Delft University of Technology

Fac. of Information Technology and Systems **Control Systems Engineering**

Technical report bds:99-04

Approaches to modelling, analysis, and control of hybrid systems*

R.K. Boel, B. De Schutter, G. Nijsse, J.M. Schumacher, and J.H. van Schuppen

If you want to cite this report, please use the following reference instead:
R.K. Boel, B. De Schutter, G. Nijsse, J.M. Schumacher, and J.H. van Schuppen, "Approaches to modelling, analysis, and control of hybrid systems," *Journal A*, vol. 40, no. 4, pp. 16–27, Dec. 1999.

Control Systems Engineering
Faculty of Information Technology and Systems
Delft University of Technology
Delft, The Netherlands
phone: +31-15-278.51.19 (secretary)
fax: +31-15-278.66.79
Current URL: `http://www.dcsc.tudelft.nl`

*This report can also be downloaded via `http://pub.deschutter.info/abs/99_04.html`

Approaches to Modelling, Analysis, and Control of Hybrid Systems

R.K. Boel*, B. De Schutter†, G. Nijsse‡, J.M. Schumacher§, J.H. van Schuppen¶

Abstract

Both industry and the academic world are becoming more and more interested in techniques to model, analyse, and control complex systems that contain both analog and logical components. Such systems are called hybrid systems. Examples of hybrid systems are telecommunication networks, flexible production systems, parallel processing systems, traffic networks, and so on. In this paper we present an overview of some modelling, analysis, and control methods for hybrid systems, mainly from a systems and control point of view. All these methods have been studied intensively by the authors of this paper in the framework of their research projects.

1 Introduction

1.1 Hybrid systems

Hybrid systems arise throughout business and industry in areas such as interactive distributed simulation, traffic control, plant process control, aircraft and robot design, and path planning. There are several possible definitions of hybrid systems. For some authors a hybrid system is a coupling of a continuous-time or analog system and a discrete-time or digital system (in practice often a continuous-time, analog plant and an asynchronous, digital controller). We shall use a somewhat different definition. For us, hybrid systems arise from the interaction between continuous-variable systems (i.e. systems that can be described by a system of difference or differential equations) and discrete-event systems (i.e. asynchronous systems where the state transitions are initiated by events). In general we could say that a hybrid system can be in one of several modes of operation, whereby in each mode the behaviour of

*Vakgroep Elektrische Energietechniek, University of Ghent, Technologiepark-Zwijnaarde, B-9052 Ghent, Belgium, tel: +32-9-264.56.58, fax: +32-9-264.58.40, email: Rene.Boel@rug.ac.be

†Control Laboratory, Faculty of Information Technology and Systems, Delft University of Technology, P.O. Box 5031, 2600 GA Delft, The Netherlands, tel: +31-15-278.51.17, fax: +31-15-278.66.79, email: b.deschutter@its.tudelft.nl

‡Measurement and Systems Technology Chair, Faculty of Applied Physics, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands, tel: +31-53-489.10.89, fax: +31-53-489.31.84, email: G.Nijsse@TN.UTwente.NL

§CWI (Centre for Mathematics and Computer Science), P.O. Box 94079, 1090 GB Amsterdam, The Netherlands, tel: +31-20-592.40.90, fax: +31-20-592.41.99, email: jms@cwi.nl

¶CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands, tel: +31-20-592.40.85, fax: +31-20-592.41.99, email: J.H.van.Schuppen@cwi.nl

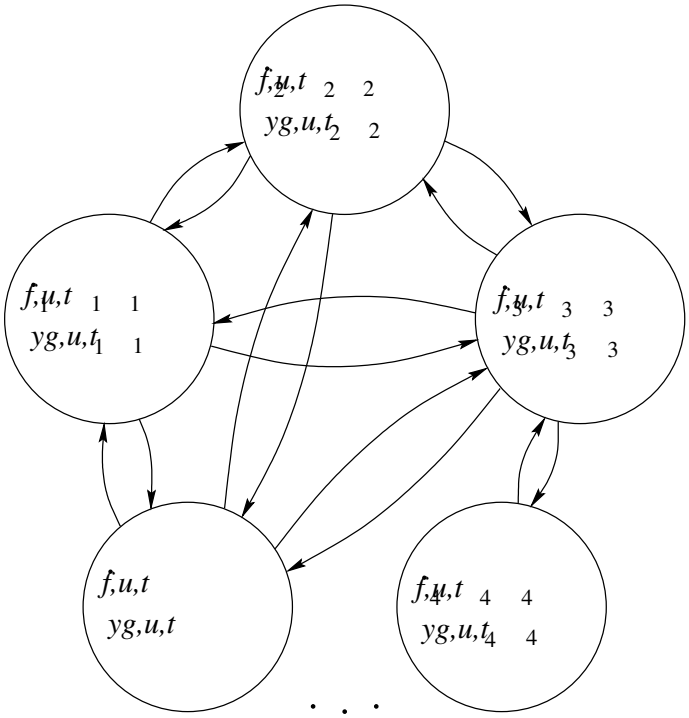

Figure 1: A schematic representation of a hybrid system with N modes. In each mode the behaviour of the hybrid system is described by a system of differential (or difference) equations. The system goes from one mode to another due to the occurrence of an event (this is indicated by the arrows). The vector x_i represents the state of the system in mode i, the vector u represents the input of the hybrid system, and the vector y the output.

the system can be described by a system of difference or differential equations, and that the system switches from one mode to another due to the occurrence of events (see Figure 1). The mode transitions may be caused by an external control signal, by an internal control signal (if the controller is already included in the system under consideration), or by the dynamics of the system itself, i.e. when a certain boundary in the state space is crossed. The latter is expressed by "guards": as long as the guard conditions of a given mode are not crossed the system stays in that mode. At a switching time instant there may be a reset of the state (i.e. a jump in the values of the state variables) and/or the dimension of the state may change.

There are many modelling and analysis techniques for hybrid systems. Typical modelling techniques are predicate calculus, real-time temporal logics, timed communicating sequential processes, hybrid automata, timed automata, timed Petri nets, and object-oriented modelling languages such as Modelica, SHIFT or Chi. Current analysis techniques for hybrid systems include formal verification, perturbation analysis, and computer simulation. Furthermore, special mathematical analysis techniques have been developed for specific subclasses of hybrid systems. We shall only discuss some of these methods. For more information on the other methods the interested reader is referred to [5, 6, 32, 34, 56] and the references cited therein.

1.2 Model classes and analysis tools

Hybrid system models typically represent large plants, consisting of many interacting modules. The interaction between modules can be via common discrete variables, or via common continuous variables, or via both discrete and continuous variables. The usual trade-off between increased levels of detail in the model necessary for verifying desirable properties versus the increased computational complexity of the analysis dominates the study of hybrid systems. This paper presents different choices for this trade-off, corresponding to the different research interests of the authors.

Due to the intrinsically complex nature of hybrid systems the system model and the control design would become hopelessly complicated if one described or tried to control the full behaviour of the system. Therefore, it is often necessary to decompose the system or the control tasks into several smaller parts. This leads to hierarchical control, which is discussed in Section 2.

Models of hybrid systems often start from untimed discrete event systems, adding timing information to events. To each allowed transition of the discrete event system a clock is added, and the execution of the transition depends on this clock. These clocks are synchronised with a global clock representing real time, and hence such models allow verification and control of timing properties of the plant, such as throughput, deadlines, etc. Timed discrete event systems have been studied extensively [1, 3]. An example of such a model class is given in Section 3 where timed Petri nets are treated. The well-known Petri net formalism — an untimed, logical representation of the dynamics of a discrete event system — is extended by requiring that a transition be executed within a certain time interval after the transition becomes enabled. Verification of timing properties in this case uses a combination of analysis tools for untimed Petri nets — both graphical and linear algebraic — and of techniques for solving systems of linear inequalities.

A much larger class of hybrid systems can be modelled by considering a discrete event system where the continuous dynamics in each discrete state are modelled by an arbitrary differential equation or differential inclusion. In Section 4 such models are used to formulate a general control synthesis problem for hybrid systems. Various techniques can be used to describe the set of all reachable states. It has been shown [2, 8, 48] that many interesting questions for hybrid systems where the continuous dynamics are more complicated than a linearly progressing clock (e.g. the clocks in the timed automata or timed Petri nets) are NP-hard[1] or even undecidable[2]. These negative results do not imply that there cannot be many interesting heuristic tools for analysing such hybrid systems. Nevertheless, these complexity results do pose a limit to the analysis of general hybrid systems. Much effort has therefore been spent on methods for approximating general hybrid systems by timed automata with linear clocks, with rates constrained to lie within a polyhedron depending on the discrete state. The evolution of the discrete state depends on external signals, on an internal automaton model for the discrete state, or on the continuous state dynamics (e.g. via guards forcing a transition whenever the continuous state reaches a boundary), or on a combination of all of these. By taking a fine quantisation of the continuous state space making the polyhedra as small as desired, it is possible to approximate any complicated hybrid system as closely as desired using timed automata. Many authors have studied minimal requirements for achieving a

[1] A problem is NP-hard if there does not exist a polynomial-time algorithm that solves the problem unless $P = NP$ [23], which is generally assumed not to be the case.

[2] A problem is undecidable if there cannot exist generally applicable algorithms that solve the problem.

good approximation, and for using such models to prove properties of general hybrid systems [4, 15, 16, 25, 45]. This approach will not be described further in this paper.

It is also possible to approach hybrid systems by extending the state of a continuous system, represented by an ordinary differential equation, with some variables belonging to a finite set. These logical variables may represent discrete states (valve open or closed), modes of operation of a plant, and abstractions or quantisations of continuous variables the behaviour of which is too complicated to include in detail. One class of such models is the class of *complementarity systems* where mode changes occur whenever the continuous state hits some boundary (e.g. a mechanical system where an object moves around in a region bounded by "hard" walls). Such systems are studied in detail in Section 5.

