Appeared in: S. Engell, G. Frehse, E. Schnieder (Eds.): Modelling, Analysis, and Design of Hybrid Systems.
Lecture Notes in Control and Information Science, Vol. 279, pp. 153-171, Springer-Verlag, 2002.

Introduction to the Analysis and Verification of Hybrid Systems

Stefan Kowalewski

Robert Bosch GmbH, Corporate Research and Development, Frankfurt am Main, Germany, stefan.kowalewski@de.bosch.com *

Abstract. This contribution provides an introduction to the formal analysis of hybrid systems. It highlights different directions from which hybrid models and their analysis have been approached in computer science and control theory. Fundamental problems arising from the combination of discrete and continuous dynamics are discussed and related articles in this volume are put in relation to the different basic approaches.

1 Introduction

The combination of interacting discrete and continuous dynamics in one model brings up the need for appropriate analysis methods. While such methods for purely discrete or for purely continuous systems have been existing for decades, the formal analysis and verification of hybrid systems is a relatively recent research problem and standard procedures are not yet available. It was therefore one of the main tasks in the focussed research program *Analysis and Synthesis of Technical Systems with Continuous-Discrete Dynamics (KONDISK)* of the German Research Council to develop new analysis and verification approaches or extend the existing methods such that they can be applied to hybrid systems. The four papers in this chapter [40,59,50,28] as well as further contributions to this volume (e.g., [35,39,12,17] present different solutions to this problem. They illustrate the broad spectrum of problem classes, models and analysis procedures in KONDISK.

The purpose of this introductory article is to put the contributions of this chapter into the perspective of a broader view on the formal analysis of hybrid systems. We highlight the different historic starting points and directions and discuss the main problems which have to be dealt with. Based on this, the contributions are put in relation to the different approaches.

The paper is organized as follows. In the following section *Modelling of Hybrid Systems* we revisit the problem of modelling hybrid systems because of its effects on the analysis. The nature of hybrid systems as a model property is discussed and three different starting points and directions are identified

* The insights and opinions expressed in this paper were developed while the author was with the Process Control Laboratory in the Department of Chemical Engineering at the University of Dortmund, Germany.

to create hybrid modelling frameworks from discrete and continuous ones. In the section *Analysis of Hybrid Systems* we discuss different levels of rigor of the analysis methods and computational issues. An example is presented to illustrate the theoretical computational limitations and the need for abstraction which arises from these results. A summary of the current state of the art of hybrid systems analysis concludes the paper.

2 Modelling of Hybrid Systems

2.1 The Nature of Hybrid Systems

Before we take a closer look at the analysis and verification methods, we discuss different approaches to modelling hybrid systems and their relation to analysis problems and procedures. A hybrid system is usually defined as a system which combines continuous and discrete dynamics. This definition is superficial. To be more precise, the term "hybrid systems" refers to models, not systems as such. A system is not hybrid by nature, but it becomes hybrid by modelling it this way. Whether it makes sense to build a hybrid model depends not only on the system, but also on the application and the purpose of the model. The latter most often concerns the analysis that shall be performed. So, there is a strong relationship between modelling and analysis of hybrid systems.

Hybrid systems arise and must be analyzed whenever both abstraction levels – continuous and discrete – have to be considered to solve a particular problem. However, this does not exclude that during the analysis the problem is mapped to a single abstraction level and solved there: As we will see later, in many cases it is either not possible or not appropriate to perform the complete analysis with a hybrid model. In this case it is often helpful to abstract from the concrete problem, e.g. by discretizing the continuous part and solving the problem using discrete analysis techniques. Thus, hybrid systems analysis and verification is not only concerned with hybrid models, but also with the problem of how to map hybrid problems into spaces where they can be solved better. This will be discussed in more detail in Sec. 3.3.

2.2 Different Approaches to Hybrid Modelling Frameworks

In principle, there are three different ways to create a modelling framework for hybrid systems:

1. The first option is to take existing discrete formalisms, e.g., finite automata, Petri nets or logics, and extend them by continuous variables which evolve according to differential equations associated with discrete states. Discrete transitions can then switch between continuous modes, and the continuous variables can be reset when a transition takes place. Resulting frameworks of this kind are *hybrid automata* [2] or *hybrid Petri nets* [16].

2. The opposite direction is to extend models for continuous systems by discrete mechanisms like switching or resetting time dependent continuous variables. The resulting frameworks consist of differential equations, algebraic equations and/or inequalities with both continuous and binary variables. The latter are used to activate and deactivate terms by multiplication, e.g. to switch the right hand side of the state equation. This class of systems is therefore often referred to as *switched continuous systems* [34].
3. The third approach is not to extend a formalism but to employ an existing discrete model and an existing continuous model as they are and couple them by appropriate interfaces. Prominent representatives of this concept are commercial simulation tools like *Simulink/Stateflow* or *MatrixX/Betterstate*.

In the KONDISK program, all three approaches were employed. Hybrid extensions of discrete formalisms are used, among others, by Decknatel et al. in [17], Simon et al. in [50] and Huuck et al. in [28]. In the first two cases, the model is based on Petri nets, in the third case the authors chose hybrid automata. It is interesting to note that in [17] the hybrid dynamics was realized solely by applying the modelling mechanisms already offered by the Petri net tool *DesignCPN*. The underlying model did not have to be extended. A good example for the second approach is [12]. The third approach can be found in [39] and [40] in which a continuous state space model with piecewise linear (or affine) dynamics is connected to a Petri net. The coupling of discrete and continuous formalisms has also been pursued in KONDISK by approaches to the simulation of hybrid systems (see [42] and [48]).

The three fundamental modelling approaches listed above are equivalent in the sense that the models are equally expressive as long as comparable assumptions about number spaces and permitted mathematical operations are made. The choice of one of the approaches is therefore usually determined by the scientific discipline: With only few exceptions, the first approach has been followed by computer scientists whereas control theorists have preferred the second and third approach. This is not surprising since the original domains of interest in these two fields were on the opposite ends of the hybrid dynamics spectrum – purely continuous dynamic systems in control theory, discrete state systems in computer science. (And in the third approach, although appearing to be a union of equally privileged formalisms, the continuous dynamics often was first and is still dominant at closer look. See, for instance, the triggering of discrete transitions in *Stateflow* by the integration steps in *Simulink*.)

The awareness of the opposite starting points and perspectives on hybrid modelling in computer science and control is important for the perception of research results from the other field. For example, in computer science it was natural to choose timed models like *timed automata* [2] as the first class of hybrid systems to investigate. Time can reasonably be modelled as a contin-

uous variable and, from a computer science point of view, extending discrete models by continuous time is the simplest way to obtain hybrid models. In the beginning of the KONDISK program, I often experienced the misconception by control theorists that this would not be genuine hybrid systems research. Looking from the continuous systems' end of the hybrid dynamics spectrum, they felt that "real" hybrid systems require more complex differential equations than $\dot{x} = 1$. In the meantime, this prejudice vanished and I believe that it was the fruitful exchange in the KONDISK program which helped to achieve this understanding. It is now commonly agreed on that timed systems are an interesting class of hybrid systems, both with respect to theoretical limitations of computer analysis (see Sec. 3.2) and practical usefulness as an abstraction of more complex hybrid dynamics (see Sec. 3.3). In this chapter, this is demonstrated by the papers [28] and [50] in which timed automata or timestamp Petri nets, respectively, are used to analyze hybrid systems.

A different issue in modelling hybrid systems is how to deal with uncertainty. Often, for various reasons, there is not sufficient information available to determine the exact next state of a transition or the exact time of switching. One way to deal with this issue is to use transition relations instead of transition functions, like in nondeterministic automata. Another possibility is to assign probabilities to competing transitions. Stochastic automata are a model of this kind (see [35] in this volume). It is also possible to specify probability distributions for the switching times of the discrete transitions, like in stochastic Petri nets. In this volume, Wolter et al. present an extension to stochastic Petri nets, so-called *fluid* stochastic Petri nets, in which a subset of the places carries a time-dependent continuous value instead of discrete tokens (see [59]).

