Symbolic Verification of Hybrid Systems supported by Mathematical Services *

Christoph Benzmüller[1], Corrado Giromini[1], and Andreas Nonnengart[2]

[1] Fachbereich Informatik
Universität des Saarlandes, Saarbrücken, Germany
{chris,corrado}@ags.uni-sb.de

[2] German Research Center for Artificial Intelligence (DFKI)
Saarbrücken, Germany
nonnenga@dfki.de

Abstract. Verification of hybrid systems is a challenging task. Unlike many other verification methods the approach proposed in [21] avoids approximations and operates directly on the original system specifications. This approach, however, requires the solution of non-trivial mathematical subtasks. We propose to model those kinds of subtasks that are suitable for being tackled with reasoning specialists as mathematical service requests to a network of service systems like the MathWeb Software Bus [10]. So far we have identified the following candidates of mathematical service requests: the solution of differential equations, subsumption and consistency of sets of constraints. A further candidate can be the elimination of second order set variables with higher order theorem provers.

1 Introduction

The verification of hybrid systems has become an important issue in recent years. This is since hybrid systems today already control much of what we depend on our daily lives and the range of new applications is rapidly growing. Many hybrid systems, such as automated highway systems [17], air traffic management systems [18, 19], embedded automotive controllers [6, 23], manufacturing systems [24], chemical processes [9], are safety critical applications and require guarantees of safe operation. Since traditional program verification methods allow, at best, to approximate continuously changing environments by discrete sampling, special verification methods for hybrid systems, e.g. [14–16], have been developed. A common method is to model hybrid systems by hybrid automata. Hybrid systems consist of a discrete program with an analog environment [13]. We assume that a run of a hybrid system is modelled as a sequence of steps of a hybrid automaton. Within each step the system state evolves continuously according to a dynamical law until a transition occurs. Transitions are instantaneous state changes that separate continuous state evolutions.

* This work is supported by the EU training network CALCULEMUS (HPRN-CT-2000-00102) funded in the EU 5th framework.

A paradigmatic example of an hybrid system is a digital controller of an analog plant [13]. The discrete state of the controller is modelled by the vertices of a graph (control modes), and the discrete dynamics of the controller is modelled by the edges of the graph (control switches). The continuous state of the plant is modelled by points in $\mathbb{R}^n$, and the continuous dynamics of the plant is modelled by the flow conditions such as differential equations. The behaviour of the plant depends on the state of the controller: each control switch may cause a discrete change in the state of the plant, as determined by a jump condition. Dually the behaviour of the controller depends on the state of the plant: each control mode continuously observes an invariant condition of the plant state and by violating the invariant condition, a continuous change in the plant state will cause a control switch.

The verification method we will employ in our work is the "deductive" model checking approach for hybrid systems as presented in [21]. Given a specification of a hybrid system H (a hybrid automaton) and a safety property Φ this approach first generates a second order formula $\lceil \Phi \rceil_H$ such that the validity of the latter guarantees that property Φ is valid for H. To support the validation of $\lceil \Phi \rceil_H$ Nonnengart employs the elimination method. This method eliminates second order location predicates in $\lceil \Phi \rceil_H$ one by one in order to transform $\lceil \Phi \rceil_H$ into an equivalent first order formula Ψ, if possible. With the validation of Ψ the deductive model checking approach terminates.

In the context of this hybrid system verification approach we have identified the following mathematical subtasks: (1) The solution of sets of differential equations, (2) checking subsumption between sets of constraints, and (3) checking consistency of sets of constraints. Additionally we can try to (4) tackle the elimination of second order variables directly with an higher order theorem prover.

In case of linear constraints and linear differential equations the tasks (1.) – (3.) are reasonably computable. Our aim, however, is to widen the spectrum of the approach, for instance, by allowing also non-linear constraints and differential equations. One task therefore is to identify suitable mathematical service systems that could support respective requests.

Mathematical services like (1.) – (3.) may also be relevant for other hybrid system verification approaches. For instance, [11] employed the computer algebra system MATHEMATICA to solve linear constraints, later it replaced MATHEMATICA by a more efficient implementation of a specialised constraint solving algorithm [12]. However, multiple implementations of the same kinds of mathematical services in different verification systems could and should best be avoided, especially if their realization is complex and challenging, such as in our context.

