Hybrid Systems in Process Control: Challenges, Methods and Limits

Stefan Kowalewski *
Corporate Research and Development
Robert Bosch GmbH †
Frankfurt, Germany

Abstract

Hybrid dynamics in process control arise for various reasons. The main source is the interaction between computer-realized discrete control functions and the continuous physico-chemical processes in the plant. But also physical or operational constraints lead to discrete phenomena in otherwise continuous processes. The modeling, the analysis and the design of hybrid control systems pose new problems for which the conventional systems and control theory is not an appropriate tool. The paper identifies these challenges using illustrative case studies and gives a survey of recent approaches to a formal treatment of hybrid systems. Particular emphasis is put on methods which were developed in computer science in the last years. We will discuss to what extent these techniques can support the analysis and design of hybrid systems in process control or automation and what the limits are.

Keywords

Hybrid systems, Hybrid automata, Safety analysis, Verification, Synthesis, Fault diagnosis

Introduction

The term "hybrid systems" was coined around ten years ago and refers to systems with both continuous and discrete dynamics. It is now a common research field in control theory and computer science with regular conference and workshop series. The purpose of this paper is to present a survey on approaches from this field and discuss their potential for solving problems in process control or automation. In particular, we will focus on models and methods from computer science research which do not receive much attention from the process control community.

In computer science, the treatment of hybrid systems belongs to the field of *formal methods*. The term formal method refers to approaches which systematically use means of description with rigorously and consistently defined syntax and semantics to represent design specifications or existing designs/plants for the purpose of either proving that certain design requirements hold or automatically synthesizing designs which provably meet these requirements. Examples of methods which in computer science are commonly agreed to be formal are *verification* or *synthesis*, whereas *simulation* (in the engineering sense of the word) or *optimization* usually are not regarded as being formal.

In the last years formal methods have found their way from academia to industry in several domains. Applications are reported, for instance, from the design of integrated circuits, communication protocols, embedded controllers for airplanes and cars, logic control of machine tools, and flexible manufacturing systems. The use of formal techniques in these fields is motivated by the need for discovery of faults in early design stages which helps to save costs and to achieve more reliable systems. However, this development does not seem to have reached the processing industries because from this domain only a few academic projects and hardly any industrial application of formal methods are known. The reason which is often given for this situation is that the first formal methods were applicable only to purely discrete problems (e.g., correctness of switching circuits) whereas problems in processing systems usually involve plants with continuous dynamics. Now that new formal methods have been developed for hybrid systems, it seems to be worthwhile to revisit the question of what can be achieved by formal methods in process control or automation. This is the motivation of this contribution.

The paper is organized as follows. In the following section we will present a pragmatic definition of a hybrid system and discuss why hybrid models arise. After that, the hybrid automaton model is introduced with the help of an example. In the section *Analysis of Hybrid Systems* reachability analysis is presented as the main analysis method. We discuss computational issues and the need for abstraction which arises as a consequence from the computational limitations. Based on these theoretical contemplations, the section *Hybrid Systems Problems* presents an analysis of the status and and the potential of formal methods for hybrid systems in the processing industries. Four possible application domains are identified, the current practice in these fields is described, existing academic initiatives for developing methods and tools are reported, and reasons are discussed why the practitioners are still reluctant to use them. A discussion concludes the paper.

What are Hybrid Systems?

The definition of a hybrid system as a system which combines continuous and discrete dynamics is a bit superficial. To be more precise, the term "hybrid systems"

*Robert Bosch GmbH, FV/SLD, P.O. Box 94 03 50, D-60461 Frankfurt, Germany. Email: stefan.kowalewski@de.bosch.com

†The results and opinions expressed in this paper were developed while the author was with the Department of Chemical Engineering at the University of Dortmund, Germany.

refers to models, not systems as such. A system is not hybrid by nature, but it becomes hybrid by modeling 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. For example, it would be possible to describe the behavior of on/off valves by a continuous relation between the opening ratio and the control voltage. But in most situations it is sufficient to consider on/off valves as discrete switching elements. And if these valves interact with a process in which continuous phenomena are of interest, a hybrid model would be appropriate. In other words, it depends on the level of abstraction which is needed to solve a particular problem, whether the model should be discrete, continuous or hybrid.

As a consequence of this definition, hybrid systems arise in processing systems whenever both abstraction levels—continuous and discrete—have to be considered. The main source of such problems are constellations in which discrete (or logic) controllers interact with continuous physico-chemical processes. The need for the discrete abstraction level comes from the man-made control functions (but also from physical constraints), and the continuous level is necessary to describe the processes driven by natural laws. However, to stress again the fact that the appropriate model depends on the purpose: not every computer-realized controller has to be described by a discrete model, and not every chemical process needs a continuous model. If we consider continuous controllers being implemented on a digital computer, it usually is sufficient for the design to describe controller and plant as continuous systems and to abstract from the discrete implementation. On the other hand, for the analysis of a logic control function it is customary to use a discrete abstraction of the continuous process (e.g., only distinguishing between specified and unspecified process states).

The second example points to a main topic in hybrid systems research: abstraction methods. Often, it is either not possible or not appropriate to analyze a hybrid model. In this case it is often helpful to discretize the continuous part and solve the problem using discrete analysis techniques. So, hybrid systems research is not only concerned with hybrid models and their analysis, but also with the problem how to map hybrid problems into spaces where they can be solved better. This will be discussed in more detail in the section *Analysis of Hybrid Systems*. The section *Hybrid Systems Problems* will present examples for hybrid problems in the design and analysis of processing systems. There we will see that it is not only discrete logic but also the human distinction between certain modes (e.g., safe, unsafe, or tolerable process states; start-up, production, or shut-down phases; specified, unspecified, or possibly deviated process behaviors) which leads to hybrid models.

Modeling of Hybrid Systems

Two Different Starting Points and Directions

Modeling frameworks for hybrid systems were introduced independently in control theory and computer science. Since the original domains of interest in these two fields were on both ends of the hybrid dynamics spectrum—purely continuous dynamic systems in control theory, discrete state systems in computer science—the modeling approaches moved in opposite directions:

- In control theory, the starting point was continuous models which were then extended by discrete mechanisms like switching or resetting. The resulting models consist of differential equations, algebraic equations and/or inequalities with continuous and binary variables. The later 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 often refered to as *switched continuous systems*.
- The computer scientists came the other way. They extended discrete formalisms (most prominently finite automata, but also Petri nets and logics) by continuous variables which evolve according to differential equations associated with discrete states. Discrete transitions can switch between continuous modes, and the continuous variables can be reset when a transition takes place. The resulting framework is called *hybrid automata*.

In principle, both modeling approaches are equivalent in the sense that the models are equally expressive[1]. In the following, we will focus our attention on the hybrid automata paradigm from computer science and discuss their potential in process control and automation. For more information about the "control theory approach" to hybrid systems the reader is referred to the literature, in particular to recent special issues on hybrid systems in various control journals (e.g., Antsaklis and Nerode, 1998; Morse et al., 1999; Antsaklis, 2000), or to Lemmon et al. (1999).

It should be mentioned that timed discrete event systems (i.e. models which consist of a discrete transition system and time as the only continuous variable) are often not regarded as hybrid systems but rather classified as discrete event systems. If this distinction is made, systems are called hybrid only when more complex differential equations than $\dot{x} = 1$ are involved. In this survey we will include the application of timed models as part of the general discussion of hybrid systems, because time is a continuous variable, and in many cases, timed models are a sufficient abstraction of hybrid dynamics.

[1] Of course, certain assumptions have to be made. For example, the value sets of the discrete variables in the switched continuous model have to be finite because the discrete state space of a hybrid automaton is finite by definition.

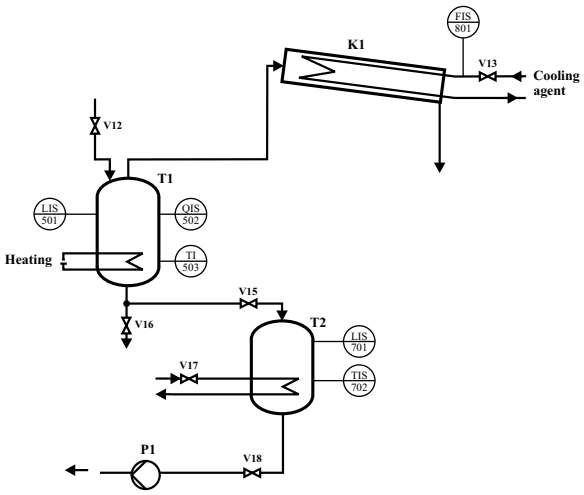

Figure 1: Example.

