Reasoning Services in the MathWeb-SB for symbolic verification of Hybrid Systems *

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

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

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

Abstract. Verification of non-linear hybrid systems is a challenging task. Unlike many other verification methods the deduction-based verification approach we investigate in this paper avoids approximations and operates directly on the original non-linear system specifications. This approach, however, requires the solution of non-trivial mathematical subtasks. We propose to model existing reasoning systems, such as computer algebra systems and constraint solvers, as mathematical services and to provide them in a network of mathematical tools in a way that they can reasonably support subtasks as they may occur in formal methods applications. The motivation is to make it simpler to implement and test verification approaches by out-sourcing complex but precisely identifiable mathematical subtasks for which specialised reasoners do already exists.

1 Introduction

Hybrid systems are heterogenous dynamical systems characterised by interacting continuous and discrete dynamics. The enormous presence of hybrid systems in safety critical applications, such as automated highway systems [18], air traffic management systems [22], embedded automotive controllers [3], and chemical processes [9], increasingly calls for safety guarantees. Since traditional program verification methods allow at best to approximate continuously changing environments by discrete sampling, special verification methods for hybrid systems, such as [15–17], have been developed. A frequently employed method is to model hybrid systems by hybrid automata. A hybrid automaton is a closed system with a *built-in* control structure determining when and how the system switches

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

between its various discrete states. Thereby the continuous behaviour in each discrete state is governed by a differential equation.

The verification method we will employ in our work is the deduction-based model checking approach for hybrid systems described in [23]. Given a specification of a hybrid system H (a hybrid automaton) and a safety property Φ the approach 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 $\lceil \Phi \rceil_H$ 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 verification approach terminates.

For the above deduction-based model checking 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.

In general, solving these tasks is feasible in case of linear constraints and linear differential equations. Our aim, however, is to widen the spectrum of the approach, for instance, by allowing also non-linear constraints and differential equations. Mathematical tasks like (1) – (3) may also be relevant for other hybrid system verification approaches. For instance, [13] employed the computer algebra system MATHEMATICA to solve linear constraints. MATHEMATICA was later replaced by a more efficient implementation of a specialised constraint solving algorithm [14]. 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.

We propose to model existing reasoning systems, such as computer algebra systems and constraint solvers, as mathematical services and to provide them in a network of mathematical tools in a way that they can reasonably support subtasks as they may occur in formal methods applications. The motivation is to make it simpler to implement and test verification approaches by out-sourcing complex but precisely identifiable mathematical subtasks for which specialised reasoners do already exists. Allowedly, in case a verification approach later turns out to be successful (see for instance [14]) it may be reasonable from efficiency aspects and also from concession aspects to replace the connections to mathematical services again by fast re-implementations of the particularly needed algorithms. However, starting with the latter may dramatically slow down a quick development and implementation of new verification environments. This is particularly true in case the automation of the mathematical subtasks is already on the edge of current research, such as given in our case. This motivates our proposal to build up a network of mathematical reasoning services for formal methods. The more services will be appropriately added to such a network the more likely it will be that also other verification approaches can directly employ them (in early development stages) for the same purpose.

In this paper we illustrate the kinds of mathematical subtasks which occur in the verification approach [23] by looking at a simple non-linear hybrid system.

Our network of choice for providing the mathematical services is the mathematical software bus MathWeb-SB [20].

The outline of the paper is as follows: we first illustrate aspects of the deduction-based elimination approach for hybrid system verification and motivate different kinds of mathematical subtasks involved. We then sketch the mathematical software bus MathWeb-SB that is our mathematical network infrastructure of choice and, furthermore, discuss candidate systems suitable for tackling the identified kinds of problems. Finally we give an outlook on some first ideas and requirements on the modelling and solving of these problems in the MathWeb-SB.