3 Analysis of Hybrid Systems

3.1 Simulation and Verification

Model-based analysis of hybrid systems can be performed on different levels of rigor. For control engineers it is customary to build a *simulation model* and use it simulate scenarios of interest. This means that the input values and open decisions in the model are fixed before the model is executed. Then properties of the system are inferred from the resulting output or state trajectories. The shortcoming of this approach is that all the properties analyzed by this procedure are only proved for the considered scenarios. It cannot be excluded that there are other inputs for which the result would be different. However, it has to be noted that in many applications this problem does not occur because the relevant scenarios are easy to identify.

The activity of proving system properties for every possible choice of free inputs and decisions is called *formal verification* (or, in the following, simply *verification*) [14]. The term originates from computer science where two

different directions of formal verification are distinguished, *algorithmic* and *deductive* verification. Algorithmic verification, often called *model-checking*, means that a computer algorithm is used which receives a model of a system and a specification of its required behavior as input and then checks whether the requirements hold for all possible behaviors of the system model. This is done basically using search techniques, very often by computing the reachable states of the system. Each verification algorithm is applicable to a particular class of systems (e.g., finite transition systems).

In principle, algorithmic verification is only possible for such classes for which it is guaranteed that the search procedure terminates. In the case of hybrid systems, this is only true for very restricted classes (see Sec. 3.2). It is therefore often necessary to find a finite abstraction of a hybrid system before algorithmic verification can be applied. On the other hand, the advantage of algorithmic verification is that the user only has to provide the system model and the requirements and can leave the rest to the algorithm. No further expertise and knowledge of the analysis technique is needed. However, a major shortcoming in comparison to simulation is the computational complexity resulting from the exhaustive search in discrete spaces.

The most important algorithmic verification procedure for hybrid systems is *reachability analysis*. It answers the question whether for a given hybrid system a certain hybrid state (discrete state and a region in the continuous space) is reachable from the initial hybrid state. This problem is so important because many problems can be reduced to a reachability problem.

When applying deductive verification, also called *theorem proving*, the question whether a system has certain desired properties is answered by creating a proof. For this purpose, the user not only has to specify the system behavior and the requirements in an appropriate logic, but also has to find a suitable sequence of arguments. Although this is often supported by a set of proof-rules which can be applied in a schematic way, in the end, the success of the verification depends much more on the intuition, creativity and experience of the user than it does for algorithmic verification. However, one main advantage of deductive verification is that the application domain is not restricted to systems with finite search spaces or finite approximations. If a suitable theory is available, infinite systems (i.e., systems with infinitely many states) can be handled also. This is important in the case of hybrid systems which are infinite by definition due to the continuous part of their state.

In this volume, several papers deal with algorithmic verification. Müller et al. in [39] and Nenninger et al. in [40] present procedures for reachability analysis. In both cases, the analysis is used to design controllers. In [50] reachability is solved by analyzing whether those transitions which switch to the relevant states (or markings, as this procedure is defined for Petri nets) can not be blocked by the timing conditions. Again, this is used for design, in this case to determine valid timer parameters of a controller. The paper

by Huuck et al. ([28]) presents an approach in which algorithmic verification and deductive verification is combined. The purpose of the combination is to overcome the complexity problem of algorithmic verification by using deduction to structure the problem into smaller, more feasible subproblems [29]. Finally, [17] is a representative of the simulation approach to the analysis of hybrid systems. The authors simulate different scenarios in a railway system to determine the performance of the supervision system.

3.2 Computational Issues

It was mentioned before that exact algorithmic reachability analysis is only feasible for very restricted classes of hybrid systems. For the other classes, it is impossible to formulate an algorithm which computes the exact reachable state space for any system from this class in finite time. In other words, the reachability problem is *undecidable*. The major part of hybrid systems research in computer science has been concerned with identifying decidable and undecidable classes of hybrid systems (with respect to reachability, which is sometimes not mentioned explicitly). The control community, in contrast, has much less interest in this issue. As a matter of fact, two papers in this volume, [39] and [40], present reachability algorithms for a class of hybrid systems for which reachability is actually undecidable. To resolve this apparent contradiction, this section will provide a short introduction to the kind of problems which are looked at in computer science research on decidability of hybrid systems. It may help to understand not only the fundamental issues but also the practical implication that undecidability should not necessarily prevent engineers from developing reachability algorithms for the respective class of systems.

Hybrid Automata The most popular hybrid systems model in computer science is the *hybrid automaton* (HA) [1], see [28] in this volume for a formal definition. Roughly speaking, the HA model complements (discrete) finite automata by time-dependent continuous variables. While the system is in a certain discrete state, these variables evolve according to differential equations, called *flows*, which are assigned to each discrete state. Conditions can be formulated which have to be true while the system remains in a discrete state. They are called *invariants*. When an invariant evaluates to false, the discrete state must be left or must not be entered, respectively. The continuous variables can be *reset* by the discrete transitions, and finally, so-called *guards* represent conditions for taking a transition between discrete states.

Figure 1 shows an example of a HA which is borrowed from [27]. It belongs to the simplest and historically first class of HA, i.e., *timed automata* (TA) [2]. In a TA, the continuous variables are called *clocks* and their value is

always increasing with a rate of one [1]. Resets can only be assignments of zero, and the invariants and guards are independent inequalities or equalities with rational constants for each clock. In the case of Fig. 1, we have two discrete states, s_1 and s_2, and three clocks, a, b, and c. There are guards at each of the three transitions, but only with transition t_1 a reset (of the clocks b and c) will be performed. The arrow symbol is used to separate the guard from the reset. The invariant in s_1 is $a \leq W \wedge b \leq W \wedge c \leq W$, and in s_2 it is true, meaning that there is no condition restricting the entrance and the visiting time in s_2.

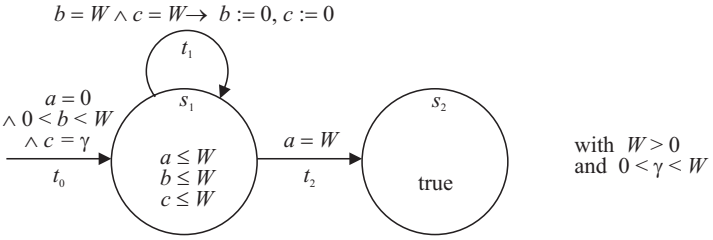

Fig. 1. Example for a timed automaton

To understand the behavior of TA (or, in computer science terms, the *operational semantics* of this model), it is helpful to know that TA were originally introduced by Alur and Dill in [2] as a generalization of *ω-automata*. ω-automata were developed to model non-terminating systems like, for instance, data base servers. They are automata over infinite words, which means their language consists of infinite sequences of symbols. Their acceptance criteria are based on accepting states which have to be visited infinitely often while reading an accepted word. So, acceptance can only be decided by looking at the infinite behavior.

The same is true for TA, only that the notion of infinity is not applied to sequences but to time. The behavior of a TA is given by a set of *runs* which are infinitely lasting trajectories in which discrete transitions (possibly with resets) and time intervals with continuous growth of the clock values alternate. Figure 2 shows a valid run for the TA from Fig. 1. It starts with transition t_0 at which the clocks b and c have the same value: $b = c = \gamma$ (note that this is not implied by the guard which would permit any value between 0 and W for b). When b and c increase to W, the invariant of s_1 will no longer permit to stay in this discrete state. Transition t_1 is possible because the guard is true, whereas t_2 cannot be taken. Performing t_1 triggers the reset of b and c, and s_1 can be entered again. When $a = W$ becomes true, transition t_2 must be taken. After that, the system remains in s_2 and

[1] Like in Fig. 1, the corresponding flows $\dot{x} = 1$ are usually omitted in the graphical representation of a TA.

the clocks can increase to infinity. Therefore, the run in Fig. 2 is infinitely lasting and, thus, is part of the behavior of the TA from Fig. 1.

