A Framework for Control, State Estimation, Fault Detection, and Verification of Hybrid Systems

Manfred Morari, Alberto Bemporad, Domenico Mignone

Prof. Dr. Manfred Morari In 1994 Manfred Morari was appointed head of the Automatic Control Laboratory at the Swiss Federal Institute of Technology (ETH) in Zurich. His theoretical interests are in the areas of robust control, predictive control and hybrid systems. He and his group are applying novel control concepts to chemical processing systems, biomedical systems, in particular functional electrical stimulation and anesthesia, and noise and vibration control problems.

Adresse: Automatic Control Laboratory, ETH - Swiss Federal Institute of Technology Zurich, Physikstrasse 3, CH-8092 Zurich; phone +41 1 632 2271, fax +41 1 632 1211, e-mail: Morari@aut.ee.ethz.ch; http://control.ethz.ch

Dr. Alberto Bemporad Since 1997 he is helding a postdoctoral position at the Automatic Control Lab, ETH Zurich, Switzerland. He has published paper in the area of model predictive control, constrained control, mobile robotics, and hybrid systems.

Adresse: Automatic Control Laboratory, ETH - Swiss Federal Institute of Technology Zurich, Physikstrasse 3, CH-8092 Zurich; phone +41 1 632 6679, fax +41 1 632 1211, e-mail: Bemporad@aut.ee.ethz.ch; http://control.ethz.ch

Dipl. Ing. ETH Domenico Mignone He received his diploma in electrical engineering from ETH Zurich in 1997. In the same year he joined the Automatic Control Laboratory of ETH Zurich as a PhD student. His research interests include Model Predictive Control and Hybrid Systems.

Adresse:

A Framework for Control, State Estimation, Fault Detection, and Verification of Hybrid Systems

Mixed Logical Dynamical (MLD) systems were introduced as a new system type by the authors recently. The MLD form is capable to model a broad class of systems arising in many applications, among them: linear hybrid systems; sequential logical systems (finite state machines, automata); piecewise linear systems. The paper reviews this modeling paradigm and gives an overview of the many control related problems (optimal feedback, estimation, fault detection) which can be formulated and solved in this framework. Generally, the on-line solution of Mixed-Integer Linear Programs or Mixed-Integer Quadratic Programs is required. Strategic formulations and taylored search procedures are proposed, so that this task should be feasible for problems of reasonable size.

1 Introduction

In many applications, systems include both continuous and discrete components, such as on/off switches or valves, gears or speed selectors. Discrete characteristics are also often introduced by the control system or the specifications which are expressed by a series of if-then-else rules. Such systems consisting of continuous and discrete "components" are commonly referred to as *hybrid systems* (no formal definition exists to the authors' knowledge). Hybrid systems arise in a large number of application areas but the understanding of these systems is rather limited at present. In practice the control of hybrid systems is left to schemes based on heuristic rules inferred from practical plant operation. For the time being, the most common analysis tool is exhaustive simulation.

Our interest in hybrid systems is motivated by several clearly discernible trends in the process industries which point toward an extended need for new tools to design control and supervisory schemes for these systems and to analyze their performance. First, there has been the trend for Programmable Logic Controllers (PLC) and Digital Control Systems (DCS) to approach each other in terms of functionality and the underlying hardware. With the cost of computing power falling rapidly, the hardware capabilities of DCSs and PLCs are expected to become indistinguishable in the next decade. To take full advantage of this trend, tools are needed to tackle the combination of sequence, logic and continuous control tasks in a transparent and efficient manner.

Second, the traditional layers of the control hierarchy (measurements, regulatory control, supervisory control, real time optimization, scheduling and planning) have shown a tendency to merge and the clear boundaries which once existed have disappeared. To make this confluence of the different layers possible new modelling tools are needed which can describe such systems in a unified manner and thus provide the information to allow the control system to make appropriate decisions for

such hybrid systems.

In summary, the rapid advances in computer and information technology are enabling the closer integration of the various decision and control tasks which were traditionally distributed among a broad set of decision makers ranging from PLCs at the lowest level to planning and scheduling departments at the highest level. This integration should eventually lead to a smoother, more responsive and more competitive functioning of the entire organization. It requires the development of new tools for the analysis and synthesis of such large complex systems involving continuous and discrete states whose behavior is governed by dynamics, logical statements and constraints.