2 A deductive model checking approach

We summarise the deductive model checking approach described in [21] and identify the subtasks we want to model as mathematical service requests.

2.1 Modelling of Hybrid Systems

Hybrid systems are modelled as hybrid automata, which are presented as finite graphs whose nodes correspond to global states (locations). An example of an hybrid automaton is given in Fig. 1.

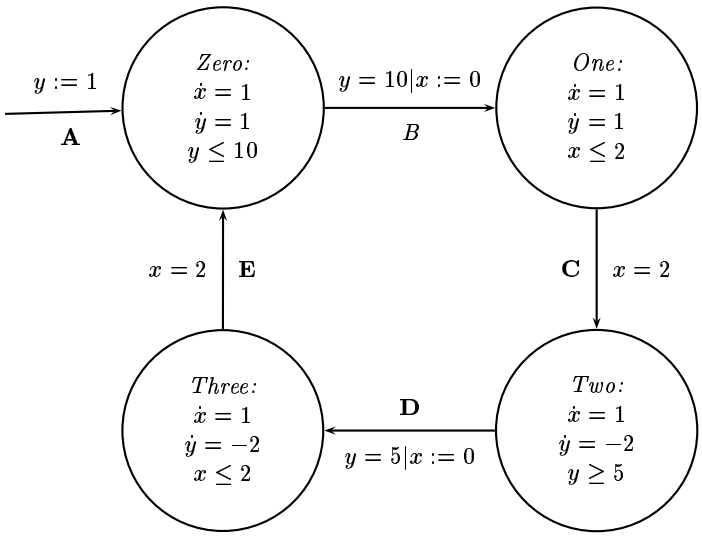

Fig. 1. *Water level monitor:* the water level in a tank is controlled through a monitor, which continuously senses the water level and turns a pump on and off. We wish to keep the water level (denoted by the data variable y) always between 1 and 12. The figure describes the monitor that signals whenever the water level passes 5 and 10 inches. x is the clock of the system that describes the delay of the pump startup/shutdown. We have to prove the ICTL property $AG(1 \leq y \wedge y \leq 12)$.

Formally this automaton is a tuple $(\mathrm{X}, \mathrm{L}, \mathrm{E}, \mathit{dif}, \mathit{inv}, \mathit{guard}, \mathit{act})$, where $\mathrm{X} = \{x, y\}$ is a set of *real valued data variables*, $\mathrm{L} = \{Zero, One, Two, Three\}$ a set of *locations* (nodes of the graph), $\mathrm{E} = \{A, B, C, D, E\}$ is a set of *transitions* (edges of the graph), and *dif*, *inv*, *guard*, and *act* are *mappings*. $\mathit{dif} : \mathrm{L} \times \mathrm{X} \to CT$ maps pairs of locations and data variables to constraint terms (CT); this mapping represents the *change of data variables over time* within locations, e.g. $\mathit{dif}(Three, x) = 1$ and $\mathit{dif}(Three, y) = -2$. $\mathit{inv} : \mathrm{L} \to CF$ maps each location to a constraint formula representing the *location invariant*, e.g. $\mathit{inv}(Zero) = (y \leq 10)$. $\mathit{guard} : \mathrm{E} \to CF$ describes the *transition condition* for the edges, e.g. $\mathit{guard}(B) = (y = 10)$. Finally, $\mathit{act} : \mathrm{E} \times \mathrm{X} \to CT$ specifies the *value assignment* for the data variables within the transitions, e.g. $\mathit{act}(B, x) = 0$. The set of *constraint terms* CT is build over the set X, the set of real-valued constants from $\mathbb{R}$, and the operations $+$ and $*$.[1] *Constraint formulas* have the form $\bot$, $\top$, or

[1] construction limited

$t \# t'$, where $t, t' \in CT$ and $\# \in \{<, >, \leq, \geq, =\}$
A state σ of a hybrid system is a pair (L, ϕ), with $L \in$ L and where ϕ is a valuation function for the terms and formulas in CT and CF. (L, ϕ) is *admissible* if $\phi(inv(L))$ holds. For instance, given a ϕ with $\phi(y) = 5$ then (One, ϕ) is admissible.