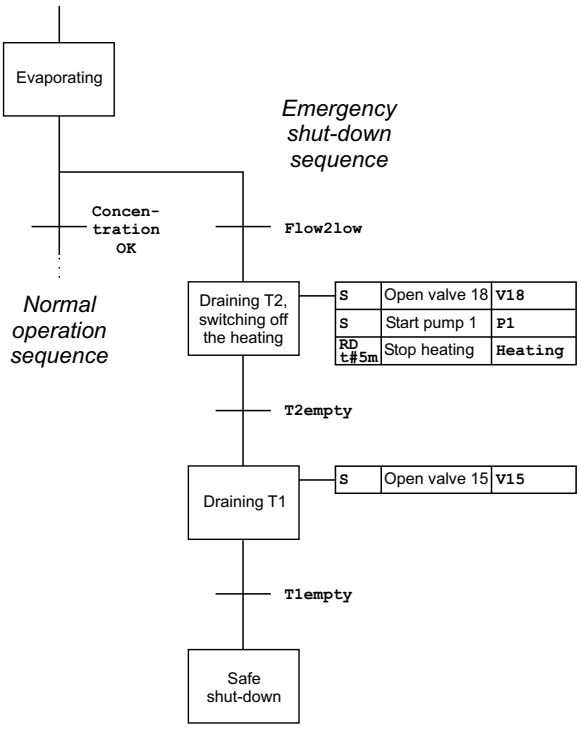

Figure 2: Sequential function chart.

An Example

To introduce hybrid automata we will use a simple example which illustrates that this modeling approach arises naturally for certain classes of process control or automation problems. The example is taken from Kowalewski et al. (2001) and has been used as a benchmark case for instance in the European Union research project *Verification of Hybrid Systems* (VHS, 2000).

Figure 1 shows the piping and instrumentation diagram of the example plant. It is a batch evaporator in which the following production sequence takes place. First, a solution is filled into tank T1 and the solvent is evaporated until a desired concentration of the dissolved substance is reached. During the evaporation stage, the condenser K1 is in operation and collects the steam coming from T1. When the desired concentration is reached, the material is drained from T1 into T2 as soon as T2 is available (i.e., emptied from the previous batch). A post-processing step then takes place in T2, before the material can be pumped out of T2 to a downstream part of the plant.

We focus our attention on the problem of an appropriate reaction of the controller to a cooling breakdown in the condenser. This failure will lead to a temperature and pressure increase in the condenser tube K1 and the evaporator tank T1, if the evaporation process is continued. It must be avoided that the pressure in K1 will rise above a dangerous upper limit. To achieve this, the heating in T1 has to be switched off before the safety pressure valve is triggered. This, in turn, causes a decrease of the temperature of the material in T1. When the temperature in T1 becomes too low, a crystallization effect leads to precipitation of solids, which spoils the batch. This, of course, is an undesired situation, too. Thus, the timespan between the cooling failure and switching-off of the heating is critical: it has to be short enough such that the pressure increase is limited but on the other hand as long as possible such that crystallization will not occur. Any given control program for this process has to be checked against this specification.

Figure 2 shows a suggestion for a controller which shall realize the required behavior. The representation in Figure 2 is a Sequential Function Chart (SFC) according to the IEC 1131 standard (IEC, 1992). In particular, action blocks are used to specify the control actions performed in each step. An action block consists of a qualifier, an action name, and a manipulated variable. The left branch of Figure 2 represents the normal operation sequence. In the undisturbed case, the system will leave the step *Evaporating* when the desired concentration is reached. The part branching to the right describes the control actions during an emergency shut-down as a reaction on a cooling breakdown. There is a waiting time implemented between the cooling failure and stopping the heater. As discussed above, the controller will open valve 18 and start pump 1 to drain T2 as soon as the flow of cooling agent is too low. The corresponding actions are labeled with the qualifier "S", which means that the variables V18 and P1 are set to TRUE and that they will remain TRUE after the step was left. The label "RD t#5m" on the third action in the step *Draining T2, switching off the heating* symbolizes that the heating is switched off with a delay of 5 minutes, even if the step has become inactive in the meantime. As soon as T2 is empty, the SFC will switch to the subsequent step and

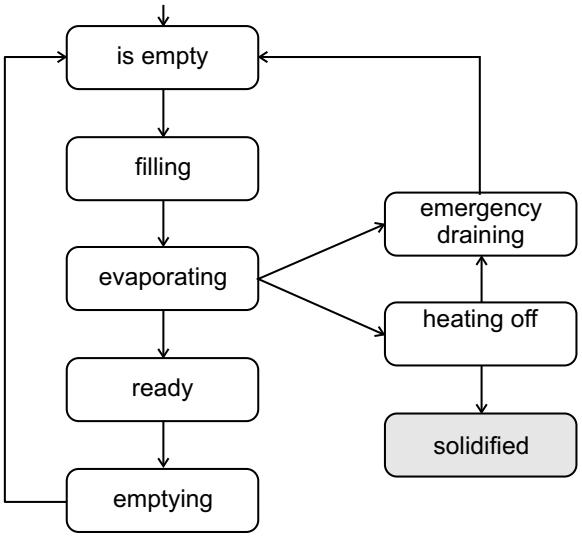

Figure 3: Finite automaton model of the evaporator tank T1.

open valve 15. When T1 is empty and the heating is switched off, the controller assumes the system to be in a state of safe shut-down.

Hybrid Automata

When we want to analyze whether the controller in the example from the previous subsection meets the requirements (i.e., the pressure will never become too high and crystallization will not occur), a model of the plant is needed which represents the relevant behavior. Obviously, a large part of this behavior is discrete because the interaction between controller and plant is specified in discrete terms. Figure 3 shows a corresponding finite automaton model of tank T1 of the evaporator. The discrete states represent the discrete steps during the evaporation cycle (*is empty, filling, evaporating, ready, and emptying*) as well as three important modes in the case of a cooling failure (*emergency draining, heating off, and solidified*). When the failure occurs, T1 will be in the state *evaporation*. The controller will then empty tank T2 and start the clock. This is not captured by the model of Figure 3 but happens in the environment (which would have to be modeled for a formal analysis, too). If T2 is empty before the waiting time is elapsed, the system will move to *emergency draining*, stop the heating, and safely go back to *is empty*. The more interesting case is when the waiting time is elapsed before T2 becomes empty. In the model of Figure 3 this means that the system moves to the state *heating off*. In this state the temperature in T1 decreases while the controller waits for T2 to become empty. Now the question is whether T2 will become empty before the temperature drops below the crystallization threshold. If this is the case, the transition to *emergency draining* will be taken and the system is safe. If not, the state *solidified* will be reached, representing that the crystallization has started.

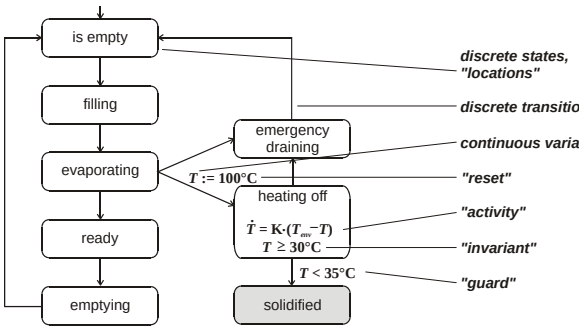

Figure 4: Example of a hybrid automaton.

Obviously, a purely discrete model like the one above is not able to answer this question. What is needed here is the information about how fast the temperature in T1 decreases. Thus, a hybrid model is necessary. Figure 4 shows how the additional information can be added to the discrete model by using a hybrid automaton (Alur et al., 1995). Roughly speaking, the hybrid automaton model complements finite automata by continuous variables. These variables can be reset by discrete transitions (here: the temperature is defined to be 100° Celcius when the heating is switched off). While the system is in a certain discrete state (called *location*), the continuous variables evolve according to differential equations, called *activities* (here: a linear first order ODE). Conditions can be formulated which have to be true while the system remains in a discrete state. They are called *invariants*. Finally, so-called *guards* represent conditions for transitions between discrete states. In the example the invariant and the guard together express that the transition to *solidified* will occur between 35° and 30° Celcius.