Proving properties about systems where discrete variables are added to a continuous model often leads to questions on the existence of solutions of mathematical programming problems where continuous and integer variables are mixed. One example is the work of Morari and Bemporad [7] where classical linear system models are extended with logical variables, and where mixed integer linear programming is used to prove that the state always remains in a set of safe states. In Section 6 of this paper we consider in detail the special case of such models, where in each mode the continuous variables evolve along straight lines until a given upper or lower saturation level is reached. In that case the verification of properties and the optimisation of the plant behaviour can be written as a minimisation problem over the solution set of an extended linear complementarity problem, and in some cases also as a linear programming problem, which can be solved very efficiently.

The most widely studied technique for hybrid systems is computer simulation, via a combination of discrete event simulation and dae[3] solvers. Some computer simulation and verification tools that are (also) used for hybrid systems are BaSiP, Dymola, OmSim, HyTech, KRONOS and UPPAAL. Simulation models can represent the plant with a high degree of detail, providing a close correspondence between simulated behaviour and real plant behaviour. This approach is, for any large system, computationally very demanding, and moreover it is difficult to understand from a simulation how the behaviour depends on model parameters. This difficulty is even more pronounced in the case of large hybrid systems which consist of many interacting modules. Fast simulation techniques based on variance reduction, and perturbation analysis techniques [11] have been developed in order to partially overcome these limitations. These simulation techniques will not be further discussed in this paper. In this paper we only present those analytical techniques for proving properties of hybrid systems, and for designing control laws for hybrid systems, with which the authors are most familiar.

2 Hierarchical control of hybrid systems

In this section we introduce the hierarchical control paradigm. Since this paradigm does not apply exclusively to hybrid systems but can be used for any kind of large complex system, we shall give a rather general description of this concept. First we present three fundamental control paradigms: centralised control, decentralised control, and hierarchical control. We discuss the differences between these methods, and their advantages and disadvantages. We conclude this section with two applications of hierarchical control for hybrid systems: platooning in automatic highway systems and air traffic management.

[3] dae: differential algebraic equation.

2.1 Centralised, decentralised and hierarchical control

The most common way of controlling "small-scale" systems employs a centralised (feedback) control paradigm. The output of the system is collected and processed centrally, and the input of the plant is driven to track the desired output. In this paradigm the control settings of all the actuators in the system are computed on the basis of the same information. The main advantage of centralised control is that one can try to globally optimise the output of the system, because the central system makes decisions based upon all the information about the system that is currently available. However, certain classes of systems, such as complex large-scale systems that are characterised by a large number of inputs, outputs, and states, can hardly be controlled centrally. An example of such a system is an air traffic management system (see [49] and Section 2.2): the growing number of planes in the sky requires new controlling techniques, since it is getting more and more difficult and inefficient to control all the traffic over an entire region or country in a central way, i.e. from the control tower. There are several reasons why this kind of systems can not be controlled in a central way, e.g. [38, 44]:

1. The amount of information that needs to be communicated is far too large. These communication requirements also make the centralised system very unreliable, and very sensitive to failures in the communication links.

2. The *on-line* implementation of a centralised controller may require a huge amount of computational power and computation speeds that exceed the capacity of currently available DSPs, especially if a non-static or adaptive controller is required in order to account for the changes in the system or its environment.

3. There is no scalability. Since every actuator and every sensor must be connected to the central system, a centralised control system is not very extendible.

A solution to overcome these difficulties is to use a modular approach. One possible modular approach is decentralised control [44, 50]. In decentralised control, a process is split into several modules or subprocesses where every subprocess has a local controller (see Figure 2). To achieve global control objectives, the subprocesses must exchange information amongst themselves. Decentralised control allows us to overcome the computational problems that are associated with on-line control of large-scale systems. However, the interactions between the modules should also be taken into account: even if each local controller is an optimal controller for its own subprocess, this does not imply that the overall behaviour of the global system is optimal. Some of the local controllers may even counteract each other, which might lead to instability.

A compromise between centralised and decentralised control is hierarchical control. It combines the advantages of centralised control (i.e. global system performance) and decentralised control (i.e. tractability). In a hierarchical control paradigm the process is also split into subprocesses with every subprocess having a local controller. The local controllers make up the lowest level in a hierarchical tree; the higher levels in the tree deal with more global aspects of the performance (see Figure 3).

2.2 Examples of hierarchical control for hybrid systems

Typical applications of hierarchical control for hybrid systems can be found in automated highway systems [24,37] and in air traffic management systems [55]. In the automated highway

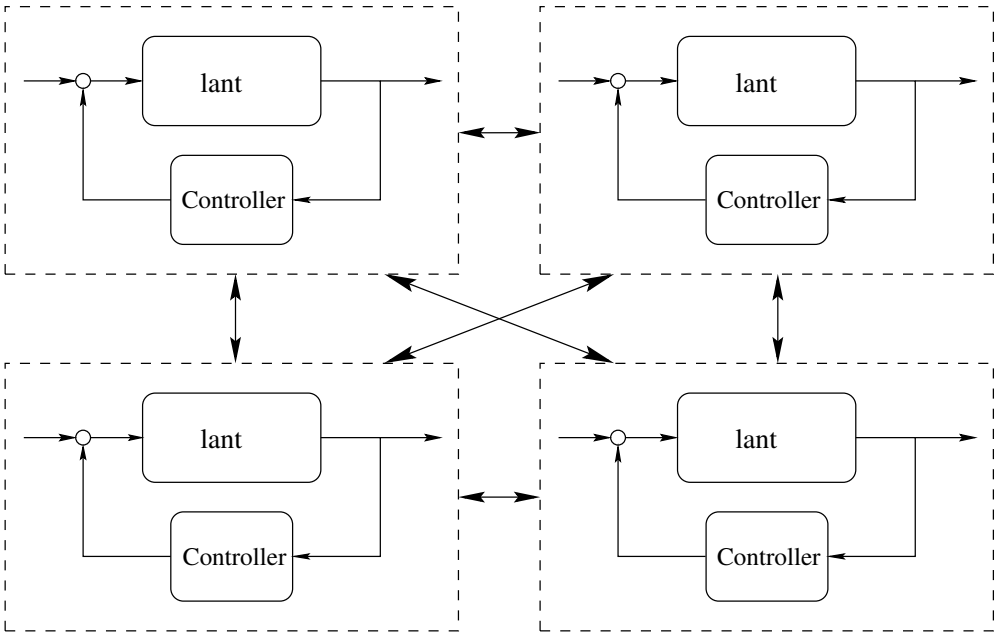

Figure 2: The decentralised control paradigm. The process under consideration is split into subprocesses (indicated by dashed rectangles). Each subprocess has a local controller, and the controlled subprocesses exchange information amongst themselves (which is indicated by the thick arrows between the different subprocesses).

systems described by Godbole *et al.* [24,37] five layers can be distinguished (see Figure 3 with $N = 5$). The first layer (top layer) in the design is the *network* layer which takes care of the flow of traffic in the entire highway system; one of its tasks is e.g. to prevent congestion. The second layer in the design, the *link* layer, coordinates the operation of specific sections of the highway, such that e.g. the throughput on a specific section is maximised, while maintaining safety. Typical functions of this layer are calculating an optimal platoon size and vehicle speed, deciding which lanes the car should take to get as quickly as possible to their destination and to take decision what to do when incidents happen. The task of the third layer in the design, the *coordination* layer, is to coordinate the operation of platoons[4] that are in each other's neighbourhood e.g. to merge platoons which are too small, or to split up large platoons into smaller platoons. Typically the nature of this layer is a discrete event system since it carries out commands like "*accelerate and merge with the preceding platoon*", i.e. it asks specific platoons to carry out a specific manoeuvre. The fourth layer in the design, the *regulation* layer, translates the commands given by the coordination layer into controller set-points for the individual cars such as steering angle, throttle opening, etc. Finally, the fifth layer in the design (bottom layer), the *physical* layer, consists of the vehicles with their sensors, actuators and communication equipment; the vehicles try to follow the setpoints provided by the regulation layer.

Another typical example of a complex hybrid control system is an air traffic management system as described by Tomlin *et al.* [55]. Nowadays commercial aircraft usually have to follow certain predefined corridors in the airspace. As a consequence, a large part of the airspace is

[4]A platoon is a number of cars grouped together.

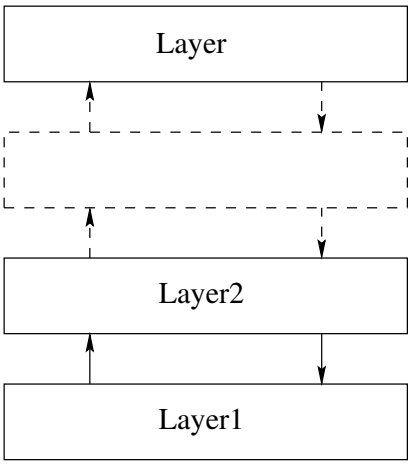

Figure 3: The hierarchical control scheme in a number of layers. Every layer in the control design has its own specific task and there is communication between the layers in order to coordinate the behaviour of the lower level controllers so as to obtain a better global performance.

not utilised very well (especially when the planes are far away from an airport). Furthermore, the workload of the air traffic controllers increases very rapidly due to the growing number of planes. Therefore, the aviation community is advocating the concept of *free flight*. In the free-flight approach pilots can determine their own routes, altitudes and speeds. However, this can only be done in non-congested areas far away from any airport. If the pilot is in a congested area, say close to the airport, the pilot's preferences will be restricted. In the approach described by Tomlin *et al.* the aircraft communicate information amongst themselves for preventing dangerous situations such as collisions. Coordination among the aircraft is in the form of manoeuvres. Every manoeuvre is a finite sequence of flight modes such as cruise mode, altitude change, and so on. The modelling of these manoeuvres is typically done by a finite state automaton, which interacts with a set of control systems. This leads to a so called multi-agent hybrid control system [55]. Just as in the automated highway systems the overall control system consist of several layers (see Figure 3). The top layer of the architecture is the *strategic planner*, the task of which is to determine a coarse trajectory. The second layer is the *tactical planner*, the task of which is to determine the course more accurately using kinematic models. The task of the third layer, the *trajectory planner* layer, is to determine the trajectory even more accurately using a detailed kinematic model. The fourth layer consists of the *regulation layer* and tracks the trajectory. Finally, the fifth layer consists of the aircraft.