2 Mathematical Subtasks in Hybrid System Verification

We briefly sketch the deduction-based model checking approach (DMC) of Nonnengart described in [23], starting from the description of a hybrid system. We also identify some of the mathematical subtasks that result from the application of this approach. Our presentation follows a very simple example of a non-linear hybrid system for which the deduction-based model checking approach results in non-linear constraints.

2.1 Structure of a Hybrid System

Hybrid systems are typically modelled as hybrid automata that are presented as finite graphs whose nodes correspond to global states (locations). The discrete dynamics, i.e. the state transitions, of the automaton is modelled by the edges of the graph. The continuous dynamics of the automaton is modelled by differential equations associated with each state.

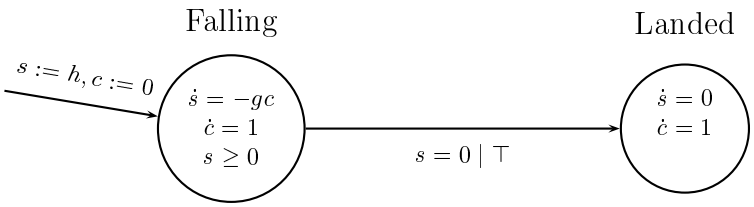

Fig. 1. A simple hybrid automaton

Fig. 1 shows a simple hybrid automaton modelling Galilei's gravity test: Given a tower of height h, we assume to stand on top of this tower and let a stone fall down. The stone falls until it reaches the bottom. Hence, we say, the stone can be in two main states: `Falling` and `Landed`. At the beginning of our

experiment the clock counter c, which mirrors the flow of time t in our system, is set to 0, and we are dropping the stone from height h. Thus, the height of the falling stone (at time t), which is represented by variable s, is initialised with h.

While the stone falls it accelerates according to the physical law of gravity:

$$\dot{s}(t) = -gt, \tag{1}$$

where g is the gravitational constant ($g = 9.81 m/sec^2$). This is represented as the invariant $\dot{s} = -gc$ in the state `Falling`. Another invariant of `Falling` is $\dot{c} = 1$ expressing that the clock c is increasing linearly. $s \geq 0$, the last invariant condition, says that the stone has not reached the ground yet. The condition for a state transition to `Landed` simply is $s = 0$.

2.2 The Deduction-Based Model Checking Approach

We now apply the DMC to our example automaton in Fig. 1. Suppose we want to know when the stone falling from the tower will reach the ground level. In terms of a *Integrator Computation Tree Logic* (ICTL) [23] formula this means to prove the property:

$$\exists\Diamond \ (Landed \wedge c \leq k), \tag{2}$$

where k is a parameter to be instantiated by the proof procedure. A first task in the DMC approach is the formalisation of the reachability theory for our automaton. This theory formalises the conditions and invariants for staying in the states `Falling` and `Landed` as well as the conditions causing a transition from state `Falling` to `Landed`.

For the formalisation of this theory as a second order formula we, for instance, have to solve the differential equation (1). Taking into consideration the initialisation information we get:

$$c(t) = t, \ \ s(t) = h - \tfrac{1}{2}gt^2. \tag{3}$$

and thus

$$c(t+\delta) = c(t) + \delta, \ \ s(t+\delta) = s(t) - gc(t)\delta^2 - \tfrac{1}{2}g\delta^2 \tag{4}$$

for any change in time δ.

The complete reachability theory of the Galilei automaton is given as the following conjunctive set of second order formulas (F and L are second order variables)[1]:

$$F(s,c) \rightarrow \forall\delta. \left(\delta \geq 0 \wedge s' = s - gc\delta - \frac{1}{2}g\delta^2 \wedge c' = c + \delta \wedge s' \geq 0 \rightarrow F(s',c')\right)$$
$$F(s,c) \rightarrow s \geq 0$$
$$F(s,c) \rightarrow s = 0 \rightarrow L(s,c)$$
$$L(s,c) \rightarrow \forall\delta \ \ (\delta \geq 0 \wedge s' = s \wedge c' = c + \delta \rightarrow L(s',c'))$$

[1] Some of the formulas are already somewhat simplified.

