Proceedings of the 40th IEEE
Conference on Decision and Control
Orlando, Florida USA, December 2001

An Assessment of the Current Status of Algorithmic Approaches to the Verification of Hybrid Systems

B. Izaias Silva[1], Olaf Stursberg[2], Bruce H. Krogh[1], Sebastian Engell[2]
[1] Dept. of Electrical and Computer Eng., Carnegie Mellon University,
Pittsburgh, PA 15213 (USA), {izaias,krogh}@cmu.edu
[2] Process Control Lab (CT-AST), University of Dortmund,
D-44221 Dortmund (Germany), {o.stursberg,s.engell}@ct.uni-dortmund.de

Abstract

This paper reviews the current status of implemented verification techniques for hybrid systems. We focus on tools that perform *model checking* for hybrid systems with varying levels of complexity. Features of the tools are described using a batch reactor example to illustrate what is required to develop an appropriate model for each tool. The concluding section suggests directions for future research and tool development based on the needs of industry for tools to perform verification and validation of designs for embedded control systems.

Key Words. hybrid systems, model checking, verification, computer-aided control system design.

1 Introduction

Formal verification refers to methods for determining whether or not given properties (specifications) are true for a given model of a dynamic system. During the past decade, tools for computer-based modeling and verification of discrete-state, or logical, systems have gained industrial importance, especially in the areas of hardware design and communication protocols [8]. Extending formal verification techniques to systems with continuous as well as discrete dynamics has been one of the main themes in the field of hybrid systems. Several research groups have introduced tools based on this research that offer the possibility to perform formal verification for various classes of hybrid system models using a variety of computational methods. The objective of this paper is to provide a brief survey of the current status of these tools and to offer some perspective on what needs to be done to develop tools that will reach the same level of acceptance in industry as achieved by the tools for purely discrete systems.

In general, there are two approaches to formal verification: *theorem proving* and *model checking*. Theorem proving aims at inferring (or contradicting) a specification for a system model using the methods of logical proof systems, whereas in the model checking approach the state-transition relation is used in iterative computations to arrive at the set of states (a fixed point) for which the given specification is true. The attractive feature of the theorem proving approach is that it is not restricted to finite-state systems, making it particularly appealing for hybrid systems. Theorem proving is not algorithmic, however, meaning that the proof procedure needs human direction and intervention for all but the most trivial verification problems. In contrast, algorithms exist and have been implemented for model checking using symbolic representations of sets of system states that, in many cases, solve verification problems of considerable complexity automatically [7]. The caveat is that these algorithms handle only finite-state systems.

This paper focuses exclusively on the model checking approach because this has been the approach used in the implementation of most of the tools for computer-based verification of hybrid systems. To perform model checking, a finite-state approximation (often called an *abstraction*) of the continuous dynamics needs to be constructed. Representing and constructing these finite-state abstractions is the main function and contribution of these tools. It is well known from the hybrid systems literature that only hybrid systems with very trivial continuous dynamics (integrators and some types of simple linear dynamics [14]) have equivalent finite-state models (called *bisimulations*) that can be used for verification. Consequently, verification of properties for the finite state approximation may be inconclusive. For example, discovering a state is reachable in the finite-state approximation may not imply it is reachable in the underlying hybrid system. When the verification results are inconclusive, the tools may provide mechanisms for making the finite-state approximation less conservative, but there can be no guarantee this process of refinement will terminate in general.

Several methods and tools for performing verification of hybrid system have been reported in the literature (see, e.g., [20, 15]). This survey paper provides a comparative overview of several of these tools from the following perspectives: the basic modeling formalism, specifications, the user interface, the computational representations, and computational routines. We use an example described in the following section to illustrate how the user develops models amenable to the formalism used by each tool described in Sec. 3. Verification results for the example from two tools developed by the authors are also included in this section. Based on experiences with the example application and on other case studies in literature, we assess the status of hybrid system verification tools in Sec. 4 and discuss directions for further development towards the goal of making these tools accessible and useful to industrial users.