3 Timed Petri net models

This section proposes timed Petri nets as a tool for the mathematical description of the behaviour of large plants. The graphical representation of a classical Petri net [43, 46] efficiently represents the ordering of events in a plant (an event corresponds to the execution of a transition). The interaction between several modules of the plant is represented by ensuring that transitions which are common to several Petri net components can only be executed when their execution is allowed in each of the Petri net components. In this section we extend the

Petri net model by including in the description the real time at which events take place. A model of a transportation system illustrates how timed Petri nets can represent fairly large plants. We also indicate how classical analysis techniques for Petri nets can be extended to study properties of timed Petri nets.

3.1 Semantics of timed Petri net models

Consider as an example an integrated multimodal transport system where different goods, following different routes, are moved from origin to destination, using different types of vehicles; transshipment stations allow transfer of goods from one type of vehicle to another type. One set of modules represents the different routes along which goods can be transported, using a timed Petri net component for each route. These components all look very similar, only the names of places and transitions, and perhaps the length of the paths, differing (compare this to classes in object oriented programming). Another class of modules represents the location and availability of all vehicles of a particular type. The overall model contains as many components of this class as there are different types of vehicles. Finally a third class of components represents the state of the resources in the different transshipment stations. Each of these modules will be represented by a timed Petri net. Figure 4 provides a very simple instantiation for each of the three classes of Petri net components necessary to represent a multimodal transportation system.

An untimed Petri net consists of a finite set of places P, a finite set of transitions T, and a set of directed arcs connecting some places to some transitions, and connecting some transitions to some places. Graphically places are represented by circles, transitions by bars, and directed arcs by lines with an arrow. The state of the module represented by the Petri net, is described by the distribution of tokens over the places of the Petri net. In other words the number of tokens $m(p) \geq 0$ in each place $p \in P$ (i.e. the number of dots in the circle representing place p) specifies the possible future behaviour of the module. Transition t is called state-enabled when each upstream place of t contains at least one token. The set of upstream places of t is denoted by ${}^\bullet t$, while output places of t are denoted by $t^\bullet$. The event modelled by transition t can be executed only if t is state-enabled. Executing transition t at time θ removes a token from each upstream place of t and at the same time adds one token to each downstream place of t.

An untimed Petri net model specifies the order in which events can occur during the operation of the plant. In applications one is not only interested in the order in which events happen, but also in the time instants at which these events can occur. One might for example want to determine the range of feasible throughputs of the transportation system, or the maximal time delay for perishable items. In order to represent these time constraints the model has to describe exactly when transitions can actually fire, after they have become state-enabled. This requires an extension of the Petri net model.

The state of the net at time θ for a timed Petri net not only remembers the number of tokens in each place, but also their arrival times. This implies that the state M_θ of the timed, marked Petri net at time θ lists, for each place p, the set $M_\theta(p) := \{\theta_1, \ldots, \theta_{m_\theta(p)}\}$ of arrival times $\theta_1 \leq \theta_2 \leq \ldots \leq \theta_{m_\theta(p)}$ $(\leq \theta)$ of the $m_\theta(p)$ tokens in place p.

With each transition t an interval $[L(t), U(t)]$ is associated (with $0 \leq L(t) \leq U(t) \leq \infty$). Transition t becomes enabled at the time $\max_{p \in {}^\bullet t} \min M_\theta(p)$. Then the transition t must fire at some time

$$\theta \in [\max_{p \in {}^\bullet t} \min M_\theta(p) + L(t), \max_{p \in {}^\bullet t} \min M_\theta(p) + U(t)] \tag{1}$$

provided the condition enabling t is maintained during the whole interval. In particular if the enabling condition is still valid at the final time $\max_{p\in{}^\bullet t} \min M_\theta(p) + U(t)$ of the firing interval, then the transition is forced to fire at this exact time.

Given the state M_0 of the net at initial time 0, one can describe the evolution of the timed Petri net as follows. One of the transitions t_1, which is enabled at time 0 will fire at some time $\theta \geq 0$ within the firing interval $[\max_{p\in{}^\bullet t_1} \min M_\theta(p) + L(t_1), \max_{p\in{}^\bullet t_1} \min M_\theta(p) + U(t_1)]$ of t_1 as specified above. The firing of transition t_1 changes the state of the net as follows: from each place $p \in^\bullet t_1$ the token $\min M_\theta(p)$ is removed; to each place in $t_1^\bullet$ a token with value the firing time θ is added.

Let us now return to the example of multimodal transport. The Petri net model of Figure 4(a) represents the route taken by a load of type $\alpha \in A$ (each element of A represents one instantiation of the class of Petri net components representing a route), while Figure 4(b) is one instantiation of a Petri net model representing availability and location of a vehicle of type x, and Figure 4(c) is an instantiation of a Petri net component representing a transshipment station where an item of type $\alpha \in A$ can be unloaded by a vehicle of type $a \in V$ and later picked up by a vehicle of type $b \in V$. Note that if K_x vehicles of type $x \in V$ are present in the system, then the Petri net for the xth instantiation of Petri net 4(b) contains K_x tokens. It is also possible that more than one item of type $\alpha \in A$ is present in the system at the same time. In a typical system, there will be constraints on how many items can be present at the same time. For example, the storage capacity of the transshipment station will typically be bounded, leading to a constraint on the set of allowable markings of the form $\sum_{\xi\in A} m(p_{2,\xi}) \leq B$.

A particular item to be transported is represented by a token in one place of the Petri net model of the route of that particular type of goods. This place represents the present location or stage reached by the item along its route. The item, represented by the token, can only move to the next place in the net if a token is also available in the corresponding place(s) of the Petri net model of the vehicle to be used, and possibly in the transshipment station model, and if all these enabling tokens arrived at least $L(t)$ time units ago. This synchronisation assumption models the interaction between different components in the plant model.

Consider now in detail the evolution over time of the multimodal transport example. A new item which will follow route α has arrived at time $\theta_{0,\alpha}$ (the token in place $p_{0,\alpha}$) while another item — which arrived earlier — following the same route α has already reached the input buffer $p_{1,\alpha}$ of the transshipment station at some other time $\theta_{1,\alpha}$. Moreover, a vehicle of type a has been available since $\theta_{a,0}$, as indicated by the token in $p_{a,0}$ while the transport system in the transshipment station is available since θ_2. All these arrival times are prior to the present time, but such that no transition has yet been forced to occur. Hence, the following transitions can be executed:

- either $t_{2,a,\alpha}$ — corresponding to the event "*move item of type α from arrival position to transshipment station*" — will be executed in the interval $[\max(\theta_{0,\alpha}, \theta_{a,0}) + L(t_{2,a,\alpha}), \min(\max(\theta_{0,\alpha}, \theta_{a,0}) + U(t_{2,a,\alpha}), \max(\theta_{1,\alpha}, \theta_2) + U(t_{3,\alpha}))]$,

- or $t_{3,\alpha}$ — corresponding to the event "*move item from input buffer of transshipment station to output buffer*" — will be executed in the interval $[\max(\theta_{1,\alpha}, \theta_2) + L(t_{3,\alpha}), \min(\max(\theta_{0,\alpha}, \theta_{a,0}) + U(t_{2,a,\alpha}), \max(\theta_{1,\alpha}, \theta_2) + U(t_{3,\alpha}))]$.

If $t_{2,a,\alpha}$ is executed first, then the next state will be such that $p_{1,\alpha}$ contains 2 tokens (the new token having the value θ at which the event occurred), $p_{0,\alpha}$ and $p_{a,0}$ have no tokens left, and

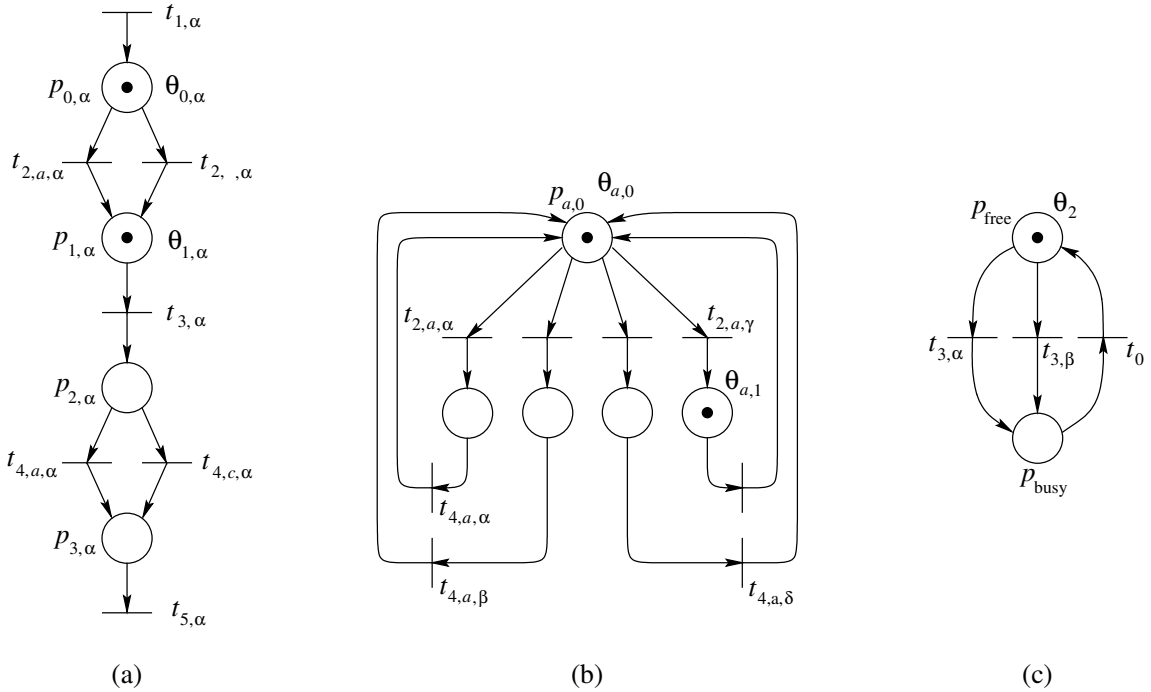

Figure 4: Petri net model of a multimodal transport system.