$F(h,0)$ represents the initial state. Let us call this theory $\Re$. For the sake of simplicity we consider now the dual of the property (2), namely $\forall\Box(L \to c > k)$. Hence, we have to prove

$$\exists F, L \quad F(h,0) \wedge \Re \wedge \forall s, c\, L(s,c) \to \lceil L \to c > k \rceil \wedge F(s,c) \to \lceil L \to c > k \rceil$$

which simplifies to

$$\exists F, L \quad F(h,0) \wedge \Re \wedge \forall s, c\, L(s,c) \to c > k.$$

Let us first eliminate location F. We start with the fix-point computation over the state Falling.

$$\begin{array}{ll} \Gamma^0(T) = T & \\ \Gamma^1(T) = s \geq 0 \wedge s = 0 \to L(s,c) & \\ \Gamma^2(T) = \forall\delta.(\delta \geq 0 \wedge s' = s - gc\delta - \frac{1}{2}g\delta^2 \wedge c' = c + \delta \wedge s' \geq 0 \to & \\ \qquad s' \geq 0 \wedge s' = 0 \to L(s',c')) & (5) \\ \Gamma^3(T) = \forall\delta.(\delta \geq 0 \wedge s' = s - gc\delta - \frac{1}{2}g\delta^2 \wedge c' = c + \delta \wedge s' \geq 0 \to & \\ \qquad \forall\delta'.(\delta' \geq 0 \wedge s'' = s' - gc'\delta' - \frac{1}{2}g\delta'^2 \wedge c'' = c' + \delta' \wedge s'' \geq 0 \to & \\ \qquad s'' \geq 0 \wedge s'' = 0 \to L(s'',c'')) & \end{array}$$

It is easy to see that in Γ^3 the computation terminates (because $\Gamma^3 \to \Gamma^2$). Hence, with the insertion of the initial condition $F(h,0)$, the result is:

$$\forall\delta \ \left(\delta \geq 0 \wedge s' = h - \frac{1}{2}g\delta^2 \wedge c' = \delta \wedge s' \geq 0 \to c' > k \wedge s' = 0 \to L(s',c')\right),$$

where c' and s' are universally quantified variables. This can be simplified to

$$c' \geq 0 \wedge s' = h - \frac{1}{2}gc'^2 \wedge s' \geq 0 \to c' > k \wedge s' = 0 \to L(s',c').$$

Further simplification leads to

$$c' \geq 0 \wedge \sqrt{\frac{2h}{g}} \geq c' \to c' > k \wedge c' = \sqrt{\frac{2h}{g}} \to L(0,c') \qquad (6)$$

At this stage it would be necessary to eliminate L as well. In fact this is very simple and therefore is omitted here. From formula 6 we can extract a constraint on the variable k, namely: $k < \sqrt{\frac{2h}{g}}$. And since we had a look at the negation of the property to be proved, we finally end up with the result that the stone is landed for all values of the clock of at least $\sqrt{\frac{2h}{g}}$. Thus the moment of landing is exactly when $c = \sqrt{\frac{2h}{g}}$.

We now point to three relevant mathematical subtasks that occur in the context of the DMC approach:

(1) In the example we have to solve the differential equations $\dot{c}(t) = 1$ and $\dot{s}(t) = -gt$. The solution is employed in the formalisation of the reachability theory.

While this is trivial in our hybrid system, the solution of non-linear differential equations is generally complex and not easily computable.

(2) The DMC approach stepwise eliminates the second order state predicates and thereby generates sets of constraints. As indicated above a subsequent task is then to check the consistency of the generated constraint sets in order to show that a model exists. In our example above, for instance, we are interested in constraints like:

$$\delta \geq 0 \wedge s' = s - gc\delta - \tfrac{1}{2}g\delta^2 \wedge$$
$$c' = c + \delta \wedge s' \geq 0 \rightarrow c' > k$$

The constraint variables are c and s,while δ, c', s' universally quantified parameters coming from the reachability theory.