Figure 5 shows an important special class of hybrid automata, a so-called *timed automaton* (Alur and Dill, 1990). Here, the continuous variable is a *clock* which can only be reset to zero and the value of which increases by the rate of one. Such a description is useful for example when detailed continuous models are not available or not necessary, but information about process durations is known.

Analysis of Hybrid Systems

The major analysis procedure for hybrid automata is the *reachability analysis*. It answers the question whether for a given hybrid automaton a certain hybrid state (discrete location and region in the continuous space) is reachable from the initial state. This problem is so important because many problems can be reduced to a reachability problem. For instance in the example above, the question whether the controller is correct is mapped into the

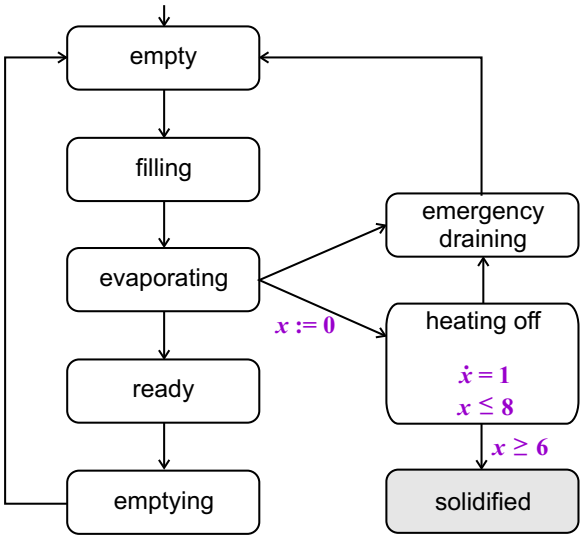

Figure 5: Example of a timed automaton.

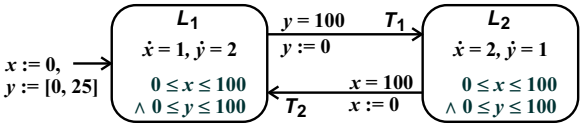

Figure 6: Example for reachability analysis.

problem whether the state *solidified* is reachable in the plant model.

However, the application of reachability analysis for hybrid automata is restricted due to computational issues. We will illustrate the basic computational problem by the small example in Figure 6 which is taken from Preußig (2000). It is a very simple hybrid automaton with only two locations and two continuous variables, x and y. In both locations, the invariants restrict the values of x and y to a square of the size $(0 \leq x \leq 100 \wedge 0 \leq y \leq 100)$. In location L_1, x grows by a rate of 1 and y by a rate of 2. In location L_2 it is vice versa. The transition T_1 can only be taken when the guard $y = 100$ is true. When it is taken, y is reset to 0. Transition T_2 is guarded by the condition $x = 100$, and the reset is $x := 0$. Since the invariants, guards and resets are linear expressions and the solutions of the differential equations for the activities are linear functions, this example belongs to the class of *linear hybrid automata* (Alur et al., 1995). It is the largest class for which tools for exact reachability analysis exist. The most prominent is *Hytech* (Henzinger et al., 1997). *Hytech* uses polyhedra as the data structure for representing, manipulating and storing regions in the continuous state space during the exploration. The basic reachability algorithm is the following:

0. Initialize hybrid automaton (location $l :=$ initial location, polyhedron $P :=$ initial continuous region P_0).
1. Intersect P with the invariant of l.
2. Let P increase with time according to the activity of location l.
3. Intersect P with the invariant of l.
4. Stop, if l was visited before with P.

   Else: For all transitions T from l do:

   5. Intersect P with the guard of T.
   6. Reset P according to the reset expression of T.
7. Set $l :=$ target location of T, go to step 1.

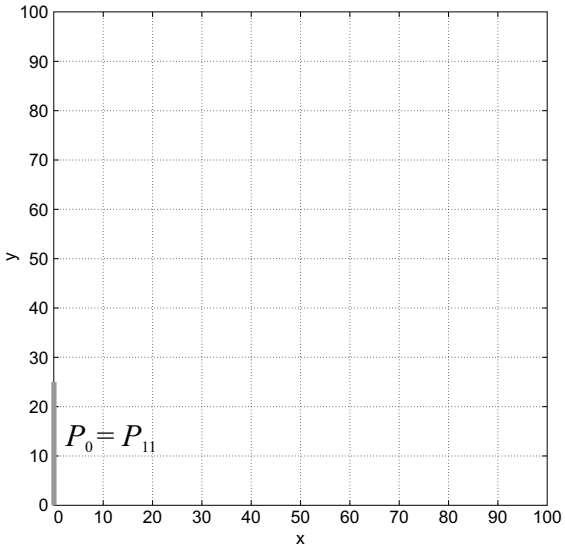

Figure 7: Situation after step 1 of the reachability analysis algorithm.

Figure 7 shows the result of the steps 0 and 1 of the algorithm: The initial values of x and y are $(x = 0 \wedge 0 \leq y \leq 25)$. Step 1 determines which of these values are possible in location L_1. In this case the set is unchanged and represented by the polyhedron P_{11} (the dark grey bar in Figure 7) The subscripts symbolize iteration 1 and step 1. In step 2, the region of the continuous state space is computed which can be reached while the system is in l, without considering the invariant (or, in other words, assuming that the system remains in L_1 forever). The result is P_{12} in Figure 8. Obviously, not all of this region is actually reachable, because the invariant will force the system to leave L_1 as soon as it is violated. Step 3 determines the part which is in accordance with the invariant, i.e. P_{13} in Figure 9.

At this stage, the algorithm would take the list of previously visited regions and check whether P_{13} (or a subset of it) had been computed for L_1 before. If this is case,

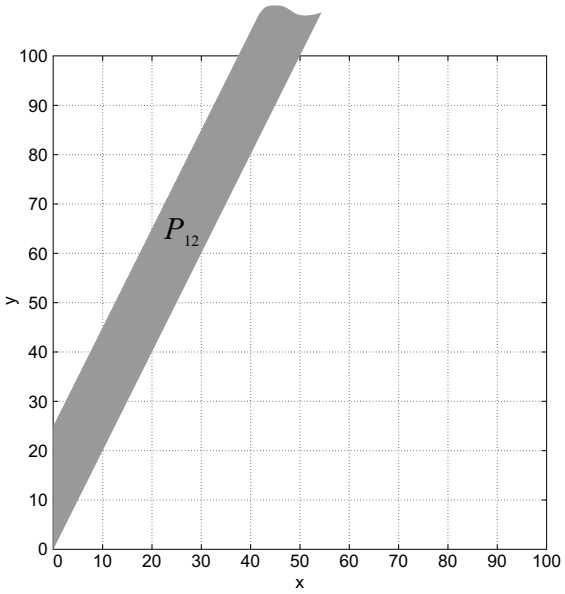

Figure 8: Result of step 2.

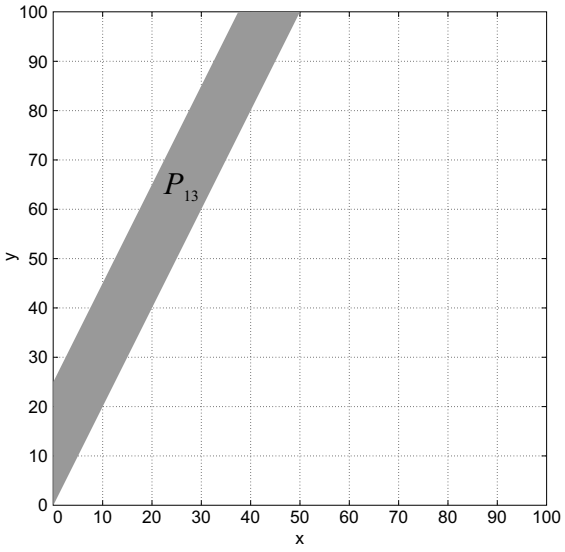

Figure 9: Result of step 3.