$p_{1,a,\alpha}$ also has one token. If on the other hand $t_{3,\alpha}$ is executed first, then the next state has tokens in places $p_{0,\alpha}$, $p_{2,\alpha}$, $p_{1,a,\alpha}$, and p_{busy} with the token in $p_{2,\alpha}$ having the value θ at which the transition occurred. Both sequences of execution are allowed, unless the upper bound $\min(\max(\theta_{0,\alpha}, \theta_{a,0}) + U(t_{2,a,\alpha}), \max(\theta_{1,\alpha}, \theta_2) + U(t_{3,\alpha}))$ is less than one of the lower bounds. External control actions may dictate the choice between these two options.

The reader can continue the enumeration of all possible sequences of future events — and the sequence of feasible trajectories — by repeatedly using the firing rules of the timed Petri net model. These rules uniquely define the set of reachable event sequences. In principle it is therefore possible to verify whether all reachable states satisfy specifications, such as maximal time delays for delivering a perishable item, or maximum number of items stored in buffers, safety constraints avoiding that several vehicles risk collision by being present simultaneously in the same location, etc. The difficulty in proving such properties is the combinatorial explosion of the size of the set of allowable event sequences. The next subsection discusses some techniques for avoiding this combinatorial explosion by using the concurrent nature of Petri nets.

3.2 Analysis and control techniques for timed Petri nets

For untimed Petri nets there is a vast literature with tools for verifying whether the Petri net model satisfies certain properties [22, 40, 41]. Many of these techniques can be extended to timed Petri nets.

Often an efficient analysis method separately analyses the properties of one component at a time. It is e.g. obvious that the number of tokens in a cyclic component like Figure 4(b) is constant at all times, independently of what happens in other components. Inequalities relating the number of times that different transitions in the same component have been

executed can be written down by considering only the behaviour of that component. In a component without choice places tokens cannot overtake each other, and one can easily write down systems of linear inequalities relating the time at which each transition is executed for the nth time.

Suppose now that one tries to derive a maximum delay for items following route α in the multimodal transport network. Clearly this can be written as

$$U(t_{1,\alpha}) + \max(\Delta(t_{2,a,\alpha}), \Delta(t_{2,b,\alpha})) + \Delta(t_{3,\alpha}) + \max(\Delta(t_{4,a,\alpha}), \Delta(t_{4,b,\alpha})) + U(t_{5,\alpha}) \quad (2)$$

where $\Delta(t_{n,x,\alpha})$ represents the maximum delay of execution transition $t_{n,x,\alpha}$. This maximum delay is determined by the behaviour of the instantiation x of the transport component in Figure 4(b). No upper bound can be given since the model of Figure 4(b) does not specify any priorities for allocating tokens in $p_{a,0}$ to transitions. In the model of Figure 4(b) the token might never be allocated to transition $t_{2,a,\alpha}$, and hence $\Delta(t_{2,a,\alpha})$ could be ∞. Hence, extra control decisions have to be added to the model in order to guarantee a certain upper bound on the transport delay.

In general we want to determine whether or not all the trajectories which are allowed by the timed Petri model satisfy the specifications of the plant. If this is not the case, then control decisions have to disable certain transitions so as to eliminate the undesirable trajectories. The system can be controlled by modifying the firing interval of certain controllable transitions, or by disabling or prioritising certain transitions provided they are state-enabled. This event scheduling in order to achieve proper behaviour of the plant is called supervisory control.

A formal definition of a controllable timed Petri net is as follows. Assume that for transitions $t \in T_c \subset T$ (T_c is called the set of controllable transitions) the lower bound $L(t)$ can be increased within certain limitations (at most up to $U(t)$), and that the upper bound $U(t)$ can be decreased within a certain range. A feedback controller specifies for each $t \in T_c$ state feedback functions $L(M_\theta, t) \in [L(t), U(t)]$, and $U(M_\theta, t) \in [L(t), U(t)]$ such that the firing delay interval $[L(t), U(t)]$ is replaced by $[L(M_\theta, t), U(M_\theta, t)]$ in the behavioural specification (1) of the controlled system. This control decision depends at time θ on the state M_θ.

The control design algorithm chooses the lowest bounds $L(M_\theta, t)$, so that the set of possible behaviours is as large as possible, but so that all reachable markings satisfy the plant specifications. This leaves freedom to add further constraints, to minimise cost criteria, etc. Such a controller gives an acceptable schedule, providing safe operation of the plant. Optimal scheduling can be accomplished by further reducing the firing interval, in such a way that some reward function (e.g. throughput of the transportation system) is maximised.

Efficient heuristic algorithms for solving the control synthesis problem are based on a decomposition of the problem in simpler subproblems. Consider the simplest possible constraint: some place p should never contain more than one token. In the case of an untimed Petri net the control synthesis algorithm requires only specification of the evolution of the marking of the places in the influencing net I_p of the place p, i.e. the marking of the set of places $p' \in P$ such that a token in p' can uncontrollably reach p. This influencing net I_p is easy to construct as it is the union of all paths, ending in p, such that all transitions in these paths are uncontrollable. It has been shown in [9, 14, 33] that there are efficient algorithms (albeit worst case exponential in the number of choice places in I_p — but this number is usually quite small compared to the size of the net) to design a control law, provided that the influencing net has certain structural properties.

Control synthesis as described in the above paragraph is easy because the worst case analysis assumes that all transitions can be postponed indefinitely. No transition is ever

forced to be executed in this model. For a controlled timed Petri net, with $U(t) < \infty$ for some t, one has to take into account that control decisions cannot delay tokens by more than $U(t)$. The influencing net in that case includes all places in paths ending in p such that all transitions in the path are either uncontrollable (in $T \setminus T_c$) or are controllable with a finite upper bound. Moreover, the decision depends on the firing times of the output transitions of p, and on the marking of all places which influence these output transitions. This makes the influencing net larger and more complicated. Control decisions change the firing times of transitions both at the boundary of the influencing net I_p (whether or not to allow new tokens to enter the influencing net) and inside the influencing net (delaying some tokens already in I_p). Algorithms for control synthesis are described in detail in [54].

For marked graph components — i.e. components with $\sharp^\bullet p \leq 1, \sharp p^\bullet \leq 1$ for each place p, which implies there is never a choice that is modelled explicitly — the evolution of the marking can be written down as a system of linear inequalities because tokens cannot overtake each other. This simplifies a rigorous proof of the results explained above [54]. As soon as choice is possible one has to consider the union of solution sets of different systems of linear inequalities, corresponding to the different choices which are possible. This leads to a combinatorial explosion in the size of the systems of equations to be solved. Various heuristics for solving these cases are under study.

4 General hybrid systems control

Informally, a *hybrid system* can be considered as a system in which the state set is a finite product of continuous state spaces. The state then consists of the discrete state of the system and, for each discrete state, of a continuous state in the continuous state space.

Control problems for which hybrid systems are appropriate models arise when engineering systems are controlled by computers and where there is a tight interaction between the logical or discrete-event behaviour and the continuous-variable behaviour. In the past, control and system theory for continuous-variable systems and for discrete-event systems have been developed separately. Because of the tight interaction in some engineering control problems these separate approaches are no longer adequate for such problems.

Examples of engineering control problems of hybrid character with which the author of this section is familiar include: control of conveyor belts, control of ethanol production, and control of a fruit juice processing plant. From the literature it appears that the following areas give rise to control problems for hybrid systems: communication networks, road traffic, air traffic control, and chemical process control. A major motivation for control and system theory of hybrid systems is the need for computer programs for control of engineering systems.

Let $\mathbb{R}$ be the set of the real numbers and $\mathbb{R}^+$ be the set of the nonnegative real numbers. Denote the set of the positive integers by $\mathbb{N}_0 = \{1, 2, 3, \ldots\}$.

A *continuous-time hybrid system* is a tuple

$$\left\{ \begin{array}{l} \mathcal{T}, \mathcal{Q}, \Sigma_{\text{in}}, \Sigma_{\text{env}}, \Sigma_{\text{cd}}, \Sigma_{\text{out}}, \mathcal{U}, \mathcal{Y}, \mathbf{U}_{\text{c}}, \mathbf{U}_{\text{ex}}, \\ \{X_q, TX_q, G_q, f_q, h_q, \forall q \in \mathcal{Q}\}, (q_0, x_{q_0,0}), \delta, r \end{array} \right\},$$

where

$\mathcal{T} = \mathbb{R}^+$, said to be the *time index set*,
$\mathcal{Q}$ is a finite set, the *discrete state set*,

Σ_{in} is a finite set, the *set of input events*,
Σ_{env} is a finite set, the set of *environmental events*,
Σ_{cd} is a finite set, the set of *events generated by the continuous dynamics*,
$\Sigma_{\text{out}} \subseteq \Sigma$ the *set of output events*, with $\Sigma = \Sigma_{\text{in}} \cup \Sigma_{\text{env}} \cup \Sigma_{\text{cd}}$,
$\mathcal{U} \subseteq \mathbb{R}^m$, the *continuous input space*,
$\mathcal{Y} \subseteq \mathbb{R}^p$, the *continuous output space*,
$\mathbf{U}_{\text{c}} \subseteq \{u : \mathcal{T} \to \mathcal{U}\}$, the set of *continuous input functions*,
$\mathbf{U}_{\text{ex}} \subseteq (\mathcal{T} \times \Sigma)^* \cup (\mathcal{T} \times \Sigma)^\omega$, the set of external timed-event sequences,
for all $q \in \mathcal{Q}$:
$X_q \subseteq \mathbb{R}^{n_q}$, the *continuous state space at discrete state* $q \in \mathcal{Q}$,
$TX_q(x) \subseteq \mathbb{R}^{n_q}$ the tangent space at $x \in X_q$,
$G_q : \Sigma_{\text{cd}} \to P_{\text{closed}}(X_q)$, the *guard* at $q \in \mathcal{Q}$, a, possibly partial, function,
where $P_{\text{closed}}(X_q)$ denotes the closed subsets of X_q,
$f_q : \mathcal{T} \times X_q \times \mathcal{U} \to TX_q$, the continuous-state evolution map,
$h_q : \mathcal{T} \times X_q \times \mathcal{U} \to \mathcal{Y}$, the output map,
$(q_0, x_{q_0,0}) \in \mathcal{Q} \times X_{q_0}$ the *initial state*,
$\delta : \mathcal{T} \times \mathcal{Q} \times \mathcal{X} \times \Sigma \to \mathcal{Q}$, the discrete *transition function*, a, possibly partial, function,
with $\mathcal{X} = \cup_{q \in \mathcal{Q}} X_q$,
$r : \mathcal{T} \times \mathcal{Q} \times \mathcal{Q} \times \mathcal{X} \times \Sigma \to \mathcal{X}$, the *reset map*, a, possibly partial, function.