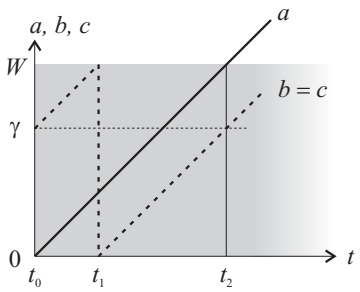

Fig. 2. A valid run for the example from Fig. 1

What would happen, if a different value for $b(t_0)$ is chosen, e.g., $b = \gamma/2$? In this case clock c would reach W first, the invariant would not permit further residence in s_1, but none of the two transitions could be taken because the guards are false (note that a and c are still less than W). Obviously, the behavior up to this point can not be extended to an infinitely lasting run – in computer science jargon, "time stops" or "a *time deadlock* occurs". In fact, $b(t_0) = c(t_0) = \gamma$ is a necessary condition for the existence of a run in the example[2].

Reachability Analysis of Hybrid Automata The (forward) reachability analysis of hybrid automata has to determine all possible runs and check whether the hybrid target state is visited during at least one of these runs. Hybrid states are pairs (s_i, R_j) consisting of a discrete state s_i and a set R_j of values of the continuous variables, often called a *region*. The analysis algorithm roughly works as follows.

1. The starting point is the initial discrete state s_0 and the initial region R_0 which is determined by intersecting the regions corresponding to the guard of the entering transition and the invariant of the initial discrete state. In our example, this would be $(s_0, R_0) = (s_1, \{(a, b, c) \in Q | a = 0 \wedge 0 < b < W \wedge c = \gamma\})$.
2. The next step is to let the region grow according to the flow assigned to the discrete state but neglecting the invariant (or in other words, assuming that the system could remain in the discrete state forever). In a TA, for instance, this means that all continuous variables increase by a rate of one. In the example, the resulting region R_1 is unbounded: $R_1 = \{(a, b, c) \in Q | c = a + \gamma \wedge a \leq b \leq a + W\})$.

[2] For this reason, the TA from Fig. 1 can be found in proofs to model a function which checks the equivalence of two values [27]. Note that $b = c = \gamma$ holds also at t_2.

3. Obviously, not all of R_1 is actually reachable because the invariant would force the system to leave s_0 as soon as it is violated. To determine the values actually possible during this visit of s_0, R_1 is intersected with the invariant (in the example: $R_2 = \{(a,b,c) \in Q | (a,b,c) \in R_1 \wedge a \leq W \wedge b \leq W \wedge c \leq W\}$).
4. At this stage, the algorithm would take the list of previously visited regions and check whether R_2 (or a subset of it) had been computed for s_0 before. If this is case, it would abort the current search branch.
5. If not, the hybrid state (s_0, R_2) is added to the reachability set and the algorithm will continue this search branch. Now, all possible transitions from s_0 have to be determined. In our example, there are only t_1 and t_2. To find out which of them are viable, we have to check whether the guards can become true for the values of a, b and c in R_2. This is computed by the intersection. In the case of t_2 the result is empty, thus, the transition is not possible. For t_1 the result is $R_3 = \{(a,b,c) \in Q | a = W - \gamma \wedge b = W \wedge c = W\}$, so it can be taken.
6. The algorithm chooses one of the possible transitions and performs the corresponding reset. The result is the region with which the system can enter the new discrete state. In the example, the reset leads to $R_4 = \{(a,b,c) \in Q | a = W - \gamma \wedge b = 0 \wedge c = 0\}$. Now, the second iteration starts and the algorithm goes back to step 2.
7. The algorithm terminates when no more new transitions can be traversed.

There are tools available in which this algorithm or variants of it are implemented. The most prominent are Kronos [60], Uppaal [32], and Hytech [25]. The class of systems which can be handled by these tools differ, but they have in common that the regions R_i can be represented by polyhedra.

For TA, this algorithm will always terminate. This means that the reachability of TA is decidable [2]. However, even small generalizations can lead to undecidability. This shall be illustrated by the example in Fig. 3 which is inspired by similar examples in [27]. The only extension to the model is that we allow one clock to be stopped and to be started again with the value at stopping time. In other words, the flow can be either $\dot{x} = 1$ or $\dot{x} = 0$. Such a clock is called a *stopwatch* [27], the resulting model is a *stopwatch automaton*. In Fig. 3, d is a stopwatch while a, b, and c are clocks as in the example before[3].

Figure 4 will help to understand the behavior of the example. It shows a fragment of a run with the choice of $b(t_0) = \gamma/2$ [4]. Clock a can be regarded as providing a clock tick with a constant frequency $1/W$. In the fragment of Fig. 4 it defines five time intervals δ_1 to δ_5.

[3] As in Fig. 1, the flows of the clocks are omitted and only the flows of the stopwatch are presented.

[4] Note that the t_i symbolize transitions and not points in time. However, for the fragment of Fig. 4 there is no difference because each transition is taken only once.

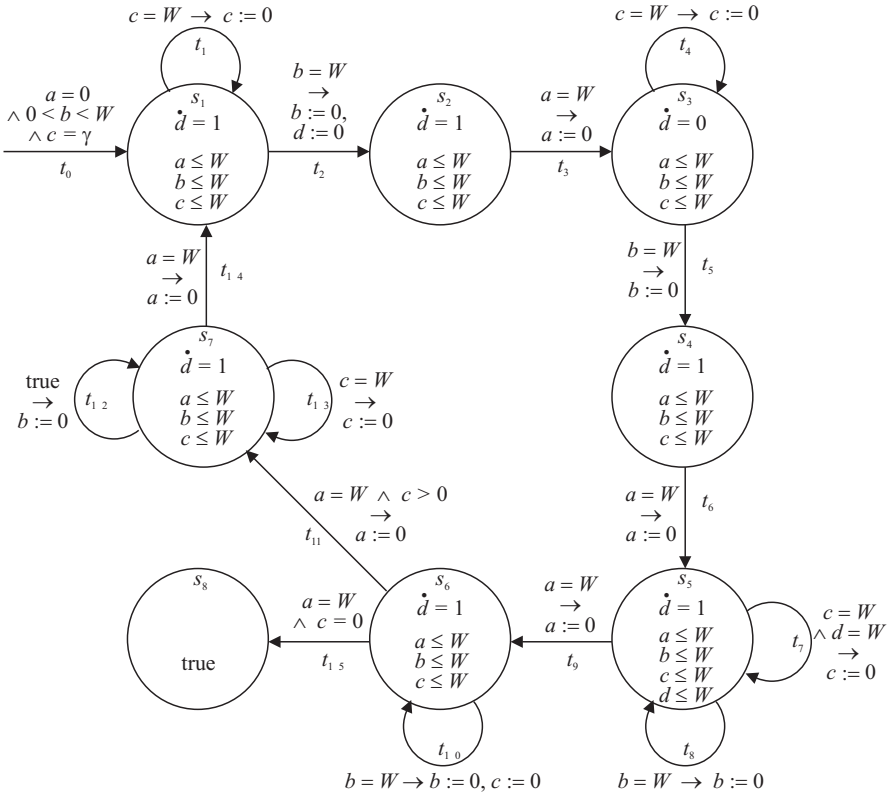

Fig. 3. Example for a stopwatch automaton with non-terminating reachability analysis