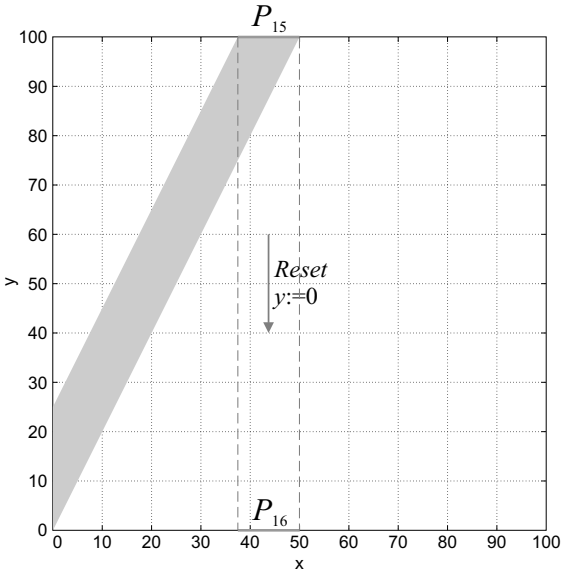

Figure 10: Results of step 5 and 6.

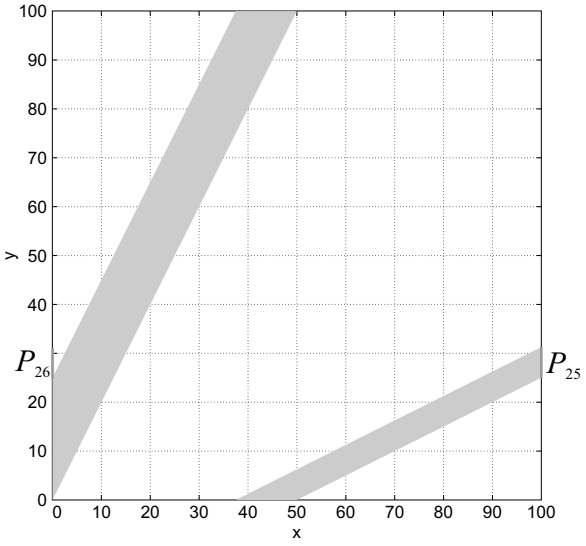

Figure 11: Result of iteration 2.

it would abort the current search branch. If not, the algorithm has to determine the possible transitions from L_1. In our example, there is only T_1. To find out whether it is possible, we have to check whether the guard can become true for the reachable values of x and y in L_1. This is computed in step 5 by the intersection. If it is empty, the transition is not possible. Here, the result is P_{15} in Figure 10. When the transition is taken, the reset leads to P_{16}. This is the possible region with which the system can enter location L_2. Now, the second iteration starts. Figure 11 shows the resulting polyhedra P_{25} and P_{26} (the light grey parts represent the already visited region).

If we continue the algorithm, it will become apparent that after each iteration the reachable set is increased by a polyhedron which becomes smaller and smaller. In fact, the length of the horizontal and vertical boundary lines is half of the length of the previous iteration. This converges asymptotically to the reachable set given in Figure 13. The consequence is that the algorithm will not terminate. This is not only true for this example—it was proven that reachability for linear hybrid automata is not decidable, which means that there does not exist

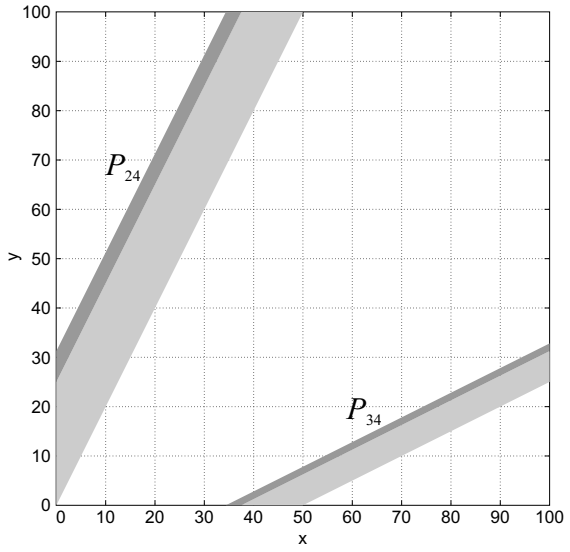

Figure 12: Results of iterations 3 and 4.

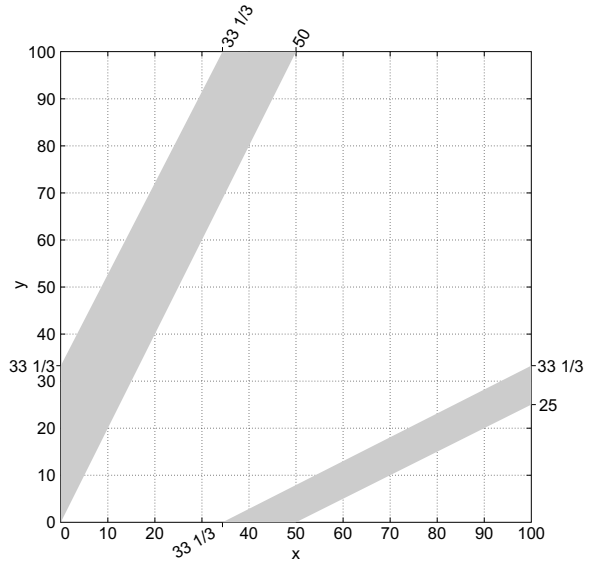

Figure 13: Reachability set.

an algorithm which will terminate for every linear hybrid automaton (Alur et al., 1995).

In practice, this undecidability result is not the only problem. Since *HyTech* is performing its computations by integer arithmetic, memory overflows for the integer variables are a common problem. A further shortcoming of this approach is the exponential complexity of the algorithm with the number of continuous variables (which adds to the discrete state space explosion problem). These shortcomings are the motivation for a very active area in hybrid systems research which is concerned with abstraction techniques for hybrid systems. The idea is to build a substitute model which is more abstract than the hybrid system under investigation (that is, it omits some details but includes the behavior of the original system), which belongs to a class of systems for which reachability is easier to solve (e.g., timed automata or purely discrete systems). A collection of papers following this approach can be found in Engell (2000). Alur et al. (2000) provided a survey on fundamental theoretical results in discrete abstractions of hybrid systems. A slightly different approach is not to do the abstraction for the whole dynamics first and then analyse reachability, but to apply approximation techniques during each single iteration step of the reachability analysis. Main representatives of this work are for instance Chutinan and Krogh (1999), Dang and Maler (1998), Greenstreet and Mitchell (1999), Raisch and O'Young (1998), and Preußig et al. (1999).

Hybrid Systems Problems

In this section we identify a list of process control problems (or in a wider sense, process automation problems) which pose challenges to formal methods for hybrid systems. The problems are presented in the order in which they arise during the life cycle of a processing plant. The project management for processing plants is built on the following life cycle model. Starting with a market analysis, the early phases take place in laboratories and pilot plants to increase the knowledge, identify the optimal chemical or physical processes for realising the desired product and estimate the economic prospects. After that, the so-called *basic engineering* begins during which the unit operations (e.g. reactions, separations) are fixed, the corresponding types of equipment are chosen, and the basic means for controlling the plant are specified. The major resulting document of this phase is the so-called *Piping and Instrumentation Diagram* (P&ID). This is a flowsheet representing all pieces of processing equipment like tanks, valves, reactors etc., all the pipes, the measurement points, as well as the continuous control loops and the safety trips (the latter two being represented roughly as connections between the initiating sensor and the corresponding actuator). Following the basic engineering is the *detail engineering* phase. Here, the information of the P&ID is specified and refined to obtain the necessary documents for *erection* (e.g., pipe routing, equipment sizes, choice of materials etc.) and *commissioning* (choice of sensors and control systems, configuration/programming of the controllers, design of operator panels, etc.). Often the boundaries between these phases are fuzzy. It is not uncommon, for example, that the programming of the control system has to be finished on site while the plant is being put into operation. After the commissioning, *operation and maintenance* is the next phase, and *decommissioning and dismantling* is regarded as a phase

of its own which finishes the life cycle of a processing plant. Refering to this life cycle model, the following tasks with the potential for the application of hybrid systems methods can be identified.

- safety analysis of plant instrumentations (*basic engineering*)
- design and analysis of discrete controllers, e.g., interlocks, trips, switching of continuous control modes, sequence control (*detail engineering, commissioning*)
- generation and analysis of control recipes for batch processes (*operation*)
- event-based fault diagnosis (*operation and maintenance*)

In the remainder of this section, the four tasks will be discussed in more detail.