The dynamics of the hybrid system is described by the discrete transition function, the reset map, the differential equation, and the output map, according to

$$q^+ = \delta(t, q^-, x^-_{q^-}, \sigma) \tag{3}$$
$$x^+_{q^+} = r(t, q^-, q^+, x^-_{q^-}, \sigma) \tag{4}$$
$$\dot{x}_q(t) = f_q(t, x_q(t), u(t)) \tag{5}$$
$$y(t) = h_q(t, x_q(t), u(t)). \tag{6}$$

The description of the operation of the hybrid system follows. At t_0 the initial state is $(q_0, x_{q_0,0}) \in \mathcal{Q} \times X_{q_0}$. Assume no immediate transition takes place at t_0 (see below for what to do if an event does occur at this time). At the discrete state $q = q_0$ the continuous dynamics proceeds according to the differential equation (5). It is assumed that for all $u \in \mathbf{U}_{\text{c}}$ this differential equation has a unique solution on $\mathbb{R}^+$. The solution will be followed till the next event. The time interval till the next event will be denoted by $[t_0, t_1)$ for $t_1 \in \mathbb{R}^+$ and for subsequent intervals by $[t_n, t_{n+1})$ for $n \in \mathbb{N}_0$.

At any time $t \in \mathcal{T}$ an event may occur that results in a change of the discrete state. The possible events at state $q \in \mathcal{Q}$ and at time $t \in \mathcal{T}$ are:

- an *input event* $\sigma \in \Sigma_{\text{in}}$ occurs if such an event is supplied on the input channel;
- an *environmental event* $\sigma \in \Sigma_{\text{env}}$ occurs if such an event is supplied by the environment;
- an *event generated by the continuous dynamics* $\sigma \in \Sigma_{\text{cd}}$ occurs immediately when $x_q(t-) \in G_q(\sigma)$, thus if the state of the system hits a guard.

If the timed event (t, σ_1) occurs then the transition is described by the discrete transition function and the reset map (3) – (4). The transition should be read as follows: the timed event (t, σ_1) transfers the system from the state $(q^-, x^-_{q^-})$ to the state $(q^+, x^+_{q^+})$. It may be

the case that the new state $(q^+, x^+_{q^+}) \in \mathcal{Q} \times X_{q^+}$ is such that the new state is again a member of a guard, say $x^+_{q^+} \in G_{q^+}(\sigma_2)$. In this case the event $\sigma_2 \in \Sigma_{\rm cd}$ takes place at the same time. It will be assumed that only a finite number of events can occur at any time. After the last event of the sequence of events occurring at moment t, the new state is $(q_{\rm f}, x^+_{q_{\rm f}})$ where $x^+_{q_{\rm f}}$ is the initial condition of the differential equation in the discrete state $q_{\rm f}$. A further extension is to make the guards time-varying.

Some definitions of hybrid systems make a distinction between "enabling" and "enforcing" conditions for the occurrence of events. In this section we only consider the latter kind of events since they can more easily be incorporated in a system-theoretic framework. Note however that enabling conditions are frequently used in informatics.

For a hybrid system to be well defined it must be proven that
(1) at any time only a finite number of events can occur; and
(2) on any finite interval only a finite number of events can occur (non-Zeno behaviour).
Condition (1) can be checked from the definitions by an analysis of the discrete part of the system. Condition (2) requires analysis of switched differential equations. To formulate general sufficient conditions for these assumptions is a rather difficult mathematical problem.

Other definitions of hybrid systems have been proposed. For an overview see the PhD thesis of Branicky [10]. Definitions similar in character to those presented above may be found in [10, Ch. 5] and [26, 39, 51, 53].

A polyhedral set is the intersection of a finite number of closed half spaces of a vector space. A discrete-time time-invariant *linear hybrid system* [51] is a hybrid system with

$$\begin{aligned}
\mathcal{U} &\subseteq \mathbb{R}^m, \mathcal{Y} \subseteq \mathbb{R}^p, \\
X_q &\subseteq \mathbb{R}^{n_q},\ \forall q \in \mathcal{Q},\ \text{are polyhedral sets}, \\
G_q(\sigma) &\subseteq X_q, \forall q \in \mathcal{Q},\ \sigma \in \Sigma_{\rm cd},\ \text{are polyhedral sets}, \\
q^+ &= \delta(q^-, x^-_{q^-}, \sigma), \\
x^+_{q^+} &= A_d(q^-, q^+, \sigma) x^-_{q^-} + B_d(q^-, q^+, \sigma), \\
x_q(t+1) &= A_q x_q(t) + B_q u(t), \\
y(t) &= C_q x_q(t) + D_q u(t).
\end{aligned}$$

The class of linear hybrid systems is a subclass of the class of piecewise-linear systems introduced by Sontag in [51, 52].

The class of hybrid systems is extremely general. Several subclasses have been studied. A subclass is that of the *switched systems* in which all the discrete events are driven by the continuous dynamics (there are neither input events nor environmental events). Problems of this class have been investigated at least since the 1960s (for references see the proceedings [42]). It is also of interest to study subclasses with input events and with environmental events.

5 Complementarity systems

5.1 Motivation

Switches between different "modes" or "regimes" are typical for hybrid systems. Such switches may e.g. be generated by an external discrete controller which acts on an otherwise continuous dynamical system. In other cases, however, regime changes are in some sense intrinsic. For

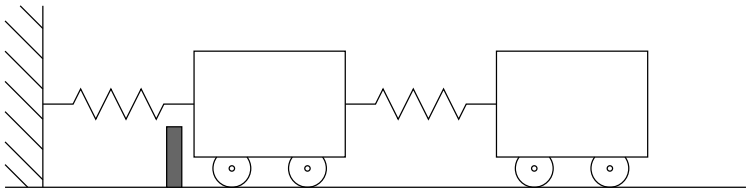

Figure 5: A mechanical example of a complementarity system.

an example of this, consider the mechanical system depicted in Figure 5. The figure shows a one-dimensional mass-spring configuration including a unilateral constraint. Under the assumption that collisions are being modelled as inelastic and instantaneous, the system can be in two modes, namely "constraint active" or "constraint inactive". It is possible to write similar equations for the two different modes as follows (where for sake of simplicity we assume that the springs are linear, we normalise all constants to 1, and we place the constraint at the equilibrium position of the left mass):

$$\ddot{x}_1(t) \;=\; -2x_1(t) + x_2(t) + u(t) \tag{7a}$$

$$\ddot{x}_2(t) \;=\; x_1(t) - x_2(t) \tag{7b}$$

together with

$$u(t) = 0 \tag{7c}$$

for the mode in which the constraint is inactive, or

$$x_1(t) = 0 \tag{7c$'$}$$

for the constrained mode. In the above equations, $x_1(t)$ and $x_2(t)$ denote the displacements of the two masses from their equilibrium positions; the variable $u(t)$ represents the *reaction force* exerted by the stop. The system must switch between the two regimes in order to prevent violation of the following inequality constraints that should be satisfied at all times:

$$x_1(t) \geq 0, \quad u(t) \geq 0. \tag{7d}$$

The mechanical system in the example belongs to the class of so-called *complementarity systems*. In general, complementarity systems are described in terms of continuous state variables which are collectively denoted by the vector $x(t)$, and pairs of *complementary variables* which we denote by $y_i(t)$ and $u_i(t)$. (In the example above, there is only one pair of complementary variables, consisting of the displacement x_1 and the reaction force u.) The state variables and the complementary variables satisfy a system of algebraic and differential equations of the form

$$F(\dot{x}, x, y, u) \;=\; 0 \tag{8a}$$

in which there are more variables than equations (Compare this with the example above, where the equations (7a) and (7b) together with the relation $y = x_1$ constitute a system of three equations in the four variables x_1, x_2, y, and u; the system might be rewritten in first-order form as in (8a), which leads to five equations in six variables.). The system is then closed by adding *complementarity relations* for each pair (y_i, u_i):

$$y_i \geq 0, \quad u_i \geq 0, \quad y_i = 0 \;\text{ or }\; u_i = 0. \tag{8b}$$

The description (8) must be completed by providing *jump rules* which describe possible discontinuous changes of the state at event times (i.e. time instants at which a mode change occurs). It is difficult to give jump rules in general, but well-motivated rules can be given in a number of cases (linear systems, low-index nonlinear systems, mechanical systems). A system governed by (8) with k pairs of complementary variables has in general 2^k different modes, corresponding to the 2^k different ways in which the choices in (8b) can be made.

The term "complementarity relation" is taken from mathematical programming where relations of this type play an important role for instance in the Kuhn-Tucker conditions for optimality. Of course the situation here differs from the one usually considered in mathematical programming since the variables in (8b) depend on time.