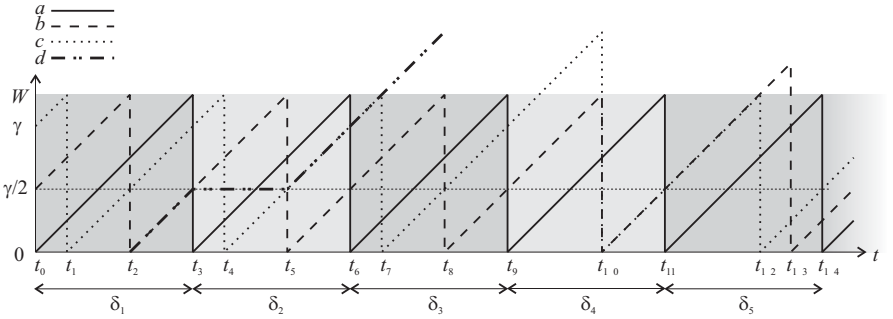

Fig. 4. A fragment of a run for the stopwatch automata from Fig. 3

- The purpose of δ_1 is to synchronize clock b and stopwatch d, which is done at t_2. Note that apart from this change, all clocks have equal values at t_0 and at t_3.
- When δ_2 starts by entering s_3, d is stopped. With t_5, it is started again and, because we chose $c(t_0) = 2{\cdot}b(t_0)$, it is now synchronized with c. At the end of δ_2 $d = \gamma$ holds and the clocks again have the same values as at t_0.
- In δ_3 it is checked whether $c(t_6) = d(t_6)$. This is done by applying the construction of Fig. 1. Transition t_7 can only be taken if $c(t_6) = d(t_6)$, otherwise there would be a time deadlock. So, at the end of δ_3, the run can only proceed if $c(t_6) = d(t_6)$. Since this requires $c(t_3) = 2{\cdot}b(t_3)$ in δ_2, the whole part of the TA from t_0 to t_9 can be regarded as a test of $b(t_0) = c(t_0)/2 = \gamma/2$.
- In δ_4, c is synchronized with b. The result is $c(t_{11}) = b(t_9)$. Now, if we consider the value of c at t_0 and at t_{11}, it becomes apparent that our stopwatch automata did nothing else but the assignment $c := c/2$.
- The purpose of δ_5 is to choose an arbitrary new value for b before s_1 is entered again and the next cycle begins. Of course, the time deadlock in s_5 can only be avoided, if the choice is such that $b(t_{14}) = b(t_{11})/2$.

When the run is continued, this cycle will repeat and every time t_{11} is taken c will increase to half of its value at the beginning of the cycle. This leads to an infinite sequence $(\gamma, \gamma/2, \gamma/4, \ldots)$ which asymptotically approaches zero but will never reach it. If we now consider the problem whether state s_8 is reachable in the automaton of Fig. 3, it is easy to see that the algorithm described above will not terminate: Every time, the possible transitions from s_6 are checked, c will have a new, smaller but positive value and t_{11} has to be taken again.

In practice, that is, when applying tools like Kronos, Uppaal, and Hytech, the algorithm will be aborted after some time. The reason for this is that the tools were implemented for exact reachability analysis and therefore the rational values for storing the regions (e.g. the corner points) are represented by two integers for the nominator and denominator, respectively. As a consequence, infinite sequences, like the one for clock c at transition $t_1 1$ in the example above, would lead to memory overflows for the integers.

Of course, finding one example for which a particular algorithm will not terminate does not prove undecidability of the general problem because the example may be analyzable by other approaches. Actually, for this particular example it can be shown by deduction that s_8 is unreachable. A proof that reachability of stopwatch automata is undecidable can be found in [27] together with several other decidability results. The proofs are conducted by reducing the reachability problem to a problem which is known to be undecidable. Some of the constructions used in the examples of Figs. 1 and 3 are taken from these proofs.

Discussion The undecidability of stopwatch automata does not mean that any class of hybrid systems that is more general than TA is undecidable. There have been other classes defined in which the flows are, for example, differential inclusions or even linear differential equations with particular properties, and decidability is achieved by certain restrictions on the guards or resets [27,31]. But the fact remains that the decidability boundary is far beyond the classes usually considered for control systems. So, what are the practical implications of this theoretical result? The answer has four aspects.

- The first aspect still follows from theory: Even for undecidable classes of hybrid systems, the reachability algorithm may well terminate for the particular problem under consideration. It is also possible that backward reachability will terminate while forward analysis does not, or vice versa.
- The second aspect is that it is often possible to find abstractions, which still are sufficient models for the analysis problem but fall into decidable classes (see Sec. 3.3).
- The third aspect is more pragmatic: It is usually a much bigger problem to cope with the exponential complexity of the algorithm with respect to the number of continuous variables (which adds to the discrete state space explosion problem), so that even for decidable problems it may be impossible to wait for the analysis result or memory overflows occur.
- Finally, from an engineering point of view, absolute exactness of the analysis is not appropriate because the models and the requirements already are of limited accuracy. If, for example, s_8 would represent a dangerous state in a technical process and reachability analysis should check whether it will be avoided, $\dot{c} = 1$ would only be an approximation of the real clock speed. It would therefore be sufficient for the analysis, if the sequence would be stopped when the value for c is rounded down to zero. Thus, in applications, it is reasonable to use approximate analysis with numerical rounding as long as the error is bounded (as it has been practice in control engineering for a long time).

For these reasons it can be justified to ignore the undecidability issue when developing analysis procedures for hybrid systems.

3.3 Abstraction

The computational problems of reachability analysis (undecidability, complexity) are the motivation for a very active area in hybrid systems research which is concerned with abstraction techniques. The basic idea is the following: Instead of trying to analyze the original system under investigation, the analysis problem is mapped into a class of problems which is easier to solve. The mapping consists of two steps. First, a substitute model is created by omitting details from the original model (for example by replacing the exact continuous state by a discretized one). This model will include the behavior

of the original system but, in general, allow additional behavior because of the less concrete specification. In the second step the original property to check is generalized so that it can be reformulated for the substitute model. This kind of mapping (and its result) is called an *abstraction*.

The first advantage of abstractions is that the resulting model is rougher and, thus, often simpler and analyzable with less computational effort. Of course, this gain of efficiency must be paid for by a loss of accuracy. The consequence for dynamic models is that the degree of uncertainty will increase. For instance, state space discretization of a deterministic continuous system in general will lead to a nondeterministic discrete system. This means for the analysis that the results can be inconclusive. If, for example, reachability analysis of an abstracted system shows that an abstracted state region is reachable, this could be because the original target region is reachable in the concrete system. But it may as well be that it is just one of the additional trajectories in the abstracted model which reaches the target region, or that the reachable part of the target region was just added by the abstraction. However, if the abstract region is *not reachable*, we can be sure that the corresponding concrete region is not reachable in the original system, neither. This is the second advantage of abstractions (which general estimations do not have): Problems can be posed such that one of the possible analysis results is conclusive and provides a guaranteed solution to the original problem.

Because of the potential inconclusiveness and the loss of accuracy, abstractions are only useful if two conditions are fulfilled: First, the properties of interest must be formulated such that conclusive results are possible, e.g., reachability of forbidden states for a conservative safety analysis. Second, the level of uncertainty of the dynamics must not be so low that only inconclusive results exist (e.g., the whole state space becomes reachable).

This problem and further issues concerning abstractions of hybrid systems are discussed, for example, by Lunze and Raisch in this volume in [35] for the case of discrete systems as substitute models. In [4], Alur et al. provides a survey on fundamental theoretical results from computer science on this topic. Particular methods to generate a discrete abstraction for a given hybrid or continuous system are presented, for instance, in [13,15,22,47]. Further approaches can be found in the special issue [20]. The idea of representing continuous dynamic systems by discrete abstractions, however, is older than the recent research on hybrid systems. It was already the basis of the work on qualitative simulation, see for example [30].