Safety Analysis of Plant Instrumentations

Task Description. Many processing plants represent a potential danger for the life and health of the people working in the plant and those living in their vincinity and for the environment. Therefore, in most countries the national authorities demand a thorough and systematic analysis of the possible hazards and the safety concept of a new plant. This procedure has to take place at the end of the basic engineering phase because it is important to identify safety-related design faults in an early stage of the project, so that modifications can be done with little effort. The aim of the safety analysis is to discover potential sources of hazards and to check that the proposed safety devices are capable of handling these hazards in an appropriate way.

Clearly, many safety problems are of a hybrid nature. It often depends on the interaction between discrete safety devices (e.g., limit switches, relief valves) and the continuous dynamics (e.g., an exothermal reaction) whether dangerous situations can occur or whether the instrumentation can prevent them.

Current Practice. In practice, the safety analysis is carried out by a team of experts coming from different disciplines involved in the plant design. These experts get together and discuss the safety concept based on the P&ID diagram. They literally start at one end of the plant and move from one piece of equipment to the other. Disturbances and equipment failures are assumed and the consequences are estimated. The results are documented thoroughly. This procedure can take weeks.

To provide guidelines for these discussions, companies and authorities have developed a handful of systematic methods for safety analyses. Among those the Hazard and Operability Studies (HAZOP) methodology (Lawley, 1974) is now established as the most popular one. In HAZOP, a list of guidewords (e.g., "none", "more than", or "reversed") is applied to each piece of equipment and each relevant physical variable such that all possible deviations of the nominal values can be determined. However, even for a medium sized plant it is still impossible to consider all cases being brought up by the HAZOP method. The experts have to select what they think are the most important cases. This bears the danger of missing a possible hazard. Another problem is that hazard analyses are based on the "one failure at a time" assumption. This is also done to keep the number of considered cases manageable. However, hazards caused by combinations and sequences of failures will not be discovered this way. Finally, the consequences of the interaction between continuous and discrete dynamics can only be estimated. This can be problematic because responses of hybrid systems are much less predictable than purely continuous systems.

State of Research. Since at the time of the safety analysis the available information about the plant behavior is rough, a model-based analysis will have to build on qualitative models which abstract from the continuous dynamics. First approaches in this direction were presented by Vaidhyanathan and Venkatasubramanian (1995), Yang and Chung (1998) and Graf and Schmidt-Traub (1998). In all cases, the idea is to follow the HAZOP methodology and to provide an automated checker for the guideword questions which uses a quantitative plant model. In Vaidhyanathan and Venkatasubramanian (1995) and Yang and Chung (1998) the model is built on signed directed graphs, in Graf and Schmidt-Traub (1998) state transition systems are used which are specified and simulated using the tool *Statemate* (Harel and Naamad, 1996). In the latter case, the HAZOP method essentially is mapped to a reachability problem for discrete state transition systems. This points to the similarity of the safety analysis for qualitative models and formal verification of data processing systems (see next section). It can be expected that more approaches from this field will be applied to hazard identification and safety analysis in the future, including the application of the hybrid analysis method presented in the first part of this paper.

Discussion. From the description of the currently applied procedures for hazard and safety analysis it becomes clear that there is a potential for improving the current practice by applying formal methods in the sense of an automated, rigorous model-based safety analysis which could then overcome the problem of overlooked hazards as well as the single failure assumption.

The approaches mentioned in the previous section are promising because they provide a means to automatically check the consequences of large numbers of deviations or failures and combinations of them. Already during the modeling step, valuable insight will be gained about

the safety of the plant design. And when reachability analysis algorithms are applied, it is guaranteed that every hazard being considered in the model—explicitly or implicitly—will be discovered. However, before these approaches can be applied in practice, two problems have to be solved. First, the modeling effort is very high and for the practitioner it is often not apparent whether it will pay off during the analysis. Second, the automated analysis will result in a large number of scenarios leading to dangerous states differing only in small details which would have been considered as one hazard in a manual analysis. Also, due to the intrinsic nondeterminism of qualitative models, many "dangerous" trajectories are physically impossible. To overcome these problems, methods for filtering the analysis results before they are presented to the user have to be developed.

Design and Analysis of Discrete Controllers

Task Description. In the process industries, discrete controllers are realized by programmable logic controllers (PLCs) or distributed control systems (DCSs). They perform basic control functions, e.g., process supervision to avoid unwanted or even dangerous states of the process or damage to the equipment, sequence control, startup and regular or emergency shutdown procedures, switching the mode of operation of PID controllers, or supervision of sensor inputs and outputs to actuators. A significant part of this discrete control logic is critical for the safe operation of the process. The correctness of the logic control system often depends on the interaction between the discrete control function and the continuous process.

Current Practice. In practice, discrete control logic is still very seldom produced in a systematic manner. Usually, rough and often incomplete specifications lead to a first design and direct implementation which is then improved by testing onsite. The support of the software design is restricted to programming environments and a standard for programming languages and common standard software elements (IEC (1992), see Maler (1999) for a critical analysis). This situation seems to be inappropriate in particular for the citical parts of the logic control software.

State of Research. The research on a more systematic and reliable design for logic controllers in processing systems follows a number of approaches. Among those, the following three can be regarded as the most important methods with a formal basis: verification, synthesis, and code generation from a formal specification.

Formal Verification. The notion of formal verification originates from computer science where, in general terms, it means a mathematical proof that a model of an algorithm fulfills given formal properties. In the last decades, different representations and methods have been developed and, in the recent years, some of them have been applied successfully in the area of hardware and communication protocol design (see Clarke and Kurshan, 1996, for a survey). However, in the field of logic controllers for processing systems, in particular when continuous processes are involved as it is the case for most processing plants, formal verification is currently not applied in practice and only a few research projects are described in the literature (see below).

The pioneering work in the verification of logic controllers for industrial and in particular, chemical processing systems was done by Powers and his co-workers (1992) for control programs represented in Relay Ladder Logic. They applied the symbolic model checking method from Burch et al. (1992) in which the system that has to be verified is modeled as a finite state machine and the specifications of the desired behavior are represented by temporal logic expressions. Their approach has been extended to include plant models and recent work shows that the formal verification of logic controllers for processes of moderate size is possible (Probst et al., 1997). The continuous dynamics of subsystems of the plant are discretized in an elementary manner: The range of values of each continuous variable is partitioned into intervals and the model simply describes the possible transitions between these intervals. Timers are incorporated in the same qualitative fashion by neglecting the timer value and keeping only two states, running and elapsed. This suffices if checking real-time constraints is not required, e.g., whether a controller response to a plant event is fast enough to avoid unwanted process behavior.

In the last years, the model checking approach has been extended to incorporate real-time and hybrid specifications and analysis. This work is mostly based on the timed or hybrid automata model and it resulted in the development of analysis tools for such systems (Yovine, 1997; Larsen et al., 1997; Henzinger et al., 1997). Applications of this framework to the modeling and analysis of PLC programs are reported, for example, in Mader and Wupper (1999) and VHS (2000). In Kowalewski et al. (1999) an approach is presented which integrates a number of available model checking tools from computer science with an engineering-oriented modeling interface. For this purpose, a modular, block-oriented framework for modeling hybrid systems is introduced which is based on hybrid automata connected by certain discrete valued signals. It aims at applications where formal verification requires a quantitative analysis of the interaction between timers or threshold values in the logic control program and the plant dynamics. Further examples of verification approaches to PLC programs including time or hybrid aspects are Herrmann et al. (1998) and Heiner et al. (1999).

A recent development in the chemical engineering community is represented by approaches to apply tech-

niques from optimization, in particular mathematical programming, to verification problems. For instance, the same basic problem as in the approach by Moon et al. (1992), i.e. model checking of Relay Ladder Logic diagrams, has been treated by Park and Barton (1997) using these techniques. Here, the problem of checking whether a temporal logic formula holds for the model of the logic controller is mapped into the feasibility problem for a system of equalities and inequalities for binary variables. The latter is then solved by integer programming. Application results show that there exist examples for which this approach generates solutions faster than classical model checking. The problem of abstracting continuous or real-time information is solved in the same way as in the first approach.

The use of mathematical programming not only for checking the control logic but also for the analysis of switched continuous models has been suggested by Dimitriadis et al. (1996, 1997). 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 suggested by Bemporad and Morari (1999). 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. The verification method is part of a comprehensive modeling and analysis approach to hybrid systems, including a scheme for model-predictive control which is reported in another paper of the CPC VI.