From a general hybrid systems point of view, complementarity systems form a rather limited class with very special forms of guards and invariants. However, complementarity conditions do come up naturally in a number of situations. The case of mechanical systems with unilateral constraints has already been mentioned. One may also note that equation (8b) represents, up to a sign, the ideal diode characteristic and so electrical networks with ideal diodes can be written in complementarity form, with voltages across and currents through the diodes as pairs of complementarity variables. Other examples of complementary pairs include pressure / flow (in hydraulic systems with one-way valves), and slack variable / Lagrange multiplier (in optimal control problems with inequality constraints). There are also other applications; e.g. it turns out that one may rewrite piecewise linear relations in complementarity form, so that systems with Coulomb friction or saturation effects can be modelled as complementarity systems as well [13, 35, 47].

As has been noted before in this paper, developing analysis and control techniques for general hybrid systems is an intractable problem, and therefore it is useful to look for subclasses with a special structure that can be used to advantage. Complementarity systems form one example of such a subclass. Complementarity systems of specific types have been studied for a long time in the context of particular applications, such as applied mechanics and electrical network modelling. The idea of coupling complementarity conditions to a general dynamical system with external variables (8a) was first proposed in [57] and has been worked out further in a number of publications including [12, 27–29, 31, 36, 58]; see also [59, Ch. 4].

5.2 Analysis

Perhaps the most basic question that one can ask about a dynamical system is whether it has unique solutions which are defined for all time. This question is not just of theoretical interest. If one writes a simulation model for a hybrid system and it turns out that for some initial conditions the model has multiple solutions, or no solutions, or solutions that cannot be extended beyond a certain point in time, then this may be a reason to change the model. Information about existence and uniqueness of solutions can only partially be obtained from simulation runs since one can always test only a limited number of initial conditions, and moreover standard numerical solvers may not be well-equipped to deal with situations in which there may be multiple or no solutions.

At the level of general hybrid systems, the problem of finding out whether a given model has unique solutions is likely to be intractable. The well-posedness issue can however be fruitfully studied at the level of complementarity systems, although even in this case the results are not yet complete. As an example of what can be done, we present the following algorithm which works for *linear* complementarity systems, i.e. complementarity systems in

which the dynamics in (8a) is linear. For a complete specification of the solution concept in this case (including jump rules) see [28].

Algorithm. Let a system be given in standard linear input/state/output form

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t) + Du(t)$$

together with the complementarity conditions (8b). Denote the number of pairs of complementary variables (i.e. the length of the vectors y and u) by k, and write n for the length of the vector x. Denote the jth unit vector by e_j.

1. Construct a matrix N of size $k \times k$ by letting the jth column of N be equal to the first vector in the sequence $De_j, CBe_j, CABe_j, CA^2Be_j, \ldots$ that is nonzero. If all vectors in this sequence are zero, then we have a singularity and the algorithm terminates[5].

2. Construct a matrix M of size $k \times k$ in the same way as N, but this time working on row vectors rather than column vectors. That is, the jth row of M is equal to the first nonzero row vector in the sequence $e_j^T D, e_j^T CB, e_j^T CAB, \ldots$.

3. For each of the $2^k - 1$ subsets J of the set $\{1, 2, \ldots, k\}$, compute the determinant of the principal submatrix of M indicated by J, i.e. the matrix $(m_{ij})_{i \in J,\, j \in J}$. Do the same for the matrix N.

4. If all of the $2(2^k - 1)$ numbers found under 3. are positive, print the following message: *The system you have entered has for any initial condition a unique piecewise continuous solution which can be extended at least until an accumulation of event times occurs.*

A number of variations on the above theme can be found in the literature, both with weaker and stronger assumptions and with corresponding weaker or stronger conclusions; see [31] for a survey. For instance, if the result of the above algorithm is positive and moreover in the construction of the matrices N and M the first nonzero element was already found at the first or second step, then one can guarantee existence and uniqueness of solutions on intervals of arbitrary length; moreover, it is guaranteed that there will be no jumps in the trajectories of the state variable x except possibly at the initial time.

Algorithms like the one above are at this moment rarely found in commonly available simulation software. It is to be expected though that as hybrid system simulators are being developed there will be a demand for analysis methods that are capable of answering questions concerning well-posedness and other properties of interest. Although it will not be feasible to write such algorithms for free-form hybrid systems, we see from the above that e.g. for complementarity systems it is possible to come up with effective tests. Moreover, the theoretical methods that are used in well-posedness investigations have links to computational methods that are useful in carrying out simulation runs.

5.3 Relations to control theory

There are several different ways in which complementarity systems come up in control problems; these connections are briefly summarised next. In Section 6 we will see that complementarity conditions can be used to describe saturation effects. Since the relation $v = \max(w, z)$

[5]It follows from the Cayley-Hamilton theorem that if $CA^iBe_j = 0$ for $i = 0, 1, \ldots, n-1$ then actually $CA^iBe_j = 0$ for all $i = 0, 1, 2, \ldots$, so that the singularity can be detected in a finite number of steps.

is equivalent to the linear equations $v = w + u = z + y$ together with the complementarity conditions $u \geq 0$, $y \geq 0$, and $u = 0$ or $y = 0$, actually all systems that are described in terms of "max" operations can be written as complementarity systems. A similar statement holds for systems that can be seen as interconnections of smooth input/output systems with ideal relays; such a description applies to many switching control schemes such as sliding-mode control.

Complementarity conditions are often related to variational principles, and therefore it is no surprise that complementarity systems come up in optimal control problems. For instance, the necessary conditions for optimality that are derived from Pontryagin's maximum principle have a complementarity form. Another relation to optimal control is based on the fact that space discretisation of the Hamilton-Jacobi-Bellman equations of dynamic programming for problems with finite decision sets leads to a variable-structure system; such systems can be rewritten in complementarity form.

Without doubt the reader has noticed that, when the complementarity conditions are stripped from the description of a complementarity system, the resulting equations are exactly those that have been studied extensively in system theory. This fact gives reason to believe that concepts developed in system and control theory will be useful in the study of complementarity systems, and indeed this is borne out in the references mentioned above. Here we actually have an application of control theory to complementarity systems rather than vice versa. For a summary of applications of complementarity systems see also [30].

6 First order linear hybrid systems subject to saturation

In this section we consider the problem of designing optimal switching time sequences for first order linear hybrid systems subject to saturation. In general this leads to a mathematical programming problem, which we have called the Extended Linear Complementarity Problem (ELCP). This problem also appears in the analysis and control of other classes of hybrid systems such as, e.g. linear complementarity systems (see Section 5). Although the general ELCP is an NP-hard[6] problem [19], it can be shown that if there is no upper saturation then for some objective functions the optimal switching time sequence can be computed very efficiently.

6.1 The Extended Linear Complementarity Problem

The Extended Linear Complementarity Problem (ELCP) is defined as follows [19]:

> Given $A \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{q\times n}$, $c \in \mathbb{R}^p$, $d \in \mathbb{R}^q$ and m subsets ϕ_1, ϕ_2, $\ldots$, ϕ_m of $\{1, 2, \ldots, p\}$, find $x \in \mathbb{R}^n$ such that
>
> $$\sum_{j=1}^{m} \prod_{i\in\phi_j} (Ax - c)_i = 0$$
>
> subject to $Ax \geqslant c$ and $Bx = d$, or show that no such x exists.

[6]Recall that this means that there does not exist a polynomial-time algorithm that solves the problem unless $P = NP$ [23].

The ELCP can be considered as a system of linear equations and inequalities ($Ax \geqslant c$, $Bx = d$), where we can distinguish m groups of linear inequalities (one group for each index set ϕ_j) such that in each group at least one inequality should hold with equality (i.e. its residue $(Ax - c)_i$ should be equal to 0). In [19] we have developed an algorithm to compute the complete solution set of an ELCP, which in general consists of a union of faces of the polyhedron defined by the system $Ax \geqslant c$, $Bx = d$.

6.2 A class of first order linear hybrid systems with saturation

Consider a system the evolution of which is characterised by consecutive phases. In each phase each state variable exhibits a linear growth or decrease until a certain upper or lower saturation level is reached; then the state variable stays at that level until the end of the phase. A system the behaviour of which satisfies this description will be called a *first order linear hybrid system with saturation*. So the evolution of the ith component of the state vector q of the system in the kth phase is given by

$$\frac{dq_i(t)}{dt} = \begin{cases} \alpha_{i,k} & \text{if } l_{i,k}^{\text{sat}} < q_i(t) < u_{i,k}^{\text{sat}} \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

where $\alpha_{i,k}$, $l_{i,k}^{\text{sat}}$ and $u_{i,k}^{\text{sat}}$ are respectively the net growth rate, the lower saturation bound and the upper saturation bound of the ith component of the state vector in the kth phase.

Note that the system described above can indeed be considered as a hybrid system. The system can operate in several modes (one mode for each phase) and in each mode the behaviour of the system is characterised by differential equations of the form (10).

A typical example of a first order linear hybrid system with saturation is a traffic signal controlled intersection provided that we use a continuous approximation for the queue lengths (see Section 6.4 and [21]). The state variables of this system correspond to the queue lengths in the different lanes. For a traffic signal controlled intersection the lower bound for the queue length is 0 since the queue length is always nonnegative. The upper bound could correspond to the maximal available storage space due to the distance to the preceding junction or due to the layout of the intersection. We assume that if this upper bound is reached then newly arriving cars take another route to get to their destination.

Another example of a first order linear hybrid system with saturation is a system consisting of several fluid containers that are connected by tubes with valves and that have a supply opening at the top and two outlets: one at the bottom (with a tube that leads to another fluid container), and one at the top (so that the fluid level in the containers can never exceed a given level). Here we assume that the increase or decrease of the fluid levels is linear if the system is not saturated.