Abstraction by Hybrid Automata The abstraction of switched continuous systems with linear or even nonlinear differential equations by discrete automata is a relatively rough approximation. All the quantitative information about the dynamics is lost and replaced by a qualitative description. One way to save quantitative information is to capture the arising uncertainty in stochastic models (see [59] and [35] in this volume). Another possibility is to

use hybrid automata as abstractions [26,51–54]. The remainder of this section is devoted to the latter approach.

As in most work on abstraction of switched continuous systems, the basis of this approach is an orthogonal partitioning of the continuous state space. This means that each dimension is divided into bounded or unbounded intervals. The result are hyper–rectangles as partition cells and hyper–planes as boundary manifolds. The abstraction by hybrid automata then consists of three steps: First, the discrete state space is defined based on the partitions. Second, the continuous dynamics in each partition cell is abstracted so that it complies to the desired class of hybrid automata. In the third step, the discrete transitions are determined by analyzing which partition cells are connected by trajectories in the abstracted dynamics.

For the first step, it is straightforward to map each partition cell into a discrete state. If we choose TA as the target model, however, this construction has the disadvantage that the state can move from a cell to a neighbored one in zero time. Thus, zero will be the lower time limit for each transition and, consequently, each trajectory in the abstracted state space could be traversed with zero time consumption. To avoid this undesired idealization, the discrete states can be defined to lie on the boundary hyper–planes between cells [53,54,51].

The abstraction of the continuous dynamics depends on the chosen class of hybrid automata. For TA, we have to determine the upper and lower limit of the time that the continuous state can reside in a partition cell or, in the case of mapping discrete states to boundary hyper–planes, of the time that is needed to move from one boundary hyper-plane to the next.

The orthogonal partitioning, however, suggests a further class of hybrid automata for abstraction, namely *Rectangular Automata (RA)* [52]. Roughly speaking, in a RA the invariants, guards, reset sets and flows are *rectangular predicates*, which means they are specified by intervals (possibly degraded to points) for each continuous variable or its time derivative, respectively. For a formal definition see [27]. In the work described here, the guards only check the equivalence of one variable to one of its bounding values, and the resets are always assignments of a bounding value to a variable.

The abstraction of the switched continuous system with orthogonal partitioning to a RA is straightforward: The invariants, guards, and resets are given by the partition boundaries. The only problem left to solve is to find upper and lower limits for the flow intervals. For non-trivial systems, the corresponding optimization problem cannot be solved analytically. In these cases the flows can be approximated conservatively by interval arithmetics [52,51].

Approximate Analysis of Rectangular Automata At this stage the question arises how the resulting RA can be analyzed algorithmically. In Sec. 3.2 it was demonstrated that reachability of stopwatch automata is undecid-

able, and RA obviously are more general than stopwatch automata. Moreover, the numerical problems due to the integer arithmetics of the mentioned tools for HA apply to RA, too. In this section, an algorithm is sketched which overcomes these problems by conservatively approximating the reachable regions in a RA [43]. The first version of the algorithm was introduced in [44], a more complex version with smaller over–approximations was presented in [46]. In the following, only the first version is sketched in order to provide an impression of the basic idea.

The algorithm is based on the concept of faces. A *face* is a rectangular predicate with one dimension fixed to a certain value. The rationale for introducing faces is to use rectangular faces to represent non-rectangular sets. A *face-region* $\mathcal{F}$ is a set $\{F_1, \ldots, F_q\}$ where each F_i is a face. The semantics of $\mathcal{F}$ is the convex hull over its q faces, i.e. $\langle \mathcal{F} \rangle = \mathit{convexhull}\{\langle F_1 \rangle, \ldots, \langle F_q \rangle\}$. This is shown for an example in Fig. 5 where a face-region $\mathcal{F}_1$ is represented by the two faces F_1 and F_2. In practice, the faces of a face-region over n variables are derived from $2n$ constraints of the form $x_j = l_1$ or $x_j = l_2$. In the example, the face F_1 corresponds to $x_1 = 1$ and the face F_2 to $x_2 = 7$, with the empty faces for $x_1 = 7$ and $x_2 = 1$ being omitted.

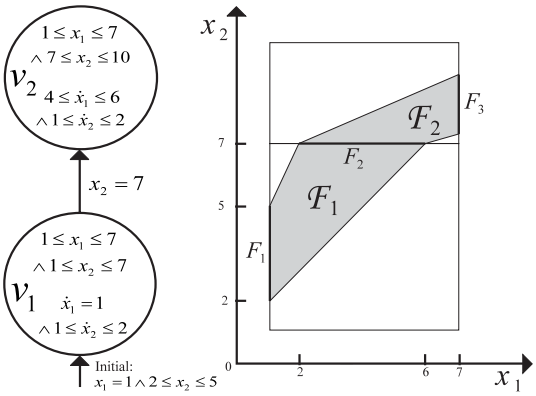

Fig. 5. Reachability analysis of RA using *faces*

A reachable face-region within the invariant can be represented by faces that lie on the invariant's bounds. Let $\mathcal{F}_1$ be a reachable face-region in a discrete state v_1. Now we want to compute the new face-region $\mathcal{F}_2$ in another discrete state v_2 that is adjacent to v_1 in terms of the invariant conditions. Then we can first check if any face of $\mathcal{F}_1$ is within the invariant condition of v_2. In our example this holds for F_2. So, this face can be used to determine a reachable region $\mathcal{F}_2$ in discrete state v_2. This is done by determining for each boundary l of an invariant of v_2 a face as the part of invariant l that can be reached starting from F_2 according to the possible flow in v_2. Here, only for the boundary $x_1 = 7$ a face can be found, namely F_3.

We use the computation of F_2 from F_1 in the example in Fig. 5 to show how an outgoing face can be computed from an ingoing face. First we determine a time interval in which any point within F_1 will be moved to F_2 according to the flow in dimension x_2. The distance between F_1 and F_2 in dimension x_2 ranges between 2 (=7-5) and 5 (=7-2). With a flow $1 \leq \dot{x}_2 \leq 2$ in v_1 this distance can/must be cleared within a time interval $T = [1; 5]$. Since the flow in each dimension is independent from the other dimensions, we can now use this time interval to compute how any point in F_1 will be shifted in the other dimensions while moving towards F_2. In our example, the only other dimension is x_1 for which we have a fixed flow $\dot{x}_1 = 1$. So in the time interval $T = [1; 5]$ a point starting from $x_1 = 1$ can flow to values ranging from 2 to 6. This yields F_2 with $2 \leq x_1 \leq 6 \wedge x_2 = 7$.

The complete reachability analysis is performed by considering all outgoing faces of an initial discrete state as ingoing faces to adjacent discrete states to which control switches exist. For these incoming faces then the outgoing faces within the invariants of the adjacent discrete states are computed. In the next step these newly computed faces are considered as ingoing to all adjacent discrete states again and so an iteration evolves. This iteration terminates when all reachable faces of a given automaton are found. The termination is guaranteed, since RA are always defined over a finite discrete state space and our analysis is approximate. Due to rounding in the approximative analysis there is only a finite number of points considered in the continuous state space. Thus, there is also only a finite number of faces that the algorithm can find within this state space.

Example Using the algorithm described in the previous subsection, the abstraction of switched continuous systems by RA can be practically applied to the analysis of switched continuous systems. We illustrate the results of this kind of analysis by a small example taken from [45]. The example is a two-tank system in which the first tank is filled by a fixed input flow F_{in} and is emptied into $Tank$ 2 through a connecting pipe (see Fig. 6). The outflow of $Tank$ 2, which is located on a lower level than $Tank$ 1 (height difference: H), is denoted by F_{out}. The flow in the connecting pipe depends on F_{in}, the liquid levels h_1 and h_2 in both tanks, and the setting of the valve controlling the flow F_{12}. The latter can be in two positions, *half–open* and *open*.