Synthesis. The synthesis of logic controllers is a transformation of the continuous control synthesis problem to a discrete event or hybrid setting: Given a model of the possible plant behavior for arbitrary control inputs and a specification of the desired process behavior, a controller has to be designed which guarantees that the closed-loop system satisfies the requirements. When the term synthesis is used in the research literature for this task, it is usually understood that the controller is derived from the plant model and the specifications *automatically* by an appropriate algorithm. Due to a traditional and deliberate mutual isolation of the control systems and computer science research communities, control engineers refer to the framework set up by Ramadge and Wonham (1989) whereas computer scientists trace the roots back to early game theory (Buechi and Landweber, 1969) when the origins of research on logic controller synthesis are concerned. In both cases, the problem formulation and algorithms are basically equivalent.

In the Ramadge and Wonham framework, the plant is modeled as a finite state machine in which the transitions represent discrete events. They either can be inhibited by the controller or are uncontrollable in the sense that their occurence is not under influence of the environment. The desired behavior can be specified by a set of acceptable event sequences, a so-called *target language* (Ramadge and Wonham, 1987b), or a set of undesired states which corresponds to the forbidden states specification mentioned in the previous section on formal verification (Ramadge and Wonham, 1987a). For both requirements, Ramagde and Wonham present synthesis algorithms with linear complexity refering to the size of the plant model. However, the size of the plant model itself increases exponentially with the number of components from which it is built.

There exist numerous extensions and derivations of this approach from which only a selection can be mentioned briefly here. While in the original approach the complete controller is computed off-line, Chung et al. (1992) introduce an on-line synthesis approach using a limited lookahead horizon. The fact that the controller can only inhibit plant events but not force them to occur is often regarded as a shortcoming of the Ramadge and Wonham framework. In Golaszewski and Ramadge (1987), Sanchez (1996), Kowalewski et al. (1996), and Krogh and Kowalewski (1996) synthesis concepts are investigated where the controller can pre-empt events by forcing other events. A further line of research is concerned with Petri nets as an alternative modeling paradigm. Examples are Hanisch et al. (1997), Li and Wonham (1993, 1994), and Chouika et al. (1998). A survey is presented in Holloway et al. (1997). Applications of formal synthesis approaches for process control problems can be found, for instance, in Tittus (1995) and Marikar et al. (1998).

An important extension is the incorporation of quantitative time, such that additional specifications for meeting deadlines or minimal state residence times can be formulated for the synthesis. This is useful for process control applications because timers are often used when information about the evolution of the continuous process dynamics is not available from measurements. Two main approaches can be distinguished: The first one is based on a discrete time axis and assumes that state transitions only occur synchronously to clock ticks. This leads back to the original untimed Ramadge and Wonham setting (Brandin and Wonham, 1994). The second approach is based on timed automata. Here the time axis is continuous and the problem is formulated as a

game between controller and plant (Asarin and Maler, 1999; Wong-Toi, 1997). Any winning strategy can serve as a specification for a correct controller.

As it was the case for the formal verification, the application of the discrete or timed synthesis methods to process systems requires the substitution of models with continuous dynamics by discrete or timed models. Systematic ways to obtain valid approximations for the synthesis of controllers are reported in Chutinan and Krogh (1999) and Raisch and O'Young (1998).

Code Generation from a Formal Specification. A more pragmatic approach to systematic design of logic controllers is represented by top-down design methods where formal specifications of the control code are automatically translated into control code. In principle, this procedure does not exclude a step in which the specification is checked against problem-specific requirements by formal verification before it is fed into the code generator, or a final proof that the generated program is in accordance with the specification. However, in the pertinent research known from the literature this is not yet done. The PLC code is either generated from the result of a synthesis (see above, Hanisch et al., 1997; Marikar et al., 1998) or the specification is set up by hand and analysed only with respect to general properties which are not specific to the particular control problem, like absence of deadlocks or reversibility of the initial state. The latter approach is mostly based on Petri net representations (Frey and Litz, 1999).

Discussion. It is safe to say that in the field of logic controllers for processing systems, in particular when continuous processes are involved as it is the case for most processing plants, the presented academic approaches for verification or synthesis are currently not applied in practice. There are several apparent reasons for this situation. The first one is that PLC or DCS software is usually developed by engineers and not by computer scientists. Thus, the developers are not familiar with the available formal methods and tools. Another reason is that most control software in this area is written by the PLC or DCS user for one particular application or a rather small number of similar plants. This is a different situation to hardware or protocol design where the software is part of a mass product and the verification effort pays off more easily.

Regarding the prospects of the verification approaches mentioned above, it has to be noted that the application scope is limited by the computational cost of the analysis and, again, the necessary effort to build the models. Therefore, more effort has to be invested mainly to increase the efficiency of verification algorithms and to support the modeling process, possibly by interfaces to already existing process information (P&ID, data sheets).

Generation and Analysis of Control Recipes for Batch Processes

Task Description. Batch processes follow a stepwise production sequence. While in continuous processes the input and output material is continuously flowing into and out of the plant, respectively, batch processes are characterized by the fact that discrete amounts of material, so-called *batches*, are processed one after the other (and possibly in parallel). Often, batch plants are designed for a flexible production of multiple products. Each product has its own processing sequence, the so-called *recipe*, which can be realized in the batch plant usually in more than one way. It is also often the case that multiple batches, possibly of different products, can be produced in parallel.

For the management and control of such plants, the concept of *recipe control* has established itself in industry and is described in the standard (ISA, 1995). The main idea is to assign a *basic recipe* to each product which specifies the necessary production steps and their sequence independently from the specific equipment available in the particular plant. Each step is described on the level of *process operations* like, for instance, mixing, heating, reaction, or separation. When the production management decides that a certain amount of a product has to be produced, the basic recipe is taken and to each process operation an appropriate plant unit (e.g., mixer, reactor, etc.) is assigned. This leads to the *control recipe*, which can be regarded as an executable control procedure for the realization of the basic recipe on a particular batch plant.

The problem arising in this procedure is that the assignment of specific equipment can lead to undesired consequences which are not easy to predict. While the eligiblity of plant units for certain operations (for instance with respect to size, available heating or cooling capacity, resistance of material etc.) can be checked statically, interference with other currently running recipes is much more difficult to analyse. In particular, it is possible that a ressource is assigned which will not be available when it is needed, because it is used by another recipe. While in manufacturing processes this situation usually only causes a delay, it can lead to the spoiling of the batch in a processing plant due to chemical and physical effects during the waiting period (think of cristallization). It is also possible that deadlocks occur due to bad equipment allocation.

Current Practice. Due to the broad use of the recipe control concept in industry, most DCS vendors offer software packages for the integration of recipe management and control into DCS applications for batch plants. These packages support editing of basic recipes, storage of recipes in data bases, and manual assignment of equipment from the controlled plant to steps of the recipes to generate control recipes. However, achieving

correctness of the control recipe in its environment (consisting of the plant and the already generated parallel control recipes) with respect to realizability and avoidance of deadlocks and critical waiting situations like the one described above, is left to the experience and intuition of the user.

State of Research. One solution to the problem described above is to simulate the execution of the control recipes on the corresponding plant. For this purpose, a discrete or hybrid simulator is necessary. While in principle any general purpose simulation package with discrete or hybrid capabilities can be used for this task, there are simulators available which specificly support simulation of recipe-driven batch plants. One example is the tool *BaSiP* (Fritz et al., 1998) which offers the possibility to specify recipes using standardized languages and to build the plant model from pre-defined blocks. For the simulation the user can choose between a discrete and a hybrid simulator or an interface to the commercial *gPROMS* package.

Most of the research in the operation of batch processes is concerned with optimization of production schedules. This problem is situated on a higher level than the one considered here. For the scheduling it is assumed that realizable control recipes are available and the task is to find optimal starting times and sequences to satisfy certain production goals. In this field mathematical programming has become one of the most popular tools. This line of research is mentioned here because it is possible to take operating requirements (like avoiding excessive waiting) into account as constraints in the optimization problem.

A lot of research activities addressing specificly the correctness and realizability of recipes in batch plants is built on timed Petri nets. Here, the pioneering work goes back to Hanisch (1992). Later approaches were reported, for example, in Tittus (1995). To check the correctness, the plant and the recipes are modeled by means of a Petri net and the available analysis techniques for this kind of model are used to determine, for example, deadlocks, reversibilty, or execution times of recipes. Meanwhile these approaches have been extended to incorporate the continuous aspect using so-called *hybrid Petri nets* (David and Alla, 1992).