Let us now derive the equations that describe the evolution of the state of a first order linear hybrid system with saturation at the switching time instants, i.e. the time instants at which the system switches from one phase to another. In analogy with a traffic signal controlled intersection, we use the word "queue lengths" to refer to the state variables of a first order linear hybrid system with saturation in the remainder of this section. Note however that our definition of a first order linear hybrid system with saturation is not limited to queueing systems only. Let M be the number of "queues". The length of queue i at time t is denoted by $q_i(t)$. The evolution of the system begins at time t_0. Let $t_1, t_2, t_3, \ldots$ be the switching time instants. The length of the kth phase is equal to $\delta_k = t_{k+1} - t_k$. Equation

(10) implies that

$$q_i(t_{k+1}) = \max\big(\min\left(q_i(t_k) + \alpha_{i,k}\delta_k,\, u_{i,k}^{\text{sat}}\right),\, l_{i,k}^{\text{sat}}\big)$$

for $k = 0, 1, 2, \ldots$ Note that the sequence $t_0, t_1, t_2, \ldots$ is not an equidistant sequence (i.e. $t_k \neq k\Delta$ for some $\Delta \in \mathbb{R}$). So (10) really describes an asynchronous system and not a regular (sampled) discrete-time system. If we define $q_{i,k} = q_i(t_k)$ and if we introduce dummy variables $z_{i,k}$, we obtain

$$z_{i,k+1} = \min\left(q_{i,k} + \alpha_{i,k}\delta_k,\, u_{i,k}^{\text{sat}}\right) \tag{11}$$

$$q_{i,k+1} = \max\left(z_{i,k+1},\, l_{i,k}^{\text{sat}}\right) \ . \tag{12}$$

In [18, 20] we have shown that a system of max-min-plus equations such as (11) – (12) can be reformulated as an ELCP.

6.3 Optimal switching time sequences

Suppose that for a given first order linear hybrid system with saturation we want to compute switching time sequences that minimise objective functions such as average queue length over all queues, worst case queue length, average waiting time over all queues, worst case waiting time, etc. Furthermore, we impose extra conditions such as minimum and maximum durations for the phases, minimum and maximum queue lengths, maximum or total durations for two consecutive phases, ... We only consider a finite control horizon, i.e. we consider a finite number N of switching time instants. This leads to a problem of the following form [18]:

$$\text{minimise } J \tag{13}$$

subject to

$$Ax_q + Bx_z + Cx_\delta + d \geqslant 0 \tag{14}$$

$$Ex_q + Fx_z + g \geqslant 0 \tag{15}$$

$$Hx_q + Jx_\delta + k \geqslant 0 \tag{16}$$

$$Lx_q + Px_\delta + q = 0 \tag{17}$$

$$\sum_i (Ax_q + Bx_z + Cx_\delta + d)_i \,(Ex_q + Fx_z + g)_i = 0 \ , \tag{18}$$

where J is the objective function we want to minimise; the vector x_q contains the queue lengths $q_{i,k}$ in the different lanes at the switching time instants, the vector x_z contains the dummy variables $z_{i,k}$, and the vector x_δ contains the durations of the phases. It is easy to verify that the system (14) – (18) is a special case of an ELCP. In order to determine the optimal switching time sequence we could first determine the solution set of this ELCP and then minimise the objective function J over this solution set. The algorithm of [19] to compute the solution set of a general ELCP requires exponential execution times. This implies that the approach sketched above is not feasible if the number of switching cycles is large. Alternative approaches are the use of nonlinear constraint optimisation, unconstrained optimisation using penalty functions, or using a moving horizon approach [18, 21]. Note however that in general these approaches only yield suboptimal solutions.

If there is no upper saturation, then there exist very efficient approaches to determine optimal switching time sequences: for the objective functions average or worst case waiting

Period	T_1	T_2
t_0–t_1	red	green
t_1–t_2	red	amber
t_2–t_3	green	red
t_3–t_4	amber	red
t_4–t_5	red	green
t_5–t_6	red	amber
$\vdots$	$\vdots$	$\vdots$

Table 1: The traffic signal switching scheme. The column with label T_1 corresponds to the traffic signals on the first street of the intersection, and the column with label T_2 corresponds to the traffic signals on the other street.

time or queue length we can transform the problem into an optimisation problem over a convex feasible set [18]. This approach allows us to compute the globally optimal switching time sequence very efficiently. Furthermore, for the objective functions average queue length and average waiting time we can even make some approximations that transform the problem into a linear programming problem.

6.4 Example: Optimal traffic signal control

Let us now consider a practical application of the design of optimal switching time sequences for linear hybrid systems with saturation. We consider an intersection of two two-way streets with controllable traffic signals on each corner. On each corner of the intersection there is a traffic signal. For each traffic signal there are three subsequent phases: green, amber, and red. The switching scheme for the intersection is given in Table 1. Since the queue lengths can never become negative, we have $l_{i,k}^{\text{sat}} = 0$ for all i, k. The upper saturation bounds $u_{i,k}^{\text{sat}}$ are determined by the layout of the intersection and the distance to the neighbouring intersections.

In order to obtain a model that fits the framework of Section 6.2, we make the following assumptions:

- the queue lengths are continuous variables,
- the average arrival and departure rates of the cars are constant or slowly time-varying,

These assumptions can be justified as follows. Recall that the ultimate goal is the design of optimal traffic signal switching time sequences. Since this is only useful if the arrival and departure rates of vehicles at the intersection are high, approximating the queue lengths by continuous variables introduces only small errors. Furthermore, assuming that the average arrival and departure rates are constant is not a serious restriction, provided that we use an *adaptive moving horizon strategy*: we compute the optimal switching time sequence for, say, the first 10 cycles, based on a prediction of the average arrival and departure rates (using historical data and/or data measured during the previous cycles) and we apply this

sequence during the first of the 10 cycles, meanwhile we update our estimates of the arrival and departure rates, compute a new optimal sequence for the next 10 cycles, and so on.

Now it is easy to verify that the evolution of the queue lengths is described by equations of the form (10) with $\alpha_{i,k}$ equal to the difference between the arrival and departure rates of cars in lane i and phase k. If there is no upper saturation then we can very efficiently compute optimal and suboptimal traffic signal switching time sequences using the methods discussed in Section 6.2.

The approach discussed above can also be extended to more complex intersections and/or switching schemes (see [17]).

7 Conclusions

In this paper we have presented some of the methods that can be used to address the problems related to the modelling, analysis, and control of hybrid systems. In general there exist two ways to cope with these complexity issues. We can consider special subclasses of hybrid systems (such as linear hybrid systems with saturation or complementarity systems) and make use of the special properties exhibited by this subclass to reduce the complexity of the problem and to come up with efficient analysis and control design techniques. Another way to approach the complexity problem is to make use of the modularity that is often present in large hybrid systems and use modular modelling frameworks such as Petri nets or modular control frameworks such as hierarchical control. However, it is clear that the control of complex man-made systems such as manufacturing systems with a very high flexibility, transportation systems, telecommunication systems, etc. still requires the solution of many different subproblems. Moreover, since hybrid systems are encountered in almost every branch of industry, this means that hybrid systems and control theory will remain a relevant and challenging research domain for the next decades.

Acknowledgements
The results presented in this paper have partially been obtained within the framework of the Belgian Program on Interuniversity Attraction Poles, initiated by the Belgian State, Prime Minister's Office, Science Policy Programming, and of ESPRIT Project 26270: Verification of Hybrid Systems (VHS). The scientific responsibility rests with its authors.
R.K. Boel is supported by the Flemish Foundation for Scientific Research as Research Associate.

References

[1] R. Alur, C. Courcoubetis, and D. Dill, "Model checking in dense real time," *Information and Computation*, vol. 104, pp. 2–34, 1993.

[2] R. Alur, C. Courcoubetis, T.A. Henzinger, P.H. Ho, X. Nicollin, A. Oliveiro, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, pp. 3–34, 1995.

[3] R. Alur and D. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

[4] R. Alur, S. Kanna, and S. La Torre, "Polyhedral flow in hybrid automata," in *Hybrid systems: Computation and Control* (F.W. Vaandrager and J.H. van Schuppen, eds.), vol. 1569 of *Lecture Notes in Computer Science*, pp. 5–18, Berlin, Germany: Springer, 1999.

[5] P. Antsaklis, W. Kohn, M.D. Lemmon, A. Nerode, and S. Sastry, eds., *Hybrid Systems V*, Lecture Notes in Computer Science, Berlin, Germany: Springer-Verlag, 1999. (Proceedings of the 5th International Hybrid Systems Workshop, Notre Dame, Indiana, Sept. 1997).

[6] *Automatica*, vol. 35, no. 3, Mar. 1999. Special Issue on Hybrid Systems.

[7] A. Bemporad and M. Morari, "Verification of hybrid systems via mathematical programming," in *Hybrid systems: Computation and Control* (F.W. Vaandrager and J.H. van Schuppen, eds.), vol. 1569 of *Lecture Notes in Computer Science*, Berlin, Germany: Springer, 1999.

[8] V.D. Blondel and J.N. Tsitsiklis, "Complexity of stability and controllability of elementary hybrid systems," *Automatica*, vol. 35, no. 3, pp. 479–489, Mar. 1999.

[9] R.K. Boel, L. Ben-Naoum, and V. Van Breusegem, "On forbidden state problems for a class of controlled Petri nets," *IEEE Transactions on Automatic Control*, vol. 40, no. 10, pp. 1717–1731, Oct. 1995.

[10] M.S. Branicky, *Studies in Hybrid Systems: Modeling, Analysis, and Control*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1995.

[11] C.G. Cassandras, *Discrete Event Systems: Modeling and Performance Analysis*. The Aksen Associates Series in Electrical and Computer Engineering, Burr Ridge, Illinois: Richard D. Irwin, Inc., 1993.

[12] M.K. Çamlıbel, W.P.M.H. Heemels, and J.M. Schumacher, "The nature of solutions to linear passive complementarity systems," in *Proceedings of the 38rd IEEE Conference on Decision and Control*, Phoenix, Arizona, Dec. 1999.

[13] M.K. Çamlıbel and J.M. Schumacher, "Well-posedness of a class of piecewise linear systems," in *Proceedings of the 5th European Control Conference (ECC'99)*, Karlsruhe, Germany, Aug.–Sept. 1999.