(3) Generally the fix-point computations involved in the DMC approach are not as trivial as in the example here. The detection of fix-points can then be supported by checking the subsumption of the constraint sets of single iterations in the fix-point computation.

The overall picture is that the verification tool, implementing the deduction-based verification approach, is processing the main steps involved. This, for instance, includes the formalisation of the reachability theory and the stepwise elimination of second order variables. The verification tool is also responsible for the generation and appropriate formulation of the concrete mathematical service requests illustrated in (1) – (3) and for passing them to the MathWeb-SB. A verification example that illustrates the work-sharing aspects of the sketched verification approach in more detail is given in [4]. As long as the verification tool is not fully implemented its tasks or parts of its tasks will be simulated by hand.

In the remainder of this paper we will concentrate on the mathematical service network MathWeb-SB, which is our network infrastructure of choice. We will also present candidate systems for supporting the mathematical subtasks we are interested in.

3 The MathWeb Software Bus

The MathWeb Software Bus (MathWeb-SB) [20] for distributed automated theorem proving supports the connection of a wide range of *mathematical services* by a common software bus. The MathWeb-SB provides the functionality to turn existing theorem proving systems, computer algebra systems, and miscellaneous tools into mathematical services that are homogeneously integrated into a proof development environment.

The MathWeb-SB is implemented in Mozart Oz [11], a multi-paradigm object-oriented programming language which fully supports concurrent and distributed programming and allows to simply distribute applications over the Internet. The services of the MathWeb-SB are used permanently by client applications, such as the ΩMEGA system [10]. The MathWeb-SB currently integrates many different reasoning and computation systems, like, for instance, automated theorem provers (e.g., Otter, Spass, etc.) and computer algebra systems (CASs) (e.g., Maple, and GAP). Fig. 2 shows parts of the MathWeb-SB as it is currently running. In the MathWeb-SB, *service servers* offer the mathematical services (e.g., an ATP, or a CAS) to their local MathWeb-SB broker. MathWeb brokers register and unregister to other brokers, so called *remote brokers*, running in the Internet and therefore build a dynamic web of brokers.

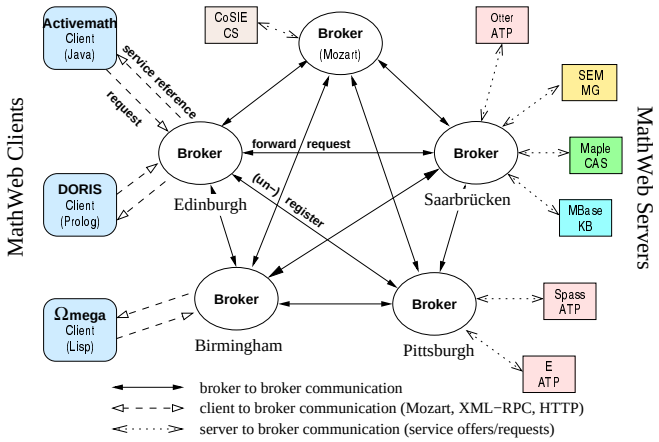

Fig. 2. The MathWeb Software Bus

Client applications, like the ΩMEGA system or a CGI-script, connect to one of the MathWeb-SB brokers and request services. If the requested service is not offered by a local server, the broker forwards the request to all remote brokers. If the requested service is found, the client application receives a reference to a newly created *service object* and can directly send messages to the object. The MathWeb-SB currently offers three interfaces to connect to a broker, namely Mozart's distributed programming interface, CGI-script access via an HTTP server, and access via XML-RPC.