2 A Batch Reactor System

To illustrate the type of problem which is typically considered in hybrid system verification, we use the discretely-controlled chemical batch reactor shown in Fig. 1 (taken from [19]): The reactor is filled by two liquid streams F_A and F_B with temperatures T_A, T_B and concentrations $c_{A,in}, c_{B,in}$ of two dissolved substances A and B. The streams can be controlled through the valves v_A and v_B in the inlet pipes. The stirred content of the reactor is cooled by a cooling jacket if the supply of cooling water is switched on by opening valve v_C. Cooling is necessary since an exothermic chemical reaction $2A + B \rightarrow D$ leads to an increase of the reactor temperature T_R. The reaction product can be discharged through the valve v_O which is (as all other valves) controlled by a discrete controller. Measurements of the temperature T_R, the liquid volume V_R, and the concentration c_A indicate whether these variables exceed specific thresholds or not. We consider the production procedure shown as a transition model in Fig. 2. Initially, one half of the reactor volume is already filled with solution B (and v_B is closed). In the first step (denoted by z_1), valve v_A is opened to supply the solution A until the volume V_R reaches an upper limit V_{High}. The chemical reaction leads to an

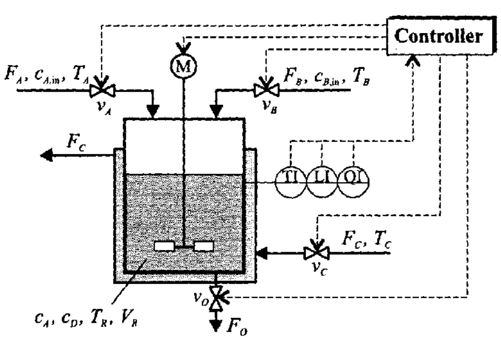

Figure 1: Batch reactor schematic.

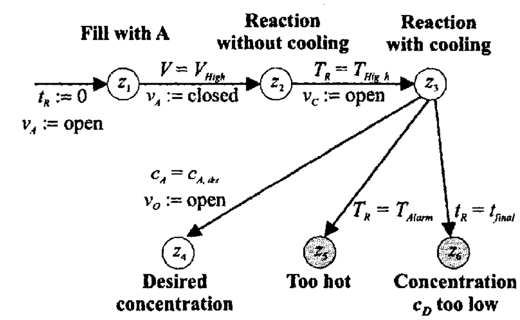

Figure 2: Operation procedure for the batch reactor.

increase of the temperature (state z_2) such that T_R eventually reaches a threshold $T_R = T_{High}$. From state z_3 (reaction with cooling) three different states can be reached: the 'normal' operation is that the concentration c_A has dropped down to $c_{A,des}$ corresponding to a sufficiently high concentration of the product D, and the reactor is emptied through valve v_O. If, alternatively, the temperature increases further to an upper threshold T_{Alarm}, the state z_5 is reached. (Note that T_R can show an over-shooting behavior when v_C is opened.) As a third possibility, a specified reaction time t_{final} can elapse before the desired concentrations are reached and the procedure terminates in state z_6. The reaction time is measured by a clock t_R, which is reset when valve v_A is opened.

Obviously, the states z_5 and z_6 should be excluded from the course of operation. A discrete controller has to be designed such that it switches the valves v_A, v_C, and v_0 in order to ensure that the operation always ends in state z_4. The objective of formal verification for this system is to determine if the temperature threshold $T_R = T_{High}$ is chosen appropriately to guarantee T_{Alarm} is never exceeded **and** to ensure that the desired product concentration is reached (for which T_{High} must not be chosen too low). The verification task requires a timed or hybrid model of the process behavior (i.e. of the continuous variables V, c_A and T_R). Tools that can handle switched nonlinear dynamics use the hybrid model given in Table 1 directly. For approaches that can handle hybrid systems with linear dynamics only, the ODEs for T_R are linearized for suitable regions of the state space. Further simplifications are needed for tools based on timed automata, as described in the following section.

3 Representative Model Checking Tools

This section surveys a representative selection of model checkers for hybrid systems and describes how they could be applied to the batch reactor system. We include one tool for timed automata, one for linear hy-

Table 1: Hybrid model of the reactor system.