A new approach to analyse or generate recipes takes advantage of the model checking tools for timed automata (see the previous section in this paper entitled *Design and analysis of discrete controllers* and also Fehnker (1999) and Niebert and Yovine (1999)). Here the idea is to model the recipe and the plant by nondeterministic timed automata. The composition will then yield a model representing any possible allocation of plant units to recipe steps and any conceivable sequence of recipe execution. Using the available reachability analysis algorithms, it is then possible to check whether certain states representing successful execution of a pre-defined number of recipes are reachable within a desired time. This approach can be used for optimization by an iterative procedure during which the desired time is decreased until no more realizable schedules can be found. However, these approaches are in their infancy and currently only applicable to downsized examples with a handful of recipes running in parallel.

Discussion. In the domain of batch processes, a lot of research is carried out and the extent to which research results have been transfered to industrial applications is larger than in the previous cases. However, this applies mostly to the optimization of schedules. The problem of finding a realizable and proper control recipe for a basic recipe and a plant is currently solved manually and not supported by formal methods. The two main reasons are the same as in the case of logic control design: modeling effort and tool performance. Usually, the companies already spent a lot of time to model the batch plant and specify the recipes. And since the mentioned simulation tools require different representations as their input, this means double effort for the industrial users. This is not regarded as worth it, in particular because experienced operators of batch plants often perform a rather good job in creating control recipes and schedules. However, considering the increasing complexity of batch plants, this situation may change and the incentive to automatically generate and analyze recipes and schedules may become stronger.

To make the academic tools more attractive to practioners, interfaces to model representations already existing for the DCS are certainly desirable. Also, the performance of the tools has to be increased. Here, the timed automata approach could gain importance if it will be possible to extend existing compositional reachability algorithms like Lind-Nielsen et al. (1998) to a timed setting.

On the methodological side, it could be fruitful to investigate the similarity between the generation of a control recipe from a basic recipe for a particular plant to the compilation of computer programs written in a high-level language to machine code executable on particular hardware. Concepts from computer science could then be used, for example, to analyze whether a control recipe is a valid "refinement" of the basic recipe, where refinement means adding more detailed information without violating the originally specified behavior.

Event Based Fault Diagnosis

Task Description. If the process variables deviate from their nominal values, usually a step-wise procedure starts before a safety device is actually triggered: First a warning is sent to the operator console, and if the value deviates further, an alarm is generated. This leaves some time to the operating personnel to identify the cause of

the fault and to get the process back into nominal behavior. In practice, discovering the cause of an alarm (or a warning) is based solely on the message associated to an alarm and on the experience and knowledge of the operators. In most cases, this is sufficient. However, it is possible that an alarm already represents an effect of a not directly determinable cause. And it is also often the case that one fault successively causes further deviations which results in a fast and long sequence of alarms on the operator screen. To determine the original fault and act correspondingly in a short time and under stress conditions may become an impossible task for the operator then.

Current Practice. The DCS supports the operating personnel in situations as described above only by identifying the plant unit and process variables which initiated the alarm, and possibly by displaying the current values or recent trends of related process variables. The analysis of these data is left to the operator.

State of Research. Methods for event-based fault diagnosis that automatically (and more or less instantly) determine the possible causing failure events are available from the field of discrete event systems research. Fundamental notions in this context are *observability*, *invertibility*, *testability*, and *diagnosability* of discrete event systems. A comprehensive, tool-supported approach for fault diagnosis of discrete event systems is presented in Sampaath et al. (1996) where a diagnoser can be generated which derives the set of the currently possible (and partly unobservable) plant behavior from the observable events.

Discussion. Tools to automatically determine causes of alarms or estimations of the current discrete plant state can be very helpful to minimize the time of plant shutdowns due to alarm trips and to reduce the risk of wrong interference of the operating personnel in an alarm situation. While these tools are available, their acceptance in industry can be achieved only if appropriate modeling environments will be integrated into the DCSs.

With respect to process systems, it is also desirable that the approaches are extended from a purely discrete to a timed or hybrid setting. This would make it possible to take the process dynamics into account and, for example, determine that certain variables change too fast or too slow by checking whether a limit switch event will occur before or after a certain time threshold.

Conclusions

This paper presented an introduction to formal methods for hybrid systems and discussed their potential with respect to problems in process control or automation. Four application fields were identified in the design and operation of processing systems where hybrid systems methods represent promising tools for a better support of the engineering process. The selection was made by the author, and it is likely that more tasks can be identified which can be supported in the same or similar way.

To summarize, it can be said that for each of these tasks appropriate methods and tools have been made available from academic researchers. However, they are currently not applied in industry. The reasons are the same for each task: First, the existence of the methods is unknown to many practitioners. And if a process engineer gets in touch with a formal method, the often inaccessible nature of its semantics and nomenclature will probably prevent her or him from applying it. If, however, this could not scare her or him away, the huge modeling effort and limited tool performance will destroy the remaining illusions. Of course, this picture is painted excessively bleak and there certainly are promising developments, but it clearly points to the open problems which have to be solved to make formal methods for hybrid systems a well established tool in the process industries.

Acknowledgments

The views presented in this survey were developed while I was a member of the Process Control Laboratory of 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 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 who helped to understand the sometimes very different worlds of process engineering, logic control design and computer science. The research projects on this topic in which I could participate have been funded by the European Commission in the ESPRIT LTR project *Verification of Hybrid Systems (VHS)*, by the German Research Council (DFG) in the focussed research program *Analysis and Synthesis of Technical Systems with Continuous-Discrete Dynamics (KONDISK)* and the temporary graduate school ("Graduiertenkolleg") *Modelling and Model-Based Design of Complex Technical Systems*, and by the German Academic Exchange Service (DAAD) in the exchange programs *British-German Academic Research Collaboration (ARC)* with the British Council and *Project-related Exchange of Personnel* with the NSF.

References

Alur, R. and D. Dill, "A theory of timed automata," *Theoretical Computer Science*, **126**, 183–235 (1990).

Alur, R., 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).

Alur, R., T. A. Henzinger, G. Lafferiere, and G. J. Pappas, "Discrete Abstractions of Hybrid Systems," *Proceedings of the IEEE*, **88**(7), 971–984 (2000).

Antsaklis, P. and A. Nerode, editors, *Special Issue on Hybrid Control Systems*, volume 43 of *IEEE Trans. Auto. Cont.* (1998).

Antsaklis, P., editor, *Special Issue on Hybrid Systems: Theory and Applications*, volume 88, no. 7 of *Proceedings of the IEEE* (2000).

Asarin, E. and O. Maler, As soon as possible: time optimal control for timed automata, In Vaandrager, F. W. and J. 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 19–30. Springer (1999).

Bemporad, A. and M. Morari, Verification of hybrid systems using mathematical programming, In Vaandrager, F. W. and J. 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).

Brandin, B. A. and W. M. Wonham, "Supervisory control of timed discrete event systems," *IEEE Trans. Auto. Cont.*, **39**, 329–342 (1994).

Buechi, J. R. and L. H. Landweber, "Solving sequential conditions by finite state operators," *Trans. AMS*, **138**, 295–311 (1969).

Burch, J. R., E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang, "Symbolic model checking: 10^{20} states and beyond," *Inform. and Comput.*, **98**(2), 142–170 (1992).

Chouika, M., B. Ober, and E. Schnieder, Model-based control synthesis for discrete event systems, In *Proc. IAESTED Int. Conf. on Modeling and Simulation, Pittsburgh, USA, 1998*, pages 276–280 (1998).

Chung, S. L., S. Lafortune, and F. Lin, "Limited lookahead policies in supervisory control of discrete event systems," *IEEE Trans. Auto. Cont.*, **37**(12), 1921–1935 (1992).

Chutinan, A. 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).

Clarke, E. M. and R. P. Kurshan, "Computer-aided verification," *IEEE Spectrum*, pages 61–67 (1996).

Dang, T. and O. Maler, Reachability Analysis via Face Lifting, In Henzinger, T. A. 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).

David, R. and H. Alla, *Petri nets and Grafcet*. Prentice Hall, New York (1992).

Dimitriadis, V. D., N. Shah, and C. C. Pantelides, "A case study in hybrid process safety verification," *Comput. Chem. Eng.*, **20, Suppl.**, S503–S508 (1996).

Dimitriadis, V. D., N. Shah, and C. C. Pantelides, "Modelling and safety verification of dicrete/continuous processing systems," *AIChE J.*, **43**(4), 1041–1059 (1997).