a logical representation of all immediate future states that can be reached from a given state. For instance, the local reachability theory for state Two is given as the first order formula

$$\forall x, y. Two(x, y) \rightarrow \begin{cases} y \geq 5 \wedge \\ \forall \delta \ \delta \geq 0 \wedge y - 2\delta \geq 5 \rightarrow Two(x + \delta, y - 2\delta) \wedge \\ y = 5 \rightarrow Three(0, y) \end{cases}$$

Here the possible immediate state transitions are characterised by the conjunct $y = 5 \rightarrow Three(0, y)$. In our example, $Three(0, y)$ is the only potential immediate future location of $Two(x, y)$ and $y = 5$ is a guard for the transition from Two to $Three$. In case of a transition data variable x is re-initialised with 0. The first conjunct tells us that any valid state $Two(x, y)$ has to satisfy the location invariant $y \geq 5$. Finally, the second conjunct characterises the potential timed successors, i.e., the change of the data variables x and y in case no transition to a successor state occurs. Similarly, an inevitability theory is introduced which characterises all states that are inevitable.

For the construction of local reachability theory for L it is generally needed to solve the differential equations characterising the timed successors of L. In our example we only consider very simple linear differential equations like $\dot{x} = 1$ and $\dot{y} = -2$ in state Two. They express that the value of x continuously increases over time with slope 1 and that the value of y continuously decreases over time with slope 2. The corresponding solutions for the differential equations are $x(\delta) = \delta + c$ and $y(\delta) = -2\delta + d$. Note that respective information is required for the formulation of the formula $\forall \delta \ \delta \geq 0 \wedge y - 2\delta \geq 5 \rightarrow Two(x + \delta, y - 2\delta)$ characterising the timed successors. If we extend the approach to handle also sets of non-linear differential equations, such as $\{\dot{x} = x + y, \dot{y} = 3\}$ or $\{\dot{x} = \sqrt{x + 3}, \dot{y} = 1, \ddot{x} = y * x\}$, the construction of the timed successor formulas can become a complicated task.

Mathematical service 1: Solving of differential equations To support the construction of the timed successor formulas we propose to identify suitable mathematical service systems for solving sets of differential equations. Our interest is to widen the current spectrum of the approach, for instance, by extending it to non-linear differential equations or second derivatives in the specification of hybrid systems.

2.2 Proving Properties of Hybrid Systems

Properties of hybrid systems are described by formulas of the *Integrator Computation Tree Logic ICTL* [1]. *ICTL* formulas are defined as the smallest set

such that: (i) $CF \subseteq ICTL$ and $L \subseteq ICTL$, (ii) if $\Phi, \Psi \in ICTL$ then $\neg\Phi \in ICTL$, and $\Phi \# \Psi \in ICTL$ for $\# \in \{\vee, \wedge, \rightarrow, \equiv\}$, (iii) if $\Phi, \Psi \in ICTL$ then $\#\Phi \in ICTL$ and $\Phi \#' \Psi \in ICTL$ for $\# \in \{AG, AF, EG, EF\}$ and $\#' \in \{EU, AU\}$, and (iv) if $\Phi \in ICTL$, z is new for X, and $\{L_1, \ldots, L_n\} \subseteq$ L then $z^{\{L_1,\ldots,L_n\}} \cdot \Phi \in ICTL$ (and z is added to X). Intuitively, the temporal operators AG, AF, EG, EF, EU, AU, mean "always", "inevitably", "possibly always", "possibly", "possibly until" and "inevitably until" respectively. An example property for our hybrid in Fig 1 is $\Phi = AG(1 \leq y \wedge y \leq 12)$.
In the context of our work we are interested in extending the definition of $ICTL$ formulas such that they may also contain $\sqrt{T}$, $sin(T)$, $cos(T)$, $tan(T)$, $cotan(T)$, and ${T_1}^{T_2}$ for $ICTL$ terms $T_1, ..., T_i$. Given an hybrid system H, e.g the system described in Fig 1, and a property $\Phi \in ICTL$, e.g. $\Phi = AG(1 \leq y \wedge y \leq 12)$. We are interested to analyse the validity of Φ for H. $\hat{\phi} = \{\phi \mid H, (L, \phi) \models \Phi\}$ describes the set of all valuations ϕ where property Φ is true. Suppose $\hat{\phi}$ can be described by a (finite) characteristic constraint formula $\lceil\Phi\rceil_H^L \in CF$. Intuitively a characteristic constraint formula describes the necessary and sufficient conditions on the data variables such that Φ holds for L in H. Checking whether $H, (L, \phi) \models \Phi$ holds can be reformulated as to checking whether $\phi(\lceil\Phi\rceil_H^L)$ is valid. Generally, it is not possible to find a characteristic constraint formula. Instead of attempting to construct $\lceil\Phi\rceil_H^L$ directly, it is described first as a formula of the second order predicate calculus. Then the elimination approach tries to simplify this description to a first order constraint formula step by step, if possible.