3.1 Solving Differential Equations

Amongst the available Computer Algebra Systems in the MathWeb-SB we choose Maple [6] as a first candidate to support our tasks. Maple is a mathematical problem-solving environment that supports a wide variety of mathematical operations such as numerical analysis, symbolic algebra, and graphics. We intend

to use the module for differential equations, especially the `dsolve` function, that is a part of MAPLE. This module has the capability to solve Ordinary Differential Equations (ODEs). The `dsolve` function is good in solving linear differential equations very efficiently and provides good results, but it is quite weak in attacking non-linear systems. A better candidate for the non-linear case is the module NODES (Non linear Ordinary Differential Equations Solver) [8], which has been developed in context of the European ESPRIT contact group CATHODE. The NODES module is implemented in the MAPLE programming language and is specialised in the analysis of systems of non-linear ODEs. It is based on the quasi-monomial transformation theory [24]. In that theory, a system of ODEs is represented by a couple of matrices and only these are manipulated. NODES identifies values of the parameters in the relative matrix representation corresponding to the integrability property of the ODE system and builds the associated first integrals.

3.2 Checking Consistency of Constraints

The Rewrite and Decision procedure Laboratory (RDL) [1] simplifies clauses in a quantifier-free first-order logic with equality using a tight integration between rewriting and decision procedures. RDL is based on CCR (Constraint Contextual Rewriting) [2], a formally specified integration schema between (ordered) conditional rewriting and a satisfiability decision procedure. As a consequence, RDL is sound, terminating and fully automatic.

RDL is an open system which can be modularly extended with new decision procedures provided these offer certain interface functionalities. In its current version, RDL offers *'plug and play'* decision procedures for the theories of Universal Presburger Arithmetic over Integers (UPAI), Universal Theory of Equality (UTE), and UPAI extended with uninterpreted function symbols. Last but not least, RDL implements instances of a generic extension schema for decision procedures. The key ingredient of such a schema is a lemma speculation mechanism which 'reduces' the satisfiability problem of a given theory to the satisfiability problem of one of its sub-theories for which a decision procedure is available. In the following we explain in few words how the lemma speculation works on constraint sets.

In the context of our subtask, subsumption and checking of constraints can be attacked in two ways. If the set of constraints looks like a set of polynomial or trigonometric functions we can simplify them using highly efficient arithmetic libraries. Due to this, we can handle constraints containing trigonometric functions such as $\sin(x)$, $\cos(x)$, etc. In case that arithmetic is unable to simplify the constraint set, we can attack the problem using the quantifier elimination approach. The mechanism that allows RDL to decide which is the best choice for the solving of the problem is Lemma Speculation. Here we sketch briefly how this mechanism works for constraint subsumption.

Lemma speculation. The goal of this mechanism is to feed the decision procedure with new facts about function symbols which are otherwise uninterpreted

in the theory T decided by the decision procedure. In other words, it inspects the context C and returns a set of ground facts entailed by C using T as the background theory. In RDL there are three kinds of lemma speculation: the simplest is `augment` that finds instances of the conclusions among the conditional lemmas which can promote further inference steps in the decision procedure; an improvement of the `augment` method is `affinize` that implements the 'on the fly' generation of lemmas about multiplication over integers. Affinization is particularly useful for non-linear inequalities and doesn't require any user intervention. As most powerful choice, there is the combination of the two mechanisms mentioned above. RDL combines augmentation and affinization by considering the function symbols occurring in the context C. For example, the top-most function symbol of the largest literal in C triggers the invocation of either mechanisms.

4 Work Plan

We will investigate whether the sketched approach is applicable to industrial-strength examples. To gain evidence for this we want to pursue case studies like air traffic management [21, 22] and the steam-boiler problem [12]. Starting with an appropriate representation of the automata we will apply the DMC approach and identify the concrete instances of the mathematical subtasks described above. We then analyse whether and how these subtasks can actually be attacked by the systems already available in the MathWeb-SB. The aim then is to suitably model the subtasks as service requests to the MathWeb-SB. We might possibly have to integrate new systems into the MathWeb-SB like, for instance, the RDL system.