Equations
$\frac{dc_A}{dt} = \frac{s_1 \cdot F_A}{V_R} \cdot (c_A^{in} - c_A) - 2 \cdot r$
$\frac{dT_R}{dt} = \frac{s_1 \cdot F_A}{V_R} \cdot (T_A - T_R) - \frac{s_2 \cdot k_C \cdot A_C}{\rho \cdot c_P \cdot V_R} \cdot (T_R - T_C) - \frac{H_R \cdot r}{\rho \cdot c_P}$
$\frac{dV_R}{dt} = s_1 \cdot F_A$, $\frac{dt_R}{dt} = 1$
$r = c_A^2 \cdot k_0 \cdot \exp\left(\frac{-E_A}{R_m \cdot T_R}\right)$, $A_C = \frac{\pi}{4} \cdot D_R^2 + \frac{4}{D_R} \cdot V_R$
Variables
c_A - concentration of substance A (kmole/m^3)
T_R - temperature of the reactor content (K)
V_R - liquid volume within the reactor (m^3)
t_R - reaction time (sec)
$s_1 \in \{0, 1\}$ - switch the valve v_A to close or open
$s_2 \in \{0, 1\}$ - switch the valve v_C to close or open

brid automata, another for hybrid systems with linear continuous dynamics, and two for systems with switched nonlinear dynamics. Three other tools are briefly described. More comprehensive descriptions of the tools UPPAAL, HYTECH, TAXYS/KRONOS, *d/dt* and the MLD-verifier can be found in the other papers in this invited session.

3.1 Uppaal

The tool UPPAAL is an environment to model, simulate, and verify systems represented as networks of timed automata (TA) [4]. These are defined according to [1] with an extension by data variables and synchronization mechanisms to model the communication of separate automata. UPPAAL can analyze simple liveness properties as well as reachability properties. By using *clock difference diagrams*, the timed automata are internally represented in a compact format (similar to that known as binary decision diagrams for discrete systems) that allows a relatively efficient analysis. Further information about UPPAAL can be found online at `www.docs.uu.se/docs/rtmv/uppaal/`.

In order to investigate the reactor system with UPPAAL, the relevant part of the process behavior has to be translated manually by the user into a set of concurrent TA. The model can then be entered using the UPPAAL graphical interface, illustrated in Fig. 3. In this figure, the controller corresponds to the operation procedure in Fig. 2. The variable t is a globally defined clock that measures the reaction time (limited to 3000 sec). The synchronization labels establish the communication with the automaton representing the reactor, and the labels are marked by '!' or '?' to denote the sender or the receiver, respectively. The upper part of the figure shows a TA that approximates the reactor behavior. The state 'Fill' denotes the filling procedure, the states 'S1' to 'S11' represent distinct regions for the temperature T_R and the concentration c_A, i.e., T_R decreases in direction from 'S1' to 'S9' and c_A increases in direction from 'S5' to 'S8'. The crucial and difficult modelling step is to determine the possible state transitions, including the guards and the state invariants. They can be obtained by simulation of the hybrid model given in Table 1, by measurements from the real plant, or by automated approximation techniques (see Sec. 3.5). Of course, the first two alternatives do not produce a conservative model in general.

The analysis of the composition of both automata using UPPAAL'S verification algorithm can then reveal if all behaviors terminate in one of the desired states 'S2', 'S5', or 'S9'. For the shown models, we get the result that the terminal state 'S10' is reachable. This state corresponds to the situation that the concentration c_A is still too high when the reaction is terminated, i.e., the desired product concentration c_D is not yet reached.

3.2 HyTech

HYTECH verifies specifications given as temporal logic expressions for so-called *linear hybrid automata* (LHA) (in which the continuous dynamics are specified by differential inclusions) using symbolic model checking in the continuous state space. HYTECH handles parallel composition of processes and parametric analysis. If the verification fails, it generates a diagnostic error trace. For further information about the

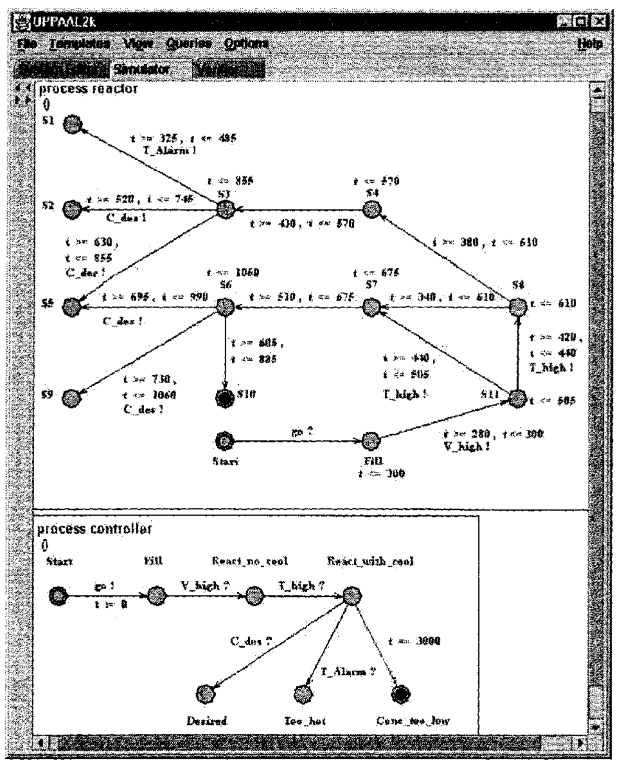

Figure 3: The reactor modeled in UPPAAL.