The dynamical behavior of this switched continuous system can be described as follows. The state vector is (h_1, h_2), and the variable $valve$ denotes the input of the system. Changes of the gradient field defined by Eq. 1 occur when either $valve$ is switched to another discrete value, or when h_2 exceeds H. The normalized parameters are: $A_1 = 1.14 \cdot 10^{-2}$, $A_2 = 1.98 \cdot 10^{-3}$, $H = 0.4$, $F_{in} = 1.11 \cdot 10^{-4}$, $K_1^1 = 1.2 \cdot 10^{-4}$, $K_2^1 = 3.4 \cdot 10^{-4}$, and $K_2 = 1.5 \cdot 10^{-4}$.

$$\dot{h}_1 = (F_{in} - F_{12})/A_1,$$
$$\dot{h}_2 = (F_{12} - F_{out})/A_2,$$

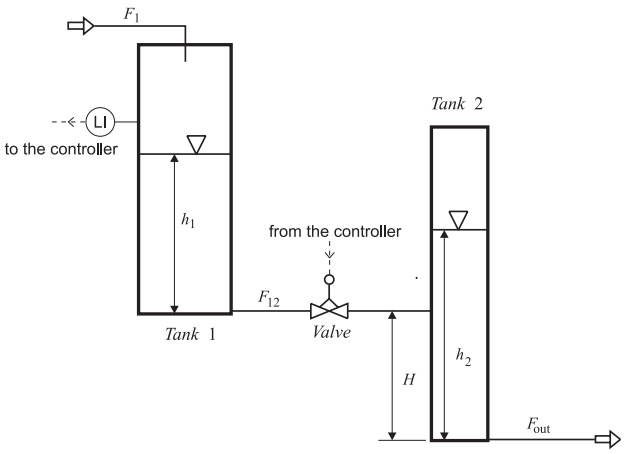

Fig. 6. Scheme of the two-tank system

$$
\begin{aligned}
h_2 < H \;:\; & F_{12} = K_1 \cdot \sqrt{h_1}, \\
h_2 \geq H \;:\; & F_{12} = K_1 \cdot \sqrt{h_1 - h_2 + H} \quad \text{if} \quad h_1 \geq h_2 - H, \\
& F_{12} = 0 \quad \text{else}, \\
F_{out} &= K_2 \cdot \sqrt{h_2}, \\
valve &= \begin{cases} 'half-open' : K_1 = K_1^1 \\ 'open' \quad\quad\;\; : K_1 = K_1^2 \end{cases}
\end{aligned} \tag{1}
$$

The analysis is concerned with the following scenario. We assume that the initial liquid heights are $h_1 = [0.2, 0.3]$ and $h_2 = [0.2, 0.3]$ and that $valve = 'half-open'$ applies. Since F_{12} is smaller than F_{in} at this setting, h_1 will rise. To prevent an overflow of $Tank\ 1$ the controller switches the value of $valve$ to 'open' as soon as it receives the information that h_1 has reached the value $h_{1,S} = 0.8$. As a consequence, h_1 will start to decrease immediately and h_2 will increase. The analysis shall check whether opening the valve can lead to a situation in which the limit $h_2 > 0.9$ is exceeded.

The abstraction is performed according to the procedure described above. The range of h_1 and h_2 is divided into 10 intervals each of equal length which leads to 100 discrete states in the RA. The result of the reachability analysis using the presented algorithm is shown in Fig. 7. The grey-shaded area marks the region which is determined as reachable from the dark-shaded initial region. To provide a better understanding of the analysis result, the continuous trajectories starting at the corners of the initial region are drawn additionally. The plot reveals that the *critical* region with $h_2 > 0.9$ is found

to be reachable, i.e. the switching value $h_{1,S}$ was not chosen correctly to avoid an overflow of $Tank$ 2. [5].

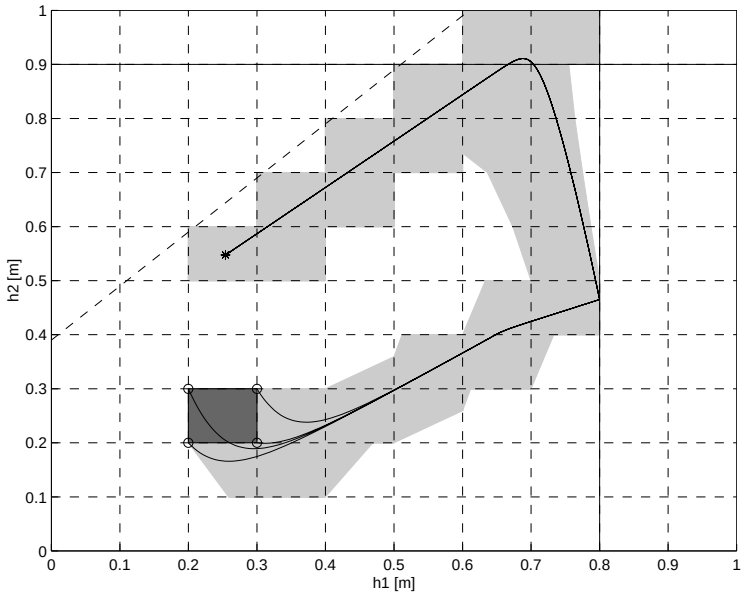

Fig. 7. Analysis results

The example demonstrates the degree of over–estimation which is the price for the simpler analysis model. It also hints at a problem arising from the orthogonal partitioning: In the first part of the trajectory, the abstraction results in reachability of complete partition cells. This is because the gradient field approaches the trajectories to the equilibrium point from both sides and change the sign of direction in both dimensions. This affect can lead to bad over–approximations which make the results inconclusive. For a more thorough analysis of this example, the reader is referred to [43].

Experiences Empirical data about the efficiency of this and other approaches to the verification of hybrid systems can be found in [43,51,56]. Experiences are reported for several examples with different complexity of the discrete part and the continuous part as well as for different abstraction and analysis methods. The abstraction of switched continuous systems

[5] Obviously, in this example the reachable region can easily be determined by simulation. However, for more complex systems exhaustive simulation will become impossible.

into TA or RA by interval arithmetics was performed for three–dimensional systems and discretizations with roughly 800 cells or discrete states, resp. [51]. Computing time on a PC (Pentium II, 266 MHz) was in the order of 20 minutes for RA and 40 minutes for TA. In [43] the reachability analysis of RA was applied to a more complex version of the two–tank example in the previous subsection. The system was three–dimensional (a continuous valve was added) and the state space was partitioned into 600 rectangles. The analysis took 11 minutes. In [56], the problem which is also treated by Wolter et al. ([59] in this volume) is solved using a TA model. The largest model was three–dimensional (three workpieces) with a discretization into seven intervals for each dimension (i.e., the temperatures of the workpieces). Computing time was 3 hours.

3.4 Alternative Approaches

An interesting alternative approach in the research on analysis of hybrid systems is the application of optimization techniques. The use of mathematical programming for the analysis of switched continuous models was suggested by Dimitriadis et al. [18,19]. The reachability problem is reformulated as an optimization problem in the discrete time domain which can be solved by mixed integer programming. Basically, the optimization determines the worst possible behavior, meaning that the system is most often in an undesired region of the continuous state space. The approach is general in the sense that it can be applied to hybrid systems as well as to purely discrete or purely continuous systems. Its strength lies in the ability to take advantage of well tested and efficient optimization procedures. A limitation is given by the fact that the size of the mixed integer program grows with the product of the number of discrete time steps and the number of equations and logical expressions describing the plant and the controller, respectively. A similar approach has been followed by Bemporad and Morari [11]. Here, an iterative scheme is used to perform conventional reachability analysis. This scheme avoids setting up a huge one-step optimization problem which is most likely not tractable. It can therefore be applied to larger problems than the approach of [18,19]. The verification method is part of a comprehensive modelling and analysis approach to hybrid systems, including a scheme for model-predictive control [10]. Further representatives of the mathematical programming approach to verification are Park and Barton who solve purely discrete model checking problems by integer programming [41]. In this volume, Stursberg et al. employ optimization techniques to design control policies for hybrid systems (see [55]).