The premise of the work described in this paper is that all questions and problems related to hybrid systems are inherently difficult because of their combinatorial nature. Consequently all *useful* techniques must involve significant off-line and/or on-line computation. Against this background we describe a new system type, Mixed Logic Dynamical (MLD) systems, introduced by the authors recently. We argue that many practical problems can be represented in MLD form. Control, estimation, and verification of MLD systems require the solution of Mixed-Integer Linear (or Quadratic) Programs (MILPs or MIQPs). Because efficient techniques not only for MILPs but also for MIQPs are becoming available, this new approach holds much promise for tackling realistic size problems.

The purpose of this paper is to give the reader an overview of the ideas and tools becoming available in this area and to give him/her an appreciation for their potential. New ideas on modeling and optimization techniques will be sketched.

2 Mixed Logic Dynamical (MLD) Systems

Any modeling framework for hybrid systems must be a compromise which circumvents some of the complexities and leads naturally to the formulation of analysis and controller synthesis techniques which are manageable for practical problems. Our formulation is motivated by the following considerations.

- A discrete time description avoids some of the complex behaviors, like an infinite number of switches in an infinitesimal time span. as studied, for example, by [17].
- Limiting the formalism to discrete time is not overly restrictive from a practical point of view because of the sampled-data nature of the control systems, which determine the evolution of these hybrid systems.
- We restrict the dynamics to be linear with the exception that some of the state variables are binary. This greatly simplifies the analysis, but nevertheless permits the description of a broad class of systems.

As we are interested in systems which have both logic and dynamics, we wish to establish a link between the two worlds. In particular, we need to establish how to build statements from operating events concerning physical dynamics. The key idea is to use techniques described, for example, in [28, 10, 24] to transform propositional logic into *mixed-integer linear inequalities*, i.e. linear inequalities involving both *continuous variables* $x \in \mathbb{R}^n$ and *binary/logical variables* $\delta \in \{0,1\}$.

The resulting Mixed Logic Dynamical (MLD) systems are described through the following linear relations

$$x(t+1) = Ax(t) + B_1u(t) + B_2\delta(t) + B_3z(t) \quad (1a)$$
$$y(t) = Cx(t) + D_1u(t) + D_2\delta(t) + D_3z(t) \quad (1b)$$
$$E_2\delta(t) + E_3z(t) \leq E_1u(t) + E_4x(t) + E_5 \quad (1c)$$

where

$$x = \begin{bmatrix} x_c \\ x_\ell \end{bmatrix},\ x_c \in \mathbb{R}^{n_c},\ x_\ell \in \{0,1\}^{n_\ell},\ n \triangleq n_c + n_\ell$$

is the state of the system, with the x_c components continuous and the x_ℓ components 0-1 . The outputs y and the inputs u are partitioned similarly. The auxiliary logical and continuous variables are represented by $\delta \in \{0,1\}^{r_\ell}$ and $z \in \mathbb{R}^{r_c}$, respectively.

The justification for the MLD form is that it is capable to model a broad class of systems arising in many applications [7]: linear hybrid systems; sequential logical systems (finite state machines, automata); nonlinear dynamic systems, where the nonlinearity can be expressed through combinational logic; some classes of discrete event systems; constrained linear systems. (Here the terms "combinational" and "sequential" are borrowed from digital circuit design jargon.) More importantly, the MLD formalism leads to the formulation of various verification, control and estimation problems in terms of MILPs or MIQPs, for which efficient algorithms are available. These problems have not been successfully addressed by other tools or only with a much higher computational effort. In a sense the ends justify the means here.

A simple illustrative example will be presented next. It will be followed by some new procedures to put propositional logic into MLD form.

2.1 Example

Consider for instance the simple automaton and linear system depicted in Fig. 1, and described by the relations

$$\begin{cases} [x_\ell(t) = 0] \wedge [x_c \leq 0] \rightarrow [x_\ell(t+1) = 0] \\ [x_\ell(t) = 0] \wedge [x_c > 0] \rightarrow [x_\ell(t+1) = 1] \\ [x_\ell(t) = 1] \rightarrow [x_\ell(t+1) = 0] \\ x_c(t+1) = ax_c(t) + bu(t) \end{cases} \quad (2)$$

The (0-1) finite state $x_\ell(t)$ remains in 0 as long as the continuous state $x_c(t)$ is non-positive. If $x_c(t) > 0$ at some t, then x_c generates a digital impulse, i.e. $x_\ell(t+1) = 1$, $x_\ell(t+2) = 0$. Hence the automaton's dynamics is driven by events generated by the underlying linear system.