In our example we are interested to verify property $\Phi = AG(1 \leq y \wedge y \leq 12)$ for automaton H sketched in Fig. 1. This means we want to verify that $H, (Zero, \phi) \models AG(1 \leq y \wedge y \leq 12)$ for all initial states $(Zero, \phi)$ and this reduces to checking whether $\phi(\lceil AG(1 \leq y \wedge y \leq 12)\rceil_H^{Zero})$ for all ϕ. According to Nonnengart's approach $\lceil AG(1 \leq y \wedge y \leq 12)\rceil_H^{Zero}$ is given as the following second order formula:

$$
\exists \begin{array}{l} Zero \\ One \\ Two \\ Three \end{array}
\left(\begin{array}{ll}
Zero(x,1) & \\
\forall x,y\ Zero(x,y) & \rightarrow \left\{\begin{array}{l} y \leq 10 \\ \forall\delta\ \delta \geq 0 \wedge y + \delta \leq 10 \rightarrow Zero(x+\delta, y+\delta) \\ y = 10 \rightarrow One(0,y) \end{array}\right. \\
\forall x,y\ One(x,y) & \rightarrow \left\{\begin{array}{l} x \leq 2 \\ \forall\delta\ \delta \geq 0 \wedge x + \delta \leq 2 \rightarrow One(x+\delta, y+\delta) \\ x = 2 \rightarrow Two(x,y) \end{array}\right. \\
\forall x,y\ Two(x,y) & \rightarrow \left\{\begin{array}{l} y \geq 5 \\ \forall\delta\ \delta \geq 0 \wedge y - 2\delta \geq 5 \rightarrow Two(x+\delta, y-2\delta) \\ y = 5 \rightarrow Three(0,y) \end{array}\right. \\
\forall x,y\ Three(x,y) & \rightarrow \left\{\begin{array}{l} x \leq 2 \\ \forall\delta\ \delta \geq 0 \wedge x + \delta \leq 2 \rightarrow Three(x+\delta, y-2\delta) \\ x = 2 \rightarrow Zero(x,y) \end{array}\right. \\
\forall x,y\ Zero(x,y) & \rightarrow 1 \leq y \wedge y \leq 12 \\
\forall x,y\ One(x,y) & \rightarrow 1 \leq y \wedge y \leq 12 \\
\forall x,y\ Two(x,y) & \rightarrow 1 \leq y \wedge y \leq 12 \\
\forall x,y\ Three(x,y) & \rightarrow 1 \leq y \wedge y \leq 12
\end{array}\right)
$$

Here $Zero(x,1)$ characterises the initial state. The next four conjuncts specify the reachability theory for H composed from the local reachability theories for all states as described before. The last four conjuncts finally guarantee the validity of property $(1 \leq y \wedge y \leq 12)$ within all (reachable) states. This formula is equivalent to

$$\exists \begin{array}{l} Zero \\ One \\ Two \\ Three \end{array} \left(\begin{array}{ll} Zero(x,1) & \\ \forall x,y\ Zero(x,y) & \rightarrow \left\{ \begin{array}{l} y \leq 10 \wedge 1 \leq y \wedge y \leq 12 \\ \forall \delta\ \delta \geq 0 \wedge y + \delta \leq 10 \rightarrow Zero(x+\delta, y+\delta) \\ y = 10 \rightarrow One(0,y) \end{array} \right. \\ \forall x,y\ One(x,y) & \rightarrow \left\{ \begin{array}{l} x \leq 2 \wedge 1 \leq y \wedge y \leq 12 \\ \forall \delta\ \delta \geq 0 \wedge x + \delta \leq 2 \rightarrow One(x+\delta, y+\delta) \\ x = 2 \rightarrow Two(x,y) \end{array} \right. \\ \forall x,y\ Two(x,y) & \rightarrow \left\{ \begin{array}{l} y \geq 5 \wedge 1 \leq y \wedge y \leq 12 \\ \forall \delta\ \delta \geq 0 \wedge y - 2\delta \geq 5 \rightarrow Two(x+\delta, y-2\delta) \\ y = 5 \rightarrow Three(0,y) \end{array} \right. \\ \forall x,y\ Three(x,y) & \rightarrow \left\{ \begin{array}{l} x \leq 2 \wedge 1 \leq y \wedge y \leq 12 \\ \forall \delta\ \delta \geq 0 \wedge x + \delta \leq 2 \rightarrow Three(x+\delta, y-2\delta) \\ x = 2 \rightarrow Zero(x,y) \end{array} \right. \end{array} \right)$$