tool, see [13], and online information is available at `http://www-cad.eecs.berkeley.edu/~tah/HyTech/`. HYPERTECH, a new version of HYTECH, focuses on performing the reachability operations using interval approximation [12].

The batch reactor was modeled approximating the original system by a LHA. The system was verified using a Linux version of HYTECH. The developers of HYTECH suggest the following three approaches to verify systems of higher complexity than that of LHA [11]:

Clock transition models. In this approach, continuous state variables are replaced by *clock variables* (pure integrators with different rates) with constraints identifying the regions for which the given rates are valid. To make this model effective, the state space is partitioned into a number of regions and the original differential equations are represented by the rates of the clock variables. For the batch reactor example, the variables T_R, c_A and t_R would be replaced by clock variables. The switches between control modes are now made with respect to the clock variables.

Rate translation. This approach retains the original state variables, but approximates their continuous behavior with piecewise-constant bounds on the first derivatives. For the batch reactor example one possible partition of the state space is illustrated in figure 4. The number and geometry of the regions were chosen such that we obtained a good representation of the nonlinear dynamics by rate intervals.

Linear phase-portrait approximation. In this approach, the derivatives of the state variables can be constrained in linear combinations, e.g., $\dot{x}_1 \leq \dot{x}_2 + c_B$, giving a better approximation to the original state equations.

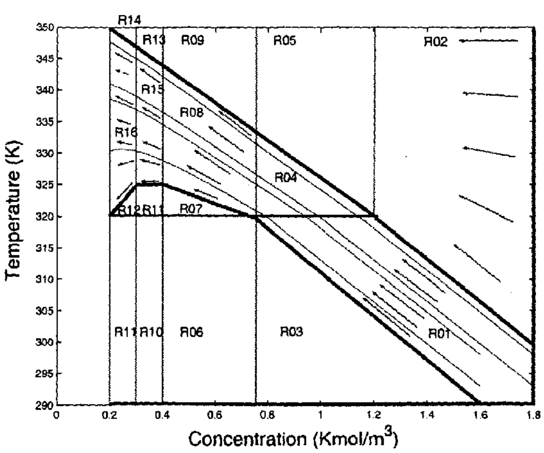

Figure 4: Partition of the state space of the batch reactor.

3.3 *d/dt*

The tool *d/dt* performs verification and control synthesis for hybrid systems with linear continuous dynamics. Reachable sets for linear dynamic systems are represented with collections of orthogonal (rectangular) polyhedra computed by performing so-called *face lifting* to create efficient over-approximations [9]. Collections of orthogonal polyhedra are represented with a very efficient data structure supported by a library of routines for performing the set operations required for model checking [6]. *d/dt* allows models with uncertainty in the input in the dynamics equation, i.e., models of the form $\dot{x} = Ax + Bu, u \in U$.

To apply the current version of *d/dt* to the batch reactor example, the user must linearize the system dynamics around the operating point of interest. To approximate the states reachable from the initial set in some time step ΔT, each face is moved by an amount that bounds all possible trajectories starting on the face. The approximation is then repeated beginning from the new set. Applying this process to partitions of the initial state set makes it possible to obtain arbitrarily close approximations to the actual reachable set. The system with temperature theshold $T = 320$K was verified used *d/dt*.

3.4 CheckMate

CHECKMATE is a MATLAB-based tool for simulation and verification of hybrid dynamic systems with arbitrary nonlinear continuous dynamics. CHECKMATE are constructed using logical operators (AND, OR, XOR, etc.), MUX/DEMUX, and the following custom Simulink blocks: *Switched Continuous System Blocks (SCSBs)* to model continuous dynamics (linear or nonlinear); *Polyhedral Threshold Blocks (PTHBs)* that generate 1 if the continuous input x lies within the polyhedron $Cx \leq d$ and 0 otherwise; and *Finite State Machine Blocks (FSMBs)*, which are MATLAB/Stateflow charts with inputs defined by logical functions of the outputs of PTHBs or FSMBs and a single scalar output (discrete data type) indicating the current active state of the finite-state machine.