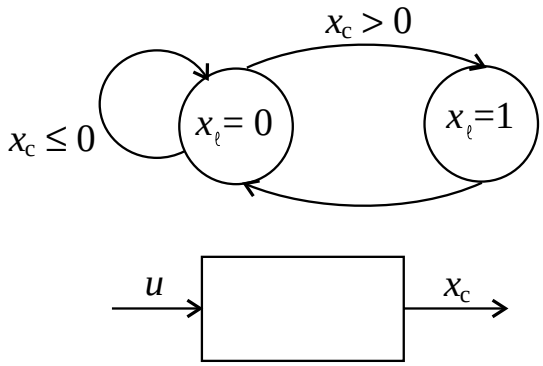

Bild 1: Automaton driven by conditions on an underlying dynamic system

Let $x \triangleq [x_c' \; x_\ell']'$, and introduce the auxiliary logical variables $\delta_1(t)$, $\delta_2(t)$ defined as

$$[\delta_1(t) = 1] \;\leftrightarrow\; [x_c(t) \le 0] \tag{3a}$$
$$\delta_2(t) = x_\ell(t+1) \tag{3b}$$

Equation (3a) can be rewritten as

$$x_c(t) \le M(1-\delta_1(t)) \tag{4a}$$
$$x_c(t) \ge \varepsilon + (m-\varepsilon)\delta_1(t) \tag{4b}$$

where M and m are upper and lower bounds on x_c, respectively, and $\varepsilon > 0$ is a small tolerance (machine precision). Moreover, from (2), it follows that

$$[\delta_2(t) = 1] \;\leftrightarrow\; [x_\ell(t) = 0] \;\wedge\; [\delta_1(t) = 0]$$

or $\delta_2(t) = (1-x_\ell(t))(1-\delta_1(t))$. Hence,

$$\delta_2(t) \le (1-\delta_1(t)) \tag{5a}$$
$$\delta_2(t) \le (1-x_\ell(t)) \tag{5b}$$
$$\delta_2(t) \ge (1-\delta_1(t)) + (1-x_\ell(t)) - 1 \tag{5c}$$

The mixed-integer linear inequalities (4)-(5) define the automaton part in system (2), which hence is an MLD system.

2.2 Expressing Propositional Logic in MLD Form

Logic components of hybrid systems can be described with propositional logic formulas. Proper processing of the propositional logic problem can produce large benefits for the numerical solution of the mixed integer program, which results in the analysis as well as controller and estimator synthesis problems for MLD systems.

There are three ways, how logic propositions can be translated into linear inequalities. These are:

- the substitution method
- the conjunctive normal form approach
- the truth table method

2.2.1 Substitution Method

In the substitution method the propositional logic formula is successively simplified by introducing additional binary variables for the subexpressions occurring in it [10]. The linear inequalities describing the overall logic proposition are given by the set of all linear inequalities introduced during the substitution procedure. A simple example illustrates the method:

Example Assume that in a reactor the heater should be turned on if either of the two following conditions is true:

- The temperature of the reactants is low and there is sufficient mass in the reactor
- The system operator increases the production rate

This qualitative description can be translated into the following logic proposition

$$[\delta_H = 1] \Leftarrow \Big([\delta_T = 0] \wedge [\delta_M = 1]\Big) \vee [\delta_u = 1] \tag{6}$$

Here, all δ's are Boolean variables with the following meaning: δ_H is a control signal to the heating, δ_T and δ_M are temperature and mass indicators, δ_u is the external operator signal. Using the substitution method, we introduce a new Boolean variable δ_i, denoting the internal causes for the heating being turned on. This variable is defined as:

$$[\delta_i = 1] \Leftrightarrow [\delta_T = 0] \wedge [\delta_M = 1] \tag{7}$$

Using the tables in [7] this relation can be directly translated into linear inequalities. The original expression reduces to

$$[\delta_H = 1] \Leftarrow [\delta_i = 1] \vee [\delta_u = 1] \tag{8}$$

for which the translation into equivalent inequalities is also given in tables [7].