where we have exactly one location predicate for each state.

2.3 The Elimination Approach

The remaining problem consists in proving the validity of the second order formula $\lceil AG(1 \leq y \wedge y \leq 12) \rceil_H^{Zero}$ above. Nonnengart proposes a stepwise transformation of these kinds of formulas to equivalent first order statements. This is done by a one by one elimination of location predicates in $\lceil AG(1 \leq y \wedge y \leq 12) \rceil_H^{Zero}$ based on the elimination theorem [22]. General applications of the elimination theorem require the evaluation of a greatest fixpoint for the location variable to be evaluated. For special applications, however, the elimination theorem can be refined in a much simpler simplification lemma which does not require such complicated computations. The special cases are those where a state L does not have direct edge transitions to itself. This is given for our automaton H. The application of the simplification lemma leads to the equivalent automaton H' depicted in Fig. 2. Two further applications of the simplification lemma lead to the equivalent one state automaton H'' depicted in Fig. 3.

Given that we can validate $\lceil AG(1 \leq y \wedge y \leq 12) \rceil_{H''}^{Zero}$ we are done. Thus it remains to prove

$$\exists\ Zero \left(\begin{array}{ll} Zero(x,1) & \\ \forall x,y\ Zero(x,y) & \rightarrow \left\{ \begin{array}{l} y \leq 10 \wedge 1 \leq y \wedge y \leq 12 \\ \forall \delta\ \delta \geq 0 \wedge y + \delta \leq 10 \rightarrow Zero(x+\delta, y+\delta) \\ x = 10 \rightarrow Zero(2,1) \end{array} \right. \end{array} \right)$$

Since we now face a self-loop the simplification lemma is no longer applicable and we have to proceed with the elimination theorem where a greatest fixpoint

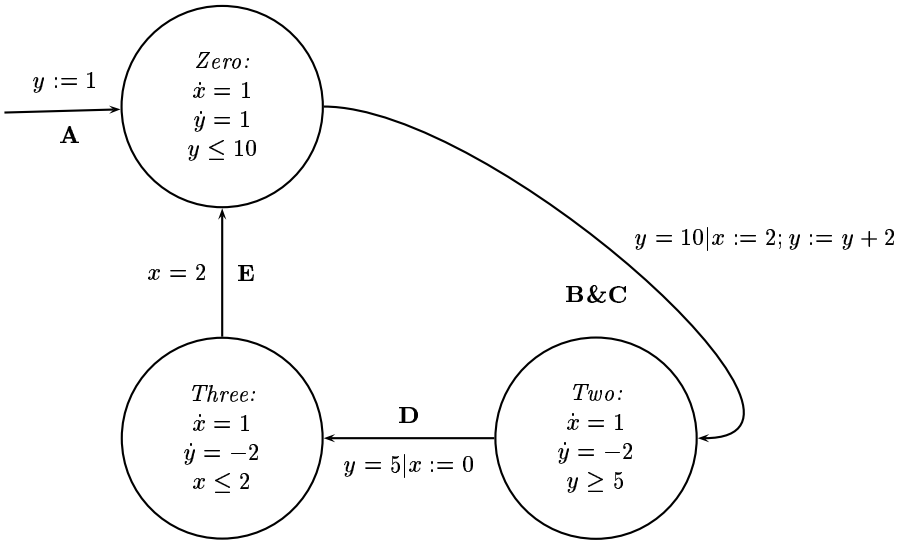

Fig. 2. Step 1: One's elimination leads to automaton H'