CHECKMATE performs verification by computing a finite-state approximation using general polyhedral over-approximations to the sets of reachable states for the continuous dynamics called *flowpipes*. The system specification is expressed as a restricted CTL formula using only universal quantifiers. If the verification is successful, the user is informed and the program terminates. If the verification result is inconclusive, the user can ask CHECKMATE to refine the current approximation and attempt the verification again. CHECKMATE automatically searches for the states of the approximate transition system that led to failure, splits them, recomputes the reachable states for the new states and, again, evaluates the logic expression representing the specification. Properties of individual trajectories of the system can also be verified. Details about the tool are given in [17] and online information is available at `www.ece.cmu.edu/~webk/checkmate`.

Figure 5 shows a CHECKMATE model for the batch reactor example. The SCSB labeled *dynamics* models the switched continuous nonlinear dynamics for the system in Table 1. The operation of the controller is represented by the FSMB *status*. This block receives events representing the crossings of the temperature threshold to toggle the heating. Also, it detects whether a) the system overheats, b) the time for the reaction is exceeded or c) a low concentration of c_A is measured. The PTHBs *low_concentration, timeout, overheat* and *cross_up* generate discrete outputs 0 or 1 depending on whether or not one of these thresholds is crossed. Verification was performed to see if the system will achieve a concentration $c_A \leq 0.2$ within a time horizon of one hour. For the initial controller design, CHECKMATE returned a negative result even after two refinements of the finite-state approximation. To analyze the situation, we plotted the 3D flowpipe shown in figure

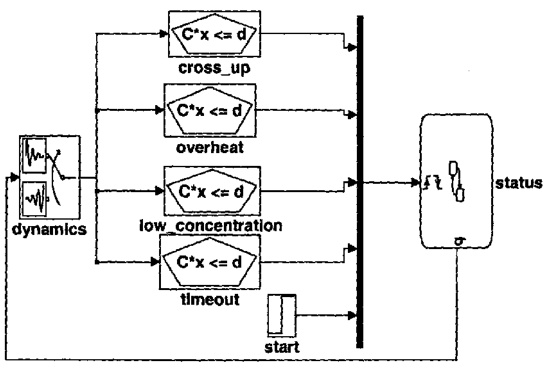

Figure 5: Batch reactor modeled in CHECKMATE.

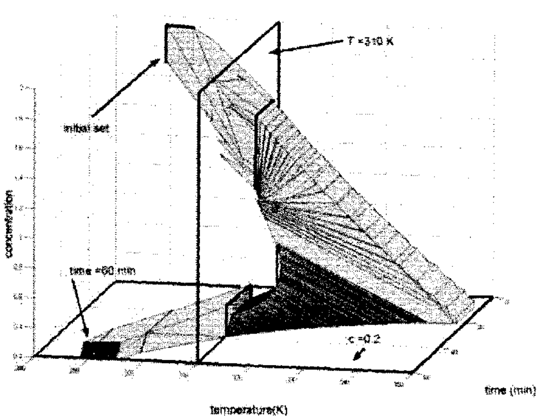

Figure 6: Flowpipe for verification with $T_{High} = 310$K.

6. We can see that the flowpipe hits the hyperplane corresponding to $t = 60$ min before hitting the concentration limit. Since the approximation is conservative, the result is inconclusive. We modified the controller threshold so that it would switch the cooler on at $T = 320$K. For this case, CHECKMATE verified the system will behave correctly on the first pass. The verification procedure for $T = 320$K took approximately 40 min. on a 600 MHz Pentium III PC.

3.5 Verdict

The tool VERDICT provides an environment for modular modelling and verification of timed and hybrid systems [19]. The modelling step is based on hybrid condition/event systems (HCES). In a block-diagram editor the structure of the system is built in a modular manner (usually a module for the controller and set of modules for the other components), and the communication is established by condition and event signals. The behavior of each module is described by a discrete, timed or hybrid transition system, where the latter can contain nonlinear ODEs with autonomous or externally triggered switching of the dynamics. While the hybrid modules are textually specified by a description language for HCES called *Celeste*, the modules with discrete and timed behavior can be build by a state-diagram editor also. The controller model can alternatively be imported as code given in the PLC languages *Instruction List* and *Sequential Function Charts*, and VERDICT comprises transformation algorithms to convert the code into modules given as HCES.