The substitution method has the advantage that no preprocessing of the logic expression is required. The big disadvantage however is that additional variables are introduced, that enlarge the description of the logic relation.

2.2.2 Conjunctive Normal Form

The introduction of additional Boolean variables can be avoided, if the expression is first translated into conjunctive normal form (CNF) [10]. Each term in the CNF (i.e. each disjunction) gives rise to one inequality. This is again illustrated for the example above. The CNF of the logic expression in (6) is:

$$([\delta_u = 0] \vee [\delta_H = 1]) \wedge ([\delta_H = 1] \vee [\delta_T = 1] \vee [\delta_M = 0]) \wedge$$
$$([\delta_H = 0] \vee [\delta_M = 1] \vee [\delta_u = 1]) \wedge ([\delta_H = 0] \vee [\delta_T = 0] \vee [\delta_u = 1])$$

For the whole expression to be true, each bracket must be true, i.e. the following four inequalities must be satisfied:

$$1 - \delta_u + \delta_H \ge 1 \tag{9}$$
$$\delta_H + \delta_T + 1 - \delta_M \ge 1 \tag{10}$$
$$1 - \delta_H + \delta_M + \delta_u \ge 1 \tag{11}$$
$$1 - \delta_H + 1 - \delta_T + \delta_u \ge 1 \tag{12}$$

2.2.3 Truth Table Methods

Recently, we succeeded [22] in developing the following alternative method which generates a set of linear inequalities corresponding to any complex logical expression without introducing any auxiliary variables. First, for each binary expression of binary variables $X_n = F(X_1, X_2, \ldots X_{n-1})$ the *truth table* is calculated, showing the result X_n for each possible combination of values for $X_1, X_2, \ldots$ row by row. We proved [22] that the

polytope P obtained as the convex hull of the points defined by the rows of the truth table describes the logical expression with a minimal number of binary variables.

Note that the last two methods do not require the introduction of additional Boolean variables. They require, however, a pre-processing of the logic proposition.

2.3 Automated Model Generation

The transformation of first principles hybrid system descriptions into MLD form requires the application of a set of given rules, like the transformation technique just described. It is lengthy and tedious and is therefore a task that is preferably automated. A compiler has been developed [3] that produces the matrices A, B_i, C, D_i and E_i in (1). The problem specification language to the compiler is HYSDEL (HYbrid System DEscription Language).

3 Theoretical Properties of Mixed Logic Dynamical Systems

In principle, the inequality (1c) might be satisfied for many values of $\delta(t)$ and/or $z(t)$. In order to define *trajectories* in the x and y-space for system (1), we wish that $x(t+1)$ and $y(t)$ are uniquely determined by $x(t)$ and $u(t)$, i.e., that the system is *well posed*. A simple numerical test for checking this property has been developed and is reported in [7]. It is based on a feasibility check of an MILP.

Needless to say, well-posedness is a *minimal* requirement for the MLD description to be meaningful. For control, reachability and controllability must be understood. For estimation, reconstructibility and observability are important properties.

These questions of controllability, observability, etc. are inherently difficult [1, 20] but some progress has been made [4]. To illustrate the ideas and the unusual behavior which can occur we will briefly discuss our work on observability here. Similar to [19, 25] we can define incremental observability without loss of generality in the following manner. The MLD system (1) is incrementally observable if and only if there exists a scalar $w > 0$ such that, $\forall x_1, x_2 \in X(0)$,

$$\sum_{t=0}^{T-1} \|y(t,x_1) - y(t,x_2)\|_\infty \geq w\|x_1 - x_2\|_1 \qquad (13)$$

For fixed values of T and w the incremental observability of an MLD system can be checked by solving an MILP. The parameters T and w appearing in this definition have practical significance. If w is very small state estimation would become difficult in the presence of noise. If T is very large, long observation periods would be required to arrive at state estimates.

Incremental observability must be tested on a case-by-case basis via condition (13). No structural properties are apparent. Even for a piece-wise linear system, a special case of an MLD system, the observability measure T is not related to the order of the constituting linear systems as is the case for LTI systems. Also the combination of observable LTI systems into a piecewise linear system is not necessarily observable. The reverse does not hold either. The combination of LTI systems which are by themselves not observable may be observable [4].

4 Control

First open loop optimal controllers will be formulated. Then the closed loop effect is accomplished by applying these open-loop optimal controllers iteratively in a receding horizon fashion.