Engell, S., editor, *Special Issue on Discrete Event Models of Continuous Systems*, volume 6, no. 1 of *Mathematical and Computer Modelling of Dynamical Systems* (2000).

Fehnker, A., Scheduling a steel plant with timed automata (1999). Technical Report CSI-R9910, Computing Science Institute Nijmegen.

Frey, G. and L. Litz, Verification and validation of control algorithms by coupling of interpreted Petri nets, In *Proc. IEEE SMC, October 1998, San Diego, USA*, pages 7–12 (1999).

Fritz, M., K. Preuß, and S. Engell, A framework for flexible simulation of batch plants, In Zaytoon, J., editor, *Proc. 3rd Int. Conf. ADPM'98, Reims, France, March 1998*, pages 263–270 (1998).

Golaszewski, C. H. and P. J. Ramadge, Control of discrete event processes with forced events, In *Proc. 26th Conf. Decision and Control*, pages 247–251 (1987).

Graf, H. and H. Schmidt-Traub, A model-based approach to process hazard identification, In *Proc. 13th Int. Congress of Chemical and Process Engineering, CHISA'98, Prague, 1998* (1998).

Greenstreet, M. and I. Mitchell, Reachability Analysis Using Polygonal Projections, In Vaandrager, F. W. and J. 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).

Hanisch, H.-M., A. Lüder, and M. Rausch, "Controller synthesis for net condition/event systems with a solution to incomplete state observation," *Euro. J. Cont.*, **3**, 280–291 (1997).

Hanisch, H.-M., "Coordination control modeling in batch production systems by means of Petri nets," *Comput. Chem. Eng.*, **16**(1), 1–10 (1992).

Harel, D. and A. Naamad, "The STATEMATE Semantics of Statecharts," *ACM Transactions on Software Engineering and Technology*, **4**(5), 293–333 (1996).

Heiner, M., P. Deussen, and J. Spranger, "A case study in design and verification of manufacturing system control software with hierarchical Petri nets," *Advanced Manufacturing Technology*, **15**, 139–152 (1999).

Henzinger, T. A., 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).

Herrmann, P., G. Graw, and H. Krumm, Compositional specification and structured verification of hybrid systems in cTLA, In *Proc. 1st IEEE int. Symposium on Object-Oriented Real-Time Distributed Computing, Kyoto, Japan* (1998).

Holloway, L. E., B. H. Krogh, and A. Giua, "A survey of petri net methods for controlled discrete event systems," *J. Disc. Event Dyn. Sys.*, **7**, 151–190 (1997).

IEC, International Standard 1131: Programmable logic controllers. Part 3: Languages, International Electrotechnical Commission, Geneva (1992).

ISA, International Standard S88.01: Batch Control, Part 1: Models and Terminology, Instrumentation Society of America (1995).

Kowalewski, S., H.-M. Hanisch, and U. Anderssohn, Logic controller synthesis for non c/u-partitionable automata with forbidden states, In *Preprints IFAC 13th World Congress, San Francisco, 1996, vol. J*, pages 359–364 (1996).

Kowalewski, S., S. Engell, J. Preußig, and O. Stursberg, "Verification of logic controllers for continuous plants using timed condition/event system models," *Automatica*, **35**(3), 505–518 (1999).

Kowalewski, S., O. Stursberg, and N. Bauer, "An Experimental Batch Plant as a Case Example for the Verification of Hybrid Systems," *Euro. J. Cont.* (2001). To appear.

Krogh, B. H. and S. Kowalewski, "State feedback control of condition/event systems," *Mathematical and Computer Modeling*, **23**(11/12), 161–174 (1996).

Larsen, K. G., P. Pettersson, and W. Yi, "UPPAAL in a nutshell," *Software Tools for Technology Transfer*, **1**(1,2), 134–152 (1997).

Lawley, H. G., "Operability studies and hazard analysis," *Chem. Eng. Prog.*, **70**, 105–116 (1974).

Lemmon, M., K. He, and I. Markovsky, "Supervisory hybrid systems," *IEEE Cont. Sys. Mag.*, **19**, 42–55 (1999).

Li, Y. and W. M. Wonham, "Control of vector discrete event systems: I. the base model," *IEEE Trans. Auto. Cont.*, **38**, 1214–1227 (1993).

Li, Y. and W. M. Wonham, "Control of vector discrete event systems: II. controller synthesis," *IEEE Trans. Auto. Cont.*, **39**, 512–531 (1994).

Lind-Nielsen, J., H. R. Andersen, G. Behrmann, H. Hulgaard, K. Kristoffersen, and K. G. Larsen, Verification of large state/event systems using compositionality and dependency analysis, In *Proc. TACAS'98*, Lecture Notes in Computer Science 1384, pages 201–216. Springer (1998).

Mader, A. and H. Wupper, Timed automaton models for simple programmable logic controllers, In *Proc. Euromicro Conf. on Real-Time Systems, York, UK, June 1999* (1999).

Maler, O., On the programming of industrial computers (1999). Report of the ESPRIT project VHS, see (VHS, 2000).

Marikar, M. T., G. E. Rotstein, A. Sanchez, and S. Macchietto, Computer aided analysis and synthesis of procedural controllers, In *Proc. Workshop on Discrete Event Systems (WODES'98), Cagliari, Italy, 1998*, pages 420–425. IEE (1998).

Moon, I., G. J. Powers, J. R. Burch, and E. M. Clarke, "Automatic verification of sequential control systems using temporal logic," *AIChE J.*, **38**(1), 67–75 (1992).

Morse, A. S., C. C. Pantelides, S. S. Sastry, and J. M. Schumacher, editors, *Special Issue on Hybrid Systems*, volume 35, issue 3 of *Automatica* (1999).

Niebert, P. and S. Yovine, Computing optimal operation schedules for multi batch operation of chemical plants (1999). Report of the ESPRIT project VHS, see (VHS, 2000).

Park, T. and P. I. Barton, "Implicit model checking of logic based control systems," *AIChE J.*, **43**(9), 2246–2260 (1997).

Preußig, J., O. Stursberg, and S. Kowalewski, Reachability Analysis of a Class of Switched Continuous Systems by Integrating Rectangular Approximation and Rectangular Analysis, In Vaandrager, F. W. and J. 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).

Preußig, J., *Formale Überprüfung der Korrektheit von Steuerungen mittels rektangulärer Automaten*, PhD thesis, Department of Chemical Engineering, University of Dortmund, Germany (2000). (in German).

Probst, S. T., G. J. Powers, D. E. Long, and I. Moon, "Verification of a logically controlled solids transport system using symbolic model checking," *Comput. Chem. Eng.*, **21**(4), 417–429 (1997).

Raisch, J. and S. O'Young, "Discrete approximation and supervisory control of continuous systems," *IEEE Trans. Auto. Cont.*, **43**(4), 569–573 (1998).

Ramadge, P. J. and W. M. Wonham, "Modular feedback logic for discrete event systems," *SIAM J. Cont. Optim.*, **25**, 1202–1218 (1987a).

Ramadge, P. J. and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Cont. Optim.*, **25**, 206–230 (1987b).

Ramadge, P. J. and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE*, **77**, 81–98 (1989).

Sampaath, M., R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Failure diagnosis using discrete event models," *IEEE Trans. Cont. Sys. Tech.*, **4**(2), 105–124 (1996).

Sanchez, A., *Formal specification and synthesis of procedural controllers for process systems*, Lecture Notes in Control and Information Sciences 212. Springer (1996).

Tittus, M., *Control synthesis for batch processes*, PhD thesis, Chalmers University of Technology, Göteborg, Sweden (1995).

Vaidhyanathan, R. and V. Venkatasubramanian, "Digraph-based models for automated HAZOP analysis," *Reliability Engineering and Systems Safety*, **50**, 33–49 (1995).

VHS, ESPRIT project Verification of Hybrid Systems (2000). `http://www-verimag.imag.fr/VHS/main.html`.

Wong-Toi, H., The synthesis of controllers for linear hybrid automata, In *Proc. Conf. Decision and Control*. IEEE (1997).

Yang, S. and P. W. H. Chung, "Hazard analysis and support tool for computer-controlled processes," *J. Loss Prevention in the Process Industries*, **11**, 333–345 (1998).

Yovine, S., "KRONOS: a verification tool for real-time systems," *Software Tools for Technology Transfer*, **1**(1,2), 123–133 (1997).