VERDICT translates the HCES model into the input languages of a choice of different model checkers for discrete and timed automata. So far VERDICT includes transformation routines into the input models of HYTECH, KRONOS, the tool SMV [8], and UPPAAL. The conversion routine includes an approximation step for the hybrid components of the HCES in order to get models to which these tools are applicable. Two different approximation algorithms are currently available, which both rely upon an orthogonal partitioning of the continuous state space of the hybrid modules [18]. One algorithm determines the transitions between different partition elements based on computation of representative trajectories. The other algorithm uses interval arithmetic, or constrained optimization, respectively, to approximate the original dynamics by differential inclusions, which then make it possible to determine the transitions and time constraints. The result is a TA model (specified in the languages of HYTECH, KRONOS or UPPAAL), a rectangular automaton (in HYTECH language), or an additional transformation generates a discrete automaton specified in SMV language. Online information about VERDICT is available at `astwww.chemietechnik.uni-dortmund.de/~verdict/`.

The VERDICT model of the reactor system is shown in Fig. 7. The right upper window contains the block-diagram containing one module each for reactor and controller. The measurements reported from the reactor to the controller are modeled as event signals, and the controller output (settings of the valves v_A and v_C) is given as condition signals. The controller model itself is specified as state-transition system (right lower window), and the hybrid behavior according to Tab. 1 is formulated in *Celeste* (left window).

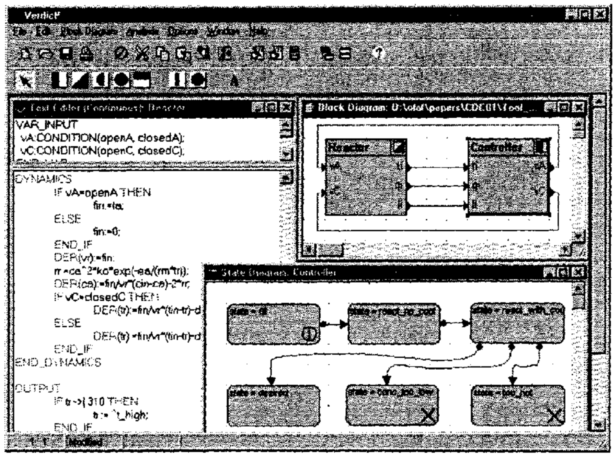

Figure 7: Reactor system in VERDICT.

For the given example, the analysis was carried out with HYTECH. Depending on the chosen partitioning and approximation method, the transformation of HCES lead to TA models of approx. 300 to 900 states and 4500 to 18000 transitions, and required between 7 and 54 min for computation (Pentium PC, 200 MHz). The analysis correctly shows that the critical state z_6 (respectively z_5 with $T_{Alarm} = 345\text{K}$) is reachable for a temperature threshold $T_{High} = 310\text{K}$ (respectively $T_{High} = 330\text{K}$). For the largest TA models the analysis could not be completed with HYTECH due to a huge memory consumption of more than 120MB. For the completed runs, the analysis took less than 10 minutes in all cases.

3.6 Other Tools

In addition to the tools described above, the following three hybrid system model checkers have attracted some attention recently. VERISHIFT: In contrast to the tools described above, which represent reachable regions by polyhedra, this tool uses ellipsoids to conservatively approximate the reachable sets for hybrid automata with linear differential inclusions [5]. The advantages of the ellipsoidal approximation are a reduced memory requirement and a polynomial complexity with respect to time (while the worst-case complexity of polyhedral operations is exponential). Online information is available at `www.robotics.eecs.berkeley.edu/~olegb`.

VERIFIER FOR MLD SYSTEMS: This tool uses mathematical programming to perform reachability analysis for Mixed Logical Dynamical (MLD) systems, which combine linear dynamics with logical propositions [3]. In contrast to the other tools discussed in this paper, MLDs model *discrete-time* hybrid systems. The algorithm determines the reachable set by solving a mixed-integer optimization problem. Online information is available at `www.aut.ethz.ch/~hybrid/verification.msql`.

KRONOS: This is a well-established model checker for TA. Based on a symbolic representation of the clock space by linear constraints for the clock values, forward or backward analysis reveals whether properties specified in the temporal logic TCTL are satisfied [10]. Recently, KRONOS has been extended by a compiler for the synchronous language *Esterel*, resulting in a tool named TAXYS. Information about KRONOS and related tools can be found online at `www-verimag.imag.fr/TEMPORISE/tools`.