In the current implementation of the MathWeb-SB, the services requested by client applications are whole reasoning systems (e.g. the CAS Maple or ATPs such as Otter). The service objects offer interface methods for using the system's reasoning capabilities, for instance the method `eval` in the case of CASs or `prove` in the case of ATPs. For our work, we have to extend the MathWeb-SB such that also abstract reasoning services, e.g. solving differential equations, can be defined, offered to MathWeb brokers, and requested by clients. We plan an implementation of abstract reasoning service as new interface methods of the services objects. We also intend to use a service description language to describe reasoning services independent of a concrete implementation. This language will be based on XML-standards WSDL [7], OpenMath [5], and OMDoc [19]. Abstract service descriptions can then be mapped to interface method calls.

References

1. A. Armando, L. Compagna, and S. Ranise. System description: Rdl–rewrite and decision procedure laboratory. In *International Joint Conference on Automated Reasoning (IJCAR2001)*, 2001.
2. A. Armando and S. Ranise. Constraint contextual rewriting. In R. Caferra and G. Salzer, editors, *Proceedings of the 2nd International Workshop on First Order Theorem Proving, FTP'98, Vienna (Austria)*, pages 65–75, 1998.

3. A. Balluchi, M. Di Benedetto, C. Pinello, C. Rossi, and A. Sangiovanni-Vincentelli. Hybrid control in automotive applications: the cut-off control, 1999.
4. C. Benzmüller, C. Giromini, and A. Nonnengart. Symbolic verification of hybrid systems supported by mathematical services. In *Proceedings of the Calculemus Symposium 2002*, 2002. forthcoming.
5. O. Caprotti and A. M. Cohen. Draft of the Open Math standard. The Open Math Society, `http://www.nag.co.uk/projects/OpenMath/omstd/`, 1998.
6. B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, and S. M. Watt. *First leaves: a tutorial introduction to Maple V.* Springer Verlag, Berlin, 1992.
7. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Service Description Language. W3C Recommendation 1.1, World Wide Web Consortium, 2001. Available at `http://www.w3.org/TR/2001/NOTE-wsdl-20010315`.
8. M. Codutti. Nodes : Non linear ordinary differential equations solver. In *Proceedings of ISSAC'92 (International Symposium on Symbolic and Algebraic Computation)*, 1992.
9. S. Engell, S. Kowalewski, C. Schulz, and O. Stursberg. analysis and optimization of continuous-discrete interactions in chemical processing plants.
10. C. Benzmüller et al. ΩMEGA: Towards a mathematical assistant. In *Proceedings of the 14^{th} International Conference on Automated Deduction (CADE–14)*, pages 252–255. Springer Verlag, Berlin, 1997.
11. The Oz group. The mozart programming system. http://www.mozart-oz.org/.
12. T. Henzinger and H. Wong-Toi. Using hytech to synthesize control parameters for a steam boiler, 1996.
13. 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.
14. 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.
15. 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.
16. 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.
17. 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.
18. R. Horowitz and P. Varaiya. Design of an automated highway system. Proceedings of the IEEE. This issue.
19. M. Kohlhase. OMDoc: Towards an internet standard for the administration, distribution and teaching of mathematical knowledge. In *Proceedings of AI and Symbolic Computation, AISC-2000*, LNAI. Springer Verlag, 2000. Forthcoming.
20. M. Kohlhase and J. Zimmer. System description: The Mathweb Software Bus for Distributed Mathmatical Reasoning; to appear.
21. C. Livadas, J. Lygeros, and N. Lynch. High-level modelling and analysis of tcas, 1999.

22. 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.
23. 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/.
24. B.-C.-V. Ung. Combinatorial identities for series of quasi-symmetric functions.