Springer, 1995.
28. T.A. Henzinger and P.W. Kopke. State equivalences for rectangular hybrid automata. In *Concurrency Theory*, LNCS 1119, pp. 530–545. Springer, 1996.
29. T.A. Henzinger and P.W. Kopke. Discrete-time control for rectangular hybrid automata. In *ICALP: Automata, Languages, and Programming*, LNCS. Springer, 1997.
30. T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proc. 27th ACM Symp. Theory of Computing*, pp. 373–382, 1995.
31. T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:193–244, 1994.
32. T.A. Henzinger and H. Wong-Toi. Linear phase-portrait approximations for nonlinear hybrid systems. In *Hybrid Systems III*, LNCS 1066, pp. 377–388. Springer, 1996.
33. 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.
34. P.-H. Ho. *Automatic Analysis of Hybrid Systems*. PhD thesis, Cornell Univ., 1995.
35. P.-H. Ho and H. Wong-Toi. Automated analysis of an audio control protocol. In *Computer-aided Verification*, LNCS 939, pp. 381–394. Springer, 1995.
36. P.W. Kopke. *The Theory of Rectangular Hybrid Automata*. PhD thesis, Cornell Univ., 1996.
37. S. Nadjm-Tehrani and J.-E. Strömberg. Proving dynamic properties in an aerospace application. In *Proc. 16th IEEE Real-time Systems Symp.*, pp. 2–10, 1995.
38. X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In *Hybrid Systems I*, LNCS 736, pp. 149–178. Springer, 1993.
39. J. Queille and J. Sifakis. Specification and verification of concurrent systems in Cesar. In *Symp. on Programming*, LNCS 137, pp. 337–351. Springer, 1981.
40. T. Stauner, O. Müller, and M. Fuchs. Using HyTech to verify an automotive control system. In *Hybrid and Real-Time Systems*, LNCS 1201, pp. 139–153. Springer, 1997.