4 The Status of Model Checking Tools for Hybrid Systems

On the one hand, the example considered in Sec. 3 as well as experiences gained from other applications show that the problem of analyzing discrete controllers for hybrid systems can be solved by available tools. On the other hand, the example reveals also that the modelling step demands some insight into the specific problem and the chosen modelling paradigm, and that it requires appropriate simplifications of the continuous dynamics for most of the tools.

The first step in verifying a given controlled system is to choose a suitable accuracy of the model. Clearly, the choice of a TA model is sufficient if the behavior of the controlled process can be accurately approximated by timed transitions. But for systems such as the reactor example, the determination of guards and invariants for a TA model is only possible by simulation of a more complex model or by experiments – neither method can ensure the completeness of the TA model. Hence, the use of verification techniques for hybrid systems with linear/nonlinear continuous dynamics is required. If linearizations of an originally nonlinear model are involved, the completeness issue has to be considered also. Having determined a suitable type of model, the best tool for analysis is usually that which is designed for that specific class of models. For example, if timed automata are used, there is no need to choose a tool for nonlinear hybrid systems, since UPPAAL and KRONOS provide special data structures that allow an efficient analysis of TA.

Table 2: Tool features.

Tool	UPPAAL	HYTECH	*d/dt*	CHECKMATE	VERDICT
Graphical user interface	✓	-	-	✓	✓
Simulation	✓	-	-	✓	-
Reachability specifications	✓	✓	✓	✓	✓
CTL specifications	✓	✓	-	✓	(✓)
Generates counterexamples	✓	✓	-	✓	(✓)
Type of model	TA	LHA	LDHA	NHA	NHA

Apart from the type of model, criteria such as modularity (and communication between modules), the type of state space partitioning, and the user-friendliness of the modelling environment are important for choosing a tool. With respect to the latter, a GUI that makes it possible to build the model graphically certainly enhances the acceptance by industrial users. Another very useful feature of a verification environment is a simulation facility. The usual course of analysis is to explore the system behavior by computing some single trajectories first. Based on the results, parameters governing the partitioning, the time steps, tolerances etc. are chosen such that an efficient verification is possible. Table 2 summarizes some properties and features of the tools that were described in Sec. 3.1 to 3.5. The type of models are denoted by: TA - timed automata, LHA - linear hybrid automata, LDHA - hybrid automata with linear ODEs, NHA - hybrid automata with nonlinear ODEs. For VERDICT, brackets are used if the feature depends on the chosen model checker.

With respect to the shortcomings of available tools, the following two points seem to be most important. First, the complexity of the computation restricts the applicability to fairly small systems. The verification of systems with around five continuous variables with nonlinear dynamics usually requires some hours of computation and also the memory consumption is enormous for some approaches. This problem immediately points to a second important shortcoming, namely, the lack of modularity in verification. The analysis of large and distributed systems may become possible if they can be decomposed into small modules for which model checking is possible. Techniques like deduction or theorem proving can then be used to derive global properties from the model checking results. A tool which follows this idea for reactive systems is MOCHA [2] – however, the extension of this approach to hybrid systems is an open problem.

Another shortcoming of current tools is the interpretation of the analysis results. If the verification fails (i.e., the required property is not fulfilled), most of the tools report a trajectory/trace that violates the specification. In most cases, it is not obvious how to redesign the controller such the requirements are met. Even more severe, there is no hint whether the violation results from an over-approximation of the real behavior or indeed from a wrong controller design. Usually, a refinement of the partitioning or the computation tolerances must reveal the actual cause of the violation.

5 Making Tools Useful for Industry

The assessment in the previous section identifies challenges that are well known to the research community. There are additional issues, however, that need to be addressed if tools for hybrid system verification are going to move out of research laboratories and become widely used in industry. Here we identify three key issues that have arisen often in our discussions with industrial colleagues.

Connecting with Existing Models. Tools for computer-aided control system design are now used extensively. This means that computer models of plants and controllers are readily available. It is unlikely that tools that require the construction of completely new models will gain wide acceptance, since the model building process is time consuming and can introduce errors that could make the verification results irrelevant to the actual system being developed.