[14] H. Chen, "Synthesis of feedback logic for controlled Petri nets with forward and backward conflict-free uncontrolled subnets," in *Proceedings of the 33rd IEEE Conference on Decision and Control*, Lake Buena Vista, Florida, pp. 3098–3103, Dec. 1994.

[15] A. Chutinan and B. Krogh, "Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations," in *Hybrid systems: Computation and Control* (F.W. Vaandrager and J.H. van Schuppen, eds.), vol. 1569 of *Lecture Notes in Computer Science*, Berlin, Germany: Springer, 1999.

[16] T. Dang and O. Maler, "Reachability analysis via face lifting," in *Hybrid Systems: Computation and Control* (T.A. Henzinger and S. Sastry, eds.), vol. 1386 of *Lecture Notes in Computer Science*, Berlin, Germany: Springer-Verlag, 1998.

[17] B. De Schutter, "The extended linear complementarity problem and optimal traffic light control," Tech. rep., Control Laboratory, Fac. of Information Technology and Systems, Delft University of Technology, Delft, The Netherlands, Jan. 1999. Submitted for publication.

[18] B. De Schutter, "Optimal control of a class of linear hybrid systems with saturation," in *Proceedings of the 38th IEEE Conference on Decision and Control (CDC'99)*, Phoenix, Arizona, Dec. 1999.

[19] B. De Schutter and B. De Moor, "The extended linear complementarity problem," *Mathematical Programming*, vol. 71, no. 3, pp. 289–325, Dec. 1995.

[20] B. De Schutter and B. De Moor, "A method to find all solutions of a system of multivariate polynomial equalities and inequalities in the max algebra," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 6, no. 2, pp. 115–138, Mar. 1996.

[21] B. De Schutter and B. De Moor, "Optimal traffic light control for a single intersection," *European Journal of Control*, vol. 4, no. 3, pp. 260–276, 1998.

[22] J. Desel and J. Esparza, *Free Choice Petri Nets*, vol. 40 of *Cambridge Tracts in Theoretical Computer Science*. 1995.

[23] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W.H. Freeman and Company, 1979.

[24] D.N. Godbole, J. Lygeros, and S. Sastry, "Hierarchical hybrid control: An IVHS case study," in *Proceedings of the 33rd IEEE Conference on Decision and Control*, Orlando, Florida, pp. 1592–1597, Dec. 1994.

[25] M. Greenstreet and I. Mitchell, "Reachability analysis using polynomial projections," in *Hybrid systems: Computation and Control* (F.W. Vaandrager and J.H. van Schuppen, eds.), vol. 1569 of *Lecture Notes in Computer Science*, Berlin, Germany: Springer, 1999.

[26] R.L. Grossman and R.G. Larson, "Viewing hybrid systems as products of control systems and automata," in *Proceedings of the 31st IEEE Conference on Decision and Control*, Tucson, Arizona, pp. 2953–2955, 1992.

[27] W.P.M.H. Heemels, J.M. Schumacher, and S. Weiland, "Complete description of dynamics in the linear complementarity-slackness class of hybrid systems," in *Proceedings of the 36th IEEE Conference on Decision and Control*, San Diego, California, pp. 1243–1248, Dec. 1997.

[28] W.P.M.H. Heemels, J.M. Schumacher, and S. Weiland, "Linear complementarity systems," Tech. rep. 97 I/01, Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands, July 1997. To appear in *SIAM Journal on Applied Mathematics*. This report can be obtained via WWW as `www.cwi.nl/~jms/PUB/ARCH/lcs.ps.Z` or `www.cwi.nl/~jms/PUB/ARCH/lcs_revised.ps.Z` (revised version).

[29] W.P.M.H. Heemels, J.M. Schumacher, and S. Weiland, "Dissipative systems and complementarity conditions," in *Proceedings of the 37th IEEE Conference on Decision and Control*, Tampa, Florida, pp. 4127–4132, Dec. 1998.

[30] W.P.M.H. Heemels, J.M. Schumacher, and S. Weiland, "Applications of complementarity systems," in *Proceedings of the 5th European Control Conference (ECC'99)*, Karlsruhe, Germany, Aug.–Sept. 1999.

[31] W.P.M.H. Heemels, J.M. Schumacher, and S. Weiland, "The rational complementarity problem," *Linear Algebra and Its Applications*, vol. 294, pp. 93–135, 1999.

[32] T.A. Henzinger and S. Sastry, eds., *Hybrid Systems: Computation and Control*, vol. 1386 of *Lecture Notes in Computer Science*, Berlin, Germany: Springer-Verlag, 1998. (Proceedings of the First International Workshop on Hybrid Systems: Computation and Control (HSCC'98), Berkeley, California.

[33] L.E. Holloway, B.H. Krogh, and A. Giua, "A survey of Petri net methods for controlled discrete event systems," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 7, no. 2, pp. 151–190, Apr. 1997.

[34] *IEEE Transactions on Automatic Control*, vol. 43, no. 4, Apr. 1998. Special Issue on Hybrid Systems.

[35] D.M.W. Leenaerts and W.M.G. van Bokhoven, *Piecewise Linear Modeling and Analysis*. Dordrecht, The Netherlands: Kluwer, 1998.

[36] Y.J. Lootsma, A.J. van der Schaft, and M.K. Çamlıbel, "Uniqueness of solutions of linear relay systems," *Automatica*, vol. 35, no. 3, pp. 467–478, Mar. 1999.

[37] J. Lygeros and D.N. Godbole, "An interface between continuous and discrete-event controllers for vehicle automation," *IEEE Transactions on Vehicular Technology*, vol. 46, no. 1, pp. 229–241, Feb. 1997.

[38] J. Lygeros, D.N. Godbole, and S.S. Sastry, "Verified hybrid controllers for automated vehicles," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 522–539, Apr. 1998.

[39] N. Lynch, R. Segala, F. Vaandrager, and H.B. Weinberg, "Hybrid I/O automata," in *Hybrid Systems III: Verification and Control* (R. Alur, T.A. Henzinger, and E.D. Sontag, eds.), vol. 1066 of *Lecture Notes in Computer Science*, pp. 496–510, 1996.

[40] K. McMillan, "Using unfoldings to avoid the state explosion problem in the verification of asynchronous cicuits," in *Computer Aided Verification* (G. von Bochmann and D.K. Probst, eds.), vol. 663 of *Lecture Notes in Computer Science*, Berlin, Germany: Springer-Verlag, 1993.

[41] J. Moody and P. Antslakis, *Supervisory Control of Discrete Event Systems Using Petri Nets*. Kluwer Academic Press, 1998.

[42] A.S. Morse, ed., *Control using logic-based switching*. No. 222 in Lecture Notes in Computer Science, Berlin: Springer, 1996.

[43] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.

[44] A.G.O. Mutambara, *Decentralized Estimation and Control for Multisensor Systems*. Boca Raton, Florida: CRC Press LCC, 1998.

[45] G.J. Pappas, G. Lafferriere, and S. Sastry, "Hybrid systems with finite bisimulations," in *Hybrid Systems V* (P. Antsaklis, W. Kohn, M.D. Lemmon, A. Nerode, and S. Sastry, eds.), Lecture Notes in Computer Science, Berlin, Germany: Springer-Verlag, 1999.

[46] J.L. Peterson, "Petri nets," *Computing Surveys*, vol. 9, no. 3, pp. 223–252, Sept. 1977.

[47] F. Pfeiffer and C. Glocker, *Multibody Dynamics with Unilateral Contacts*. Chichester, UK: John Wiley & Sons, 1996.

[48] A. Puri, *Theory of Hybrid Systems and Discrete Event Sytems*. PhD thesis, U.C. Berkeley, 1995.

[49] S. Sastry, G. Meyer, C. Tomlin, J. Lygeros, D. Godbole, and G. Pappas, "Hybrid control in air traffic management systems," in *Proceedings of the 34th IEEE Conference on Decision and Control*, New Orleans, Louisiana, pp. 1478–1483, Dec. 1995.

[50] D.D. Šiljak, *Decentralized Control of Complex Systems*. San Diego, California: Academic Press, 1991.

[51] E.D. Sontag, "Nonlinear regulation: The piecewise linear approach," *IEEE Transactions on Automatic Control*, vol. 26, no. 2, pp. 346–358, Apr. 1981.

[52] E.D. Sontag, "Remarks on piecewise-linear algebra," *Pacific Journal of Mathematics*, vol. 98, pp. 183–201, 1982.

[53] E.D. Sontag, "Interconnected automata and linear systems: A theoretical framework in discrete-time," in *Proceedings of the Workshop on Verification and Control of Hybrid Systems* (R. Alur, B. Kurshan, and E.D. Sontag, eds.), Berlin, pp. 436–448, Springer-Verlag, 1996.

[54] G. Stremersch and R.K. Boel, "On the influencing net and forbidden state control of timed Petri nets with forced transitions," in *Proceedings of the 37th IEEE Conference on Decision and Control*, Tampa, Florida, pp. 3287–3292, Dec. 1998.

[55] C. Tomlin, G. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A case study in multi-agent hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 509–521, Apr. 1998.

[56] F.W. Vaandrager and J.H. van Schuppen, eds., *Hybrid systems: Computation and Control*, vol. 1569 of *Lecture Notes in Computer Science*, Berlin, Germany: Springer, 1999. (Proceedings of the Second International Workshop on Hybrid Systems: Computation and Control (HSCC'99), Berg en Dal, The Netherlands, Mar. 1999).

[57] A.J. van der Schaft and J.M. Schumacher, "The complementary-slackness class of hybrid systems," *Mathematics of Control, Signals, and Systems*, vol. 9, no. 3, pp. 266–301, 1996.

[58] A.J. van der Schaft and J.M. Schumacher, "Complementarity modeling of hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 483–490, Apr. 1998.

[59] A.J. van der Schaft and J.M. Schumacher, *An Introduction to Hybrid Dynamical Systems*, vol. 251 of *Lecture Notes in Control and Information Science*. London: Springer, 2000. Available Nov. 1999.