4 Conclusions

The paper presented an overview on different approaches to the modelling and analysis of hybrid systems. We discussed the theoretical problem of un-

decidability and its practical implications. Approaches to overcome this and other challenges like applicability to large systems were sketched.

The current status of hybrid systems analysis can be characterized as follows. The theoretical foundations are largely established, the main obstacles on the way to practical application are identified, and first progress in this direction is made. The major challenge is still the computational complexity of the analysis procedures. The contributions to this volume provide good examples of promising approaches to move the research in hybrid systems analysis nearer to practical application.

For more information about the analysis of hybrid systems the reader is referred to the numerous proceedings volumes and to special issues of various control journals which appeared in the recent years. The main conference series are *Hybrid Systems* [23,8,3,9,6], *Hybrid Systems: Computation and Control* [37,24,57,36], or *Automation of Mixed Processes* (in future: *Analysis and Design of Hybrid Systems*) [61,21]. Examples for special issues on Hybrid Systems are [7,49,5,38], a survey on the control of hybrid systems can be found in [33]. A monograph is also available [58].

5 Acknowledgments

The results and opinions presented in this paper were developed while I was a member of the Process Control Laboratory in the Chemical Engineering Department at the University of Dortmund. They are the result of many discussions with colleagues and partners in several research projects. I am in particular grateful to Nanette Bauer, Paul Chung, Sebastian Engell, Holger Graf, Hans-Michael Hanisch, Oded Maler, Bruce Krogh, Yassine Lakhnech, Angelika Mader, Peter Niebert, Jörg Preußig, Olaf Stursberg, and Heinz Treseler. Apart from the KONDISK program, the following research projects contributed to the presented results and experiences: the ESPRIT LTR project *Verification of Hybrid Systems (VHS)* funded by the European Commission (see [38]), the temporary graduate school ("Graduiertenkolleg") *Modelling and Model-Based Design of Complex Technical Systems* funded by the German Research Council (DFG), and the exchange programs *British-German Academic Research Collaboration (ARC)* with the British Council and *Project-related Exchange of Personnel* with the NSF both funded by the German Academic Exchange Service (DAAD).

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 and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1990.

3. R. Alur, T. A. Henzinger, and E. D. Sontag, editors. *Hybrid Systems III: Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*. Springer, 1996.
4. R. Alur, T.A. Henzinger, G. Lafferiere, and G.J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, July 2000.
5. P. Antsaklis, editor. *Special Issue on Hybrid Systems: Theory and Applications*, volume 88, no. 7 of *Proceedings of the IEEE*, July 2000.
6. P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, editors. *Hybrid Systems V*, volume 1567 of *Lecture Notes in Computer Science*. Springer, 1999.
7. P. Antsaklis and A. Nerode, editors. *Special Issue on Hybrid Control Systems*, volume 43 of *IEEE Transactions on Automatic Control*, April 1998.
8. P. Antsaklis, A. Nerode, W. Kohn, and S. Sastry, editors. *Hybrid Systems II*, volume 999 of *Lecture Notes in Computer Science*. Springer, 1995.
9. P. Antsaklis, A. Nerode, W. Kohn, and S. Sastry, editors. *Hybrid Systems IV*, volume 1273 of *Lecture Notes in Computer Science*. Springer, 1997.
10. A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *automatica*, 35(3):407–427, 1999.
11. A. Bemporad and M. Morari. Verification of hybrid systems using mathematical programming. In Frits W. Vaandrager and Jan H. van Schuppen, editors, *Hybrid Systems: Computation and Control, Proc. 2nd Int. Workshop, HSCC'99, Berg en Dal, The Netherlands, March 1999*, Lecture Notes in Computer Science 1569, pages 31–45. Springer, 1999.
12. M. Buss, M. Glocker, M. Hardt, O. von Stryk, R. Bulirsch, and G. Schmidt. Nonlinear hybrid dynamical systems: Modeling, optimal control, and applications. In S. Engell, G. Frehse, and E. Schnieder, editors, *Modelling, Analysis, and Design of Hybrid Systems*, Lecture Notes in Control and Information Science, pages 311–336. Springer, 2002. (This volume).
13. A. Chutinan and B.H. Krogh. Computing approximating automata for a class of linear hybrid systems. In *Hybrid Systems V: Proc. Int. Workshop, Notre Dame, USA*, Lecture Notes in Computer Science 1567, pages 16–37. Springer, 1999.
14. E.M. Clarke and R.P. Kurshan. Computer-aided verification. *IEEE Spectrum*, pages 61–67, June 1996.
15. T. Dang and O. Maler. Reachability analysis via face lifting. In T.A. Henzinger and S. Sastry, editors, *Hybrid Systems: Computation and Control, Proc. 1st Int. Workshop, HSCC'98, Berkeley, USA, March 1998*, Lecture Notes in Computer Science 1386, pages 96–109. Springer, 1998.
16. R. David and H. Alla. *Petri nets and Grafcet*. Prentice Hall, New York, 1992.
17. G. Decknatel, R. Slovák, and E. Schnieder. Definition of a type of continuous-discrete high-level petri nets and its application to the performance analysis of train protection system. In S. Engell, G. Frehse, and E. Schnieder, editors, *Modelling, Analysis, and Design of Hybrid Systems*, Lecture Notes in Control and Information Science, pages 355–368. Springer, 2002. (This volume).
18. V.D. Dimitriadis, N. Shah, and C.C. Pantelides. A case study in hybrid process safety verification. *Computers and Chem. Eng.*, 20, Suppl.:S503–S508, 1996.
19. V.D. Dimitriadis, N. Shah, and C.C. Pantelides. Modelling and safety verification of dicrete/continuous processing systems. *AIChE Journal*, 43(4):1041–1059, 1997.

20. S. Engell, editor. *Special Issue on Discrete Event Models of Continuous Systems*, volume 6, no. 1 of *Mathematical and Computer Modelling of Dynamical Systems*, March 2000.
21. S. Engell, S. Kowalewski, and J. Zaytoon, editors. *4th Int. Conf. on Automation of Mixed Processes: Hybrid Dynamic Systems (ADPM 2000), Dortmund, Germany*. Shaker Verlag, Aachen, 2000.
22. M. Greenstreet and I. Mitchell. Reachability analysis using polygonal projections. In Frits W. Vaandrager and Jan H. van Schuppen, editors, *Hybrid Systems: Computation and Control, Proc. 2nd Int. Workshop, HSCC'99, Berg en Dal, The Netherlands, March 1999*, Lecture Notes in Computer Science 1569, pages 103–116. Springer, 1999.
23. R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors. *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*. Springer, 1993.
24. T. A. Henzinger and S. Sastry, editors. *Hybrid Systems – Computation and Control (HSCC'98)*, volume 1386 of *Lecture Notes in Computer Science*. Springer, 1998.
25. T.A. Henzinger, P.S. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1,2):110–122, 1997.
26. T.A. Henzinger, P.S. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):540–554, 1998.
27. T.A. Henzinger, P.W. Kopke, A. Puri, and P.Varaiya. What's decidable about hybrid automata. *J. Comp. Syst. Science*, 57:94–124, 1998.
28. R. Huuck, B. Lukoschus, G. Frehse, and S. Engell. Compositional verification of continuous-discrete systems. In S. Engell, G. Frehse, and E. Schnieder, editors, *Modelling, Analysis, and Design of Hybrid Systems*, Lecture Notes in Control and Information Science, pages 225–246. Springer, 2002. (This volume).
29. S. Kowalewski, P. Herrmann, S. Engell, R. Huuck, H. Krumm, Y. Lakhnech, and B. Lukoschus. Approaches to the formal verification of hybrid systems. *at-Automatisierungstechnik*, 49(2):66–74, 2001.
30. B. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.
31. G. Lafferiere, G.J. Pappas, and S. Yovine. A new class of decidable hybrid systems. In Frits W. Vaandrager and Jan H. van Schuppen, editors, *Hybrid Systems: Computation and Control, Proc. 2nd Int. Workshop, HSCC'99, Berg en Dal, The Netherlands, March 1999*, volume 1569 of *Lecture Notes in Computer Science*, pages 137–151. Springer, 1999.
32. K.G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *Software Tools for Technology Transfer*, 1(1,2):134–152, 1997.
33. M. Lemmon, K. He, and I. Markovsky. Supervisory hybrid systems. *IEEE Control Systems Magazine*, 19:42–55, August 1999.
34. D. Liberzon and A. S. Morse. Basic problems in stability and design of switched systems. *IEEE Control Systems Magazine*, 19, August 1999.
35. J. Lunze and J. Raisch. Discrete models for hybrid systems. In S. Engell, G. Frehse, and E. Schnieder, editors, *Modelling, Analysis, and Design of Hybrid Systems*, Lecture Notes in Control and Information Science, pages 67–82. Springer, 2002. (This volume).
36. N. Lynch and B. H. Krogh, editors. *Hybrid Systems – Computation and Control (HSCC 2000)*, volume 1790 of *Lecture Notes in Computer Science*. Springer, 2000.