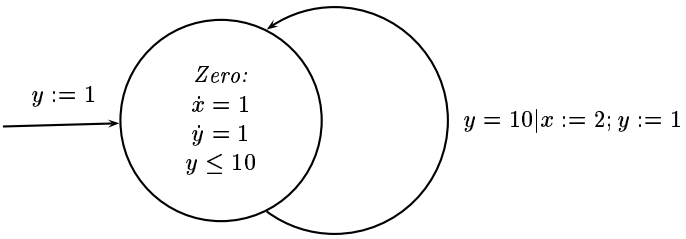

Fig. 3. Step 3: One state automaton H"

evaluation is generally required. In our special case the self-loop is directly subsumed by the initial state. In order to generally detect and evaluate fixpoints, however, we are interested to check whether a current iterative in the fixpoint computation is already subsumed by a previous iterative. The iterations thereby are typically growing compounds of first order constraint formulas like

$$y \leq 10 \wedge 1 \leq y \wedge y \leq 12$$

which is also the final greatest fixpoint to be evaluated in our example. As this constraint formula is consistent the deductive model checking approach tells us that property ϕ is indeed valid for H, hence, we are done.

Mathematical service 2: Subsumption of sets of constraints To support the fixpoint computations we are interested in subsumption checks for sets (or conjunctions) of constraints.

Mathematical service 3: Solving sets of constraints We are interested in the consistency of sets of constraints generated by the approach. Note that both tasks are relatively trivial in case of linear constraints. However, just as for the differential equations our interests is to widen the spectrum of the approach by, for instance, including non-linear constraints in the specification of hybrid systems. The challenge, however, will extend the elimination approach to handle also non-linear cases. This in turn poses a challenge for the requested mathematical services. Their strengths and weaknesses will determine how far we can go.

3 Higher Order Theorem Proving

As an alternative to the elimination method we propose to investigate whether higher order theorem provers can be employed to show the validity of the second order characteristic constraint formula. Examples of current higher theorem provers are TPS [2] and LEO [7]. The range of problems that can be handled by the TPS system is sketched in [3]. In our context the proof problems are of the form

$$\exists L.L(x_1, \ldots, x_n) \rightarrow (\ldots \wedge (\ldots \rightarrow L(t_1[x_1], \ldots, t_n[x_n])))$$

and we are interested in the synthesis of suitable formula instances for the second order location variable L. These kind of problems are still rather challenging for higher order theorem provers. The problem is that formula schemas for variables like L are typically guessed in recent approaches since general mechanisms to support a goal directed synthesis are still missing. Recently, however, some progress in this direction has been made in the TPS prover by Chad Brown [8]. It is thus a challenge to investigate whether current higher order theorem provers can be improved and adapted such that they can directly support the elimination of location variables in our approach. At least our examples can serve as an interesting test bed for this purpose.

Mathematical service 4: We want to examine whether higher order theorem provers can directly support the elimination of second order location variables in the deductive model checking approach.

4 The Work Plan

Our work plan is divided in three main sections.

Firstly, we analyse and characterise kinds of subtasks in Nonnengarts's deductive model checking approach that are suitable for being treated as mathematical service requests. We also need to clarify which extensions in the specification of hybrid systems are most useful for practical purposes and then identify their precise impact to the kinds of identified mathematical subtasks.

Secondly, we want to model the kinds of mathematical subtasks as mathematical service requests to networks of mathematical service systems like MATHWEB [10] or LBA [5]. Here, we want to closely cooperate with the researchers of the CALCULEMUS initiative (`http://www.eurice.de/calculemus`) that are interested in service descriptions.

Thirdly, we need to identify suitable mathematical service systems, such as Constraint Solvers (e.g. [20]), Computer Algebra Systems (e.g. [25]), Theorem Provers (e.g. [7]), and Rewrite Systems (e.g. [4]), that can handle our mathematical subtasks. Probably even schematic cooperations of such systems will be required. Given that we can identify appropriate systems the challenge will be to integrate them into the respective network of mathematical services such that they can actively tackle the mathematical service requests.

Finally we want to apply the approach to industrial-strength examples and evaluate it against alternative approaches.