4.1 Optimal Control of MLD Systems

For an MLD system of form (1), consider the following problem. Given an initial state x_0 and a final time T, find (if it exists) the control sequence $u_0^{T-1} \triangleq \{u(0), u(1), \ldots, u(T-1)\}$ which transfers the state from x_0 to x_f and minimizes the *performance index*

$$J(u_0^{T-1}, x_0) \triangleq \sum_{t=0}^{T-1} \|u(t) - u_f\|_{Q_1}^2 + \|\delta(t, x_0, u_0^t) - \delta_f\|_{Q_2}^2 + \|z(t, x_0, u_0^t) - z_f\|_{Q_3}^2 + \|x(t, x_0, u_0^{t-1}) - x_f\|_{Q_4}^2 + \|y(t, x_0, u_0^{t-1}) - y_f\|_{Q_5}^2 \qquad (14)$$

subject to the terminal constraint

$$x(T, x_0, u_0^{T-1}) = x_f \qquad (15)$$

and the MLD system dynamics (1a), where $\|x\|_Q^2 \triangleq x'Qx$; $Q_i = Q_i' \geq 0$, $i = 1, \ldots, 5$, are given weight matrices, and x_f, u_f, δ_f, z_f, y_f satisfy (1) in steady state for $x(t+1) = x(t) = x_f$.

This problem can be solved as a *Mixed-Integer Quadratic Program* (MIQP).

4.2 Predictive Control

Finding a stabilizing control law for an MLD system is not easy, because the system is neither linear nor even smooth. *Model predictive control* [16] provides tools to succeed in this task. In brief, one has to solve an optimization problem of the form (14)–(15) at each time step t, by finding an optimal input sequence $\{u^*(t+k)\}_{k=0,\ldots,T-1}$. Then, only the first move is applied to the plant, i.e. $u(t) = u^*(t+0)$, and the whole optimization procedure is repeated at time $t+1$, when new measurements $x(t+1)$ are available.

By appropriately defining the concepts of equilibrium and stability for MLD systems, and by using Lyapunov arguments it can be proven [7] that the control law, obtained by repeatedly solving (14)–(15) at each time step t, stabilizes the system.

>From the proof it follows that local minima do not affect stability, although the performance deteriorates. This is particularly appealing when the available computational power does not allow the full solution of the MIQP problem (14)–(15).

5 Moving Horizon Estimation for MLD Systems

The dual problem of model predictive control, i.e. the moving horizon estimation problem can also be formulated in terms of an MIQP. The goals of such an estimation can be varied, like state estimation, fault detection or disturbance estimation. The common feature in all these problems is the minimization of a quadratic cost function involving the quantities to be estimated. Contrary to the control problem, the estimation horizon extends backwards in time, allowing at time t to estimate the quantities of interest at times prior to t.

One important application of moving horizon estimation is fault detection. Many techniques for fault detection are modeling faults as additive unknown inputs affecting a linear system. Fault detection is then equivalent to determining if the estimated inputs exceed a certain threshold value. The MLD system framework allows the designer the formulation of more realistic fault detection problems. Faults can also be modeled as unmeasured binary disturbances affecting the system in a multiplicative manner. A faulty actuator, for example, is represented much more accurately in this manner.

To perform fault detection in the MLD framework, we assume that the dynamics of the system in the presence of each fault is known. To model the possibly faulty behavior of the system, we extend the MLD framework by adding three unmeasured variables:

- Fault, i.e. binary disturbance $\phi(t) \in \{0,1\}^f$
- Input disturbance $\xi(t) \in \mathbb{R}^n$
- Output disturbance $\zeta(t) \in \mathbb{R}^p$

At each time t the estimates of the faults $\hat{\phi}(t)$ and states $\hat{x}(t)$ are obtained by solving a least squares problem (MIQP) over a horizon extending backwards in time. Under certain mild assumptions the stability of the estimator can be guaranteed [6].