37. O. Maler, editor. *Hybrid and Real-Time Systems (HART'97)*, volume 1201 of *Lecture Notes in Computer Science*. Springer, 1997.
38. O. Maler, editor. *Special Issue on Verification of Hybrid Systems*, volume 7, issue 4 of *European Journal of Control*, 2001.
39. C. Müller, P. Orth, D. Abel, and H. Rake. Synthesis of a discrete control for hybrid systems by means of a petri-net-state-model. In S. Engell, G. Frehse, and E. Schnieder, editors, *Modelling, Analysis, and Design of Hybrid Systems*, Lecture Notes in Control and Information Science, pages 295–310. Springer, 2002. (This volume).
40. G. Nenninger and V. Krebs. Reachability analysis and control of a special class of hybrid systems. In S. Engell, G. Frehse, and E. Schnieder, editors, *Modelling, Analysis, and Design of Hybrid Systems*, Lecture Notes in Control and Information Science, pages 173–192. Springer, 2002. (This volume).
41. T. Park and P.I. Barton. Implicit model checking of logic based control systems. *AIChE Journal*, 43(9):2246–2260, 1997.
42. T. Pawletta, B. Lampe, S. Pawletta, and W. Drewelow. A DEVS-based approach for modeling and simulation of structure dynamics in hybrid systems. In S. Engell, G. Frehse, and E. Schnieder, editors, *Modelling, Analysis, and Design of Hybrid Systems*, Lecture Notes in Control and Information Science, pages 107–130. Springer, 2002. (This volume).
43. J. Preußig. *Formale Überprüfung der Korrektheit von Steuerungen mittels rektangulärer Automaten*. PhD thesis, Department of Chemical Engineering, University of Dortmund, Germany, 2000. (in German).
44. J. Preußig, S. Kowalewski, T.H. Henzinger, and H. Wong-Toi. An algorithm for the approximate analysis of simple rectangular automata. In *Proc. 5th Int. School and Symposium on Formal Techniques in Fault Tolerant and Real Time Systems, Lyngby, Denmark, 1998*, Lecture Notes in Computer Science 1486, pages 228–240. Springer, 1998.
45. J. Preußig, O. Stursberg, and S. Kowalewski. Reachability analysis of a class of switched continuous systems by integrating rectangular approximation and rectangular analysis. In Frits W. Vaandrager and Jan H. van Schuppen, editors, *Hybrid Systems: Computation and Control, Proc. 2nd Int. Workshop, HSCC'99, Berg en Dal, The Netherlands, March 1999*, Lecture Notes in Computer Science 1569, pages 209–222. Springer, 1999.
46. J. Preußig and H. Wong-Toi. An procedure for the reachability analysis of rectangular automata. In *Proc. American Control Conference*, pages 1674–1678, 2000.
47. J. Raisch and S. O'Young. Discrete approximation and supervisory control of continuous systems. *IEEE Trans. Automatic Control*, 43(4):569–573, 1998.
48. M.A. Pereira Remelhe, S. Engell, and M. Otter. An environment for integrated object–oriented modeling of systems with complex continuous and discrete dynamics. In S. Engell, G. Frehse, and E. Schnieder, editors, *Modelling, Analysis, and Design of Hybrid Systems*, Lecture Notes in Control and Information Science, pages 83–106. Springer, 2002. (This volume).
49. J.M. Schumacher, A.S. Morse, C.C. Pantelides, and S. Sastry, editors. *Special Issue on Hybrid Systems*, volume 35 of *Automatica*, March 1999.
50. C. Simon, K. Lautenbach, H.-M. Hanisch, and J. Thieme. Using parameterized timestamp petri nets in automatic control. In S. Engell, G. Frehse, and

E. Schnieder, editors, *Modelling, Analysis, and Design of Hybrid Systems*, Lecture Notes in Control and Information Science, pages 211–224. Springer, 2002. (This volume).
51. O. Stursberg. *Analyse gesteuerter verfahrenstechnischer Prozesse durch Diskretisierung*. PhD thesis, Department of Chemical Engineering, University of Dortmund, Germany, 2000. (in German).
52. O. Stursberg and S. Kowalewski. Approximating switched continuous systems by rectangular automata. In *Proc. European Control Conference*, 1999. CD-ROM, file 1014–4.
53. O. Stursberg and S. Kowalewski. Analysis of controlled hybrid processing systems based on approximation by timed automata using interval arithmetics. In *Proc. 8th IEEE Mediterranean Conference on Control and Automation*, 2000. CD-ROM, file TA1–3.
54. O. Stursberg, S. Kowalewski, and S. Engell. On the generation of timed discrete approximations for continuous systems. *Mathematical and Computer Modelling of Dynamical Systems*, 6(1):51–70, 2000. Special Issue on "Discrete Event Models of Continuous Systems".
55. O. Stursberg, S. Panek, J. Till, and S. Engell. Generation of optimal control policies for systems with switched hybrid dynamics. In S. Engell, G. Frehse, and E. Schnieder, editors, *Modelling, Analysis, and Design of Hybrid Systems*, Lecture Notes in Control and Information Science, pages 337–368. Springer, 2002. (This volume).
56. H. Treseler. *Ein Rechnerwerkzeug zur formalen Verifikation diskret gesteuerter verfahrenstechnischer Prozesse*. PhD thesis, Department of Chemical Engineering, University of Dortmund, Germany, 2001. (in German).
57. F. Vaandrager and J. van Schuppen, editors. *Hybrid Systems – Computation and Control, Proc. 2nd Int. Workshop HSCC'99, Berg en Dal, The Netherlands, March 1999*, volume 1569 of *Lecture Notes in Computer Science*. Springer, 1999.
58. A. van der Schaft and H. Schumacher. *An Introduction to Hybrid Systems*, volume 251 of *Lecture Notes in Control and Information Science*. Springer, London, 2000.
59. K. Wolter, A. Zisowski, and G. Hommel. Performability models for a hybrid reactor system. In S. Engell, G. Frehse, and E. Schnieder, editors, *Modelling, Analysis, and Design of Hybrid Systems*, Lecture Notes in Control and Information Science, pages 193–210. Springer, 2002. (This volume).
60. S. Yovine. Kronos: a verification tool for real-time systems. *Software Tools for Technology Transfer*, 1(1,2):123–133, 1997.
61. Janan Zaytoon, editor. *3rd Int. Conf. on Automation of Mixed Processes: Hybrid Dynamic Systems (ADPM'98), Reims, France*. Universit de Reims, 1998.