References

1. R. Alur, T. A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. In *IEEE Real-Time Systems Symposium*, pages 2–11, 1993.
2. P. B. Andrews, M. Bishop, and C. E. Brown. System description: TPS: A theorem proving system for type theory. In *Conference on Automated Deduction*, pages 164–169, 2000.
3. P. B. Andrews, M. Bishop, S. Issar, D. Nesmith, F. Pfenning, and H. Xi. TPS: A theorem proving system for classical type theory. *Journal of Automated Reasoning*, 16(3):321–353, 1996.
4. A. Armando, L. Compagna, and S. Ranise. System description: Rdl–rewrite and decision procedure laboratory. In *International Joint Conference on Automated Reasoning (IJCAR2001)*, 2001.
5. A. Armando and D. Zini. Interfacing computer algebra and deduction systems via the logic broker architecture. In *Eight Symposium on the Integration of Symbolic Computation and Mechanized Reasoning (CALCULEMUS-2000)*, 2000.
6. A. Balluchi, M. Di Benedetto, C. Pinello, C. Rossi, and A. Sangiovanni-Vincentelli. Hybrid control in automotive applications: the cut-off control, 1999.

7. C. Benzmüller and M. Kohlhase. LEO — a higher-order theorem prover. In Claude Kirchner and Hélène Kirchner, editors, *Proceedings of the 15th Conference on Automated Deduction*, number 1421 in LNAI, pages 139–144. Springer Verlag, 1998.
8. C. E. Brown. Solving for set variables in higher-order theorem proving. Unpublished document; submitted.
9. S. Engell, S. Kowalewski, C. Schulz, and O. Stursberg. analysis and optimization of continuous-discrete interactions in chemical processing plants.
10. A. Franke, S. M. Hess, G. Jung, Ch. , M. Kohlhase, and V. Sorge. Agent-oriented integration of distributed mathematical services. *J.UCS: Journal of Universal Computer Science*, 5(3):156–??, 1999.
11. T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1-2):110–122, 1997.
12. T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. Beyond HYTECH: Hybrid systems analysis using interval numerical methods. In *HSCC*, pages 130–144, 2000.
13. T.A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science*, pages 278–292. IEEE Computer Society Press, 1996.
14. T.A. Henzinger and P.H. Ho. HyTech: The Cornell Hybrid Technology Tool. In P. Antsaklis, A. Nerode, W. Kohn, and S. Sastry, editors, *Hybrid Systems II*, Lecture Notes in Computer Science 999, pages 265–293. Springer-Verlag, 1995.
15. T.A. Henzinger and P.H. Ho. A note on abstract-interpretation strategies for hybrid automata. In P. Antsaklis, A. Nerode, W. Kohn, and S. Sastry, editors, *Hybrid Systems II*, Lecture Notes in Computer Science 999, pages 252–264. Springer-Verlag, 1995.
16. T.A. Henzinger and H. Wong-Toi. Linear phase-portrait approximations for nonlinear hybrid systems. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, Lecture Notes in Computer Science 1066, pages 377–388. Springer-Verlag, 1996.
17. R. Horowitz and P. Varaiya. Design of an automated highway system.
18. C. Livadas, J. Lygeros, and N. Lynch. High-level modelling and analysis of tcas, 1999.
19. J. Lygeros, G. J. Pappas, and S. Sastry. An approach to the verification of the center-TRACON automation system. In *HSCC*, pages 289–304, 1998.
20. E. Melis, J. Zimmer, and T. Müller. Integrating constraint solving into proof planning. In H. Kirchner and C. Ringeissen, editors, *Frontiers of Combining Systems – Third International Workshop, FroCoS 2000*, volume 1794 of *LNAI*, pages 32–46, Nancy, France, March 2000. Springer.
21. A. Nonnengart. A deductive model checking approach for hybrid systems. Technical Report MPI-I-1999-2-006, Max-Planck-Institute for Computer Science, Saarbrücken, Germany, November 1999. Available via http://www.mpi-sb.mpg.de/.
22. A. Nonnengart and A. Szalas. A fixpoint approach to second-order quantifier elimination with applications to correspondence theory. Technical Report MPI-I-95-2-007, MPII Saarbrücken, Saarbrücken, Germany, 1995.
23. A. Ohata, E. Gassenfeit, K. Butts, and B. Krogh. Automotive engine systems: A need for hybrid systems simulation and analysis.
24. D. Pepyne and C. Cassandras. Hybrid systems in manufacturing, 2000.
25. Darren Redfern. *The Maple Handbook: Maple V Release 4*. Springer, 1996.