Tools for Exploring Models and Results. Experience to date with tools for hybrid system verification shows that useful results can be obtained for nontrivial systems only when the user is involved in directing the verification process. Because of the difficulties that arise in computing and representing reachable sets for the continuous dynamics, tools that help the user gain insight into the system behavior are an absolute necessity.

Tools for Building Verification Specifications and Interpreting Results. One of the biggest barriers to the use of formal verification methods is the difficulty of translating requirement specifications into the formal specifications to be verified. Here industry needs tools that facilitate this translation, and also make it possible to interpret the results of the verification procedure in terms of the original application.

Although these issues are motivated by practical considerations, we believe each of them leads to significant research problems that need to be addressed by the developers of the tools for hybrid system verification. For example, there are interesting and challenging research problems in developing methods to abstract formal models appropriate for verification from existing models of hybrid control systems. Our hope is that the assessment of the current state of the tools in this paper, and the other papers in this invited session, will stimulate discussion of these issues and provide impetus for new research directions that will eventually make hybrid system verification a standard tool for control system design.

Acknowledgements

The authors thank P. Correa, T. Dang, and H. Wong-Toi for helping with the modeling and experimental results described in this paper. The authors also acknowledge the financial support by the National Science Foundation (NSF), the German Academic Exchange Council (DAAD), and the German Research Foundation (DFG). B.I. Silva is supported by a grant from CNPq/Brazil - 20.0079/92. The research of B.H. Krogh was also supported in part by Ford Motor Co., General Electric Company, and DARPA.

References

[1] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Comp. Science*, 126:183–235, 1994.

[2] R. Alur, T. Henzinger, F. Mang, S. Qadeer, S. Rajamani, and S. Tasiran. Mocha: Modularity in model checking. In *Proc. 10^{th} Int. Conf. on Computer-aided Verification*, LNCS 1427, pages 521–525. Springer, 1998.

[3] A. Bemporad and M. Morari. Verification of hybrid systems via mathematical programming. In *Hybrid Systems: Computation and Control*, LNCS 1569, pages 31–45. Springer, 1999.

[4] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, W. Yi, and C. Weise. New generation of UPPAAL. In *Proc. Int. Workshop on Software Tools for Technology Transfer*, 1998.

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

[6] K. Bournez, O. Maler, and A. Pnueli. Orthogonal polyhedra: Representation and computation. In *Hybrid Systems: Computation and Control*, LNCS 1596, pages 46–60. Springer, 1999.

[7] E. M. Clarke, O. Grumberg, and D. E. Long. Model checking and abstraction. *ACM Trans. on Programming Languages and Systems*, 16(5):1512–1542, 1994.

[8] E. M. Clarke and R. P. Kurshan. Computer-aided verification. *IEEE Spectrum*, pages 61–67, June 1996.

[9] T. Dang and O. Maler. Reachability analysis via face lifting. In *Hybrid Systems: Computation and Control*, LNCS 1386, pages 96–109. Springer, 1998.

[10] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Hybrid Systems III*, LNCS 1066, pages 208–219. Springer, 1996.

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

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

[13] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1/2):110–122, 1997.

[14] G. Lafferriere, G. J. Pappas, and S. Yovine. A new class of decidable hybrid systems. In *Hybrid Systems: Computation and Control*, LNCS 1596, pages 137–151. Springer, 1999.

[15] N. Lynch and B. H. Krogh, editors. *Hybrid Systems: Computation and Control.* LNCS 1790. Springer, 2000.

[16] B. Silva and B. Krogh. Modeling and verification of sampled-data hybrid systems. In *Proc. 4^{th} Int. Conf. on Automation of Mixed Processes*, pages 237–242, 2000.

[17] B. Silva, K. Richeson, B. Krogh, and A. Chutinan. Modeling and verifying hybrid dynamic systems using checkmate. In *Proc. 4^{th} Int. Conf. on Automation of Mixed Processes*, pages 323–328, 2000.

[18] O. Stursberg, S. Kowalewski, and S. Engell. On the generation of timed discrete approximations for continuous systems. *Math. and Comp. Modelling of Dynamical Systems*, 6(1):51–70, 2000.

[19] O. Stursberg, S. Kowalewski, J. Preussig, and H. Treseler. Block-diagram based modelling and analysis of hybrid processes under discrete control. *Journal Europ. des Syst. Automatises*, 32(9-10):1097–1118, 1998.

[20] F. W. Vaandrager and J. H. van Schuppen, editors. *Hybrid Systems: Computation and Control.* LNCS 1596. Springer, 1999.