6 Computational Aspects

One drawback of the methods summarized in this paper lies in the complexity of the MILPs and MIQPs that must be solved. These types of optimization problems exhibit an exponential increase of the worst case complexity with an increasing number of binary variables. However, this does not necessarily preclude the application of the method. There are at least three ways to alleviate this problem:

- Devise new branch and bound strategies
- Deal with suboptimal solutions
- Move as many computations as possible offline

6.1 New Branch and Bound Strategies

Branch and bound methods are considered widely to be best for the solution of mixed integer quadratic programs [14]. However, branch and bound does not define a particular algorithm, but rather a whole class of methods that differ in the implementation details. Common to all branch and bound methods for mixed integer programming problems is the generation of a set of easier subproblems arranged in a tree structure [23, 15]. Let us assume that the MIQP is given in the following form:

$$\min_x \quad x^T Q x + b^T x \tag{16}$$

$$\text{subject to} \quad Cx + d \leq 0$$
$$x = \begin{bmatrix} x_c \\ x_d \end{bmatrix},\ x_c \in \mathbb{R}^{n_c}$$
$$x_d \in \{0,1\}^{n_d} \tag{17}$$

The idea of solving MIQPs with branch and bound methods relies on the relaxation of the integrality constraints (17), i.e. binary variables are allowed to span over the whole continuous interval $[0,1]$. We shall refer to a relaxed problem as a subproblem. The optimal values of the subproblems, if they exist, represent lower bounds on the optimal value of the original MIQP [14].

The other idea the branch and bound methods rely on is the concept of separation, i.e. the generation of new subproblems. Let ξ be a vector of dimension n_d and let the symbol $\star$ mean that the corresponding entry of ξ is relaxed, i.e. free to span the interval $[0,1]$. We associate the original MIQP without integrality constraints (17) with

$$\xi_0 = \underbrace{[\star\ ,\ \star\ ,\ \ldots\ ,\ \star]}_{n_d \quad times} \tag{18}$$

The vector ξ_0 will be assigned to the root of a binary tree. The separation of the original MIQP or any subproblem into relaxed QPs is done by setting selected integer variables to 0 or 1. The resulting new QP problems are assigned to the children of the node. We denote each child by a vector ξ_j, $\xi_j \in \{\star,0,1\}^{n_d}$. If the i-th component $\xi_j^i = 0$ (or $\xi_j^i = 1$), then the QP corresponding to that node is solved by setting the i-th binary variable to 0 (or 1). If $\xi_j^i = \star$, then the i-th binary variable of ξ_j is regarded as free within $[0,1]$. As an example, consider an MIQP with 3 binary variables. The corresponding binary tree is shown in Fig. 2.

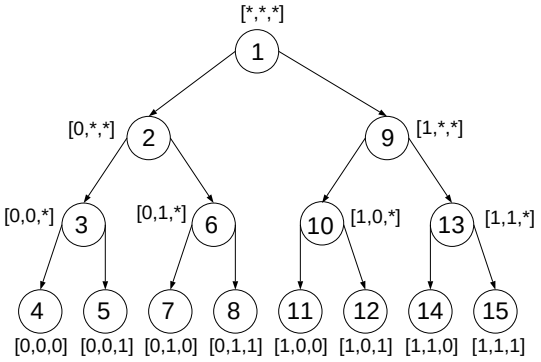

Bild 2: The binary tree for a MIQP with 3 integer variables. Each node is marked with the corresponding vector ξ_j. The numbers denote the order how the problems are solved in the depth first strategy.

One important decision that affects the average computational times is the order in which the subproblems

are solved, i.e. the *tree exploring* or *node selection* strategy [15]. Two standard choices are:

Depth First Strategy: The QPs are solved by a last-in first-out (LIFO) rule.

Breadth First Strategy: The problems at depth N are not solved before all problems at depths $N-1$ have been solved.

This choice is important because the solution of one subproblem ξ might give us the certificate, that all the future subproblems generated from ξ, cannot yield the optimal solution. In this case there will be no point in wasting computational time to solve them.

The average solution time can vary considerably according to the tree exploring strategy. We have experimented successfully with a strategy which assumes that the binary variables change only infrequently over the considered time horizon [5]. In fact, typically binary variables $\delta(t)$ are associated with conditions on continuous states $x(t)$, for instance $[\delta(t)=1] \leftrightarrow [x(t) \geq 0]$. Because the continuous components satisfy dynamic equations, their inertia will, in general, prevent frequent switches of the indicator variable $\delta(t)$. This phenomenon is even more pronounced when integer variables represent the occurrence of faults that do not occur very often and that are not recoverable. This involves an irreversible physical damage, and the integer variable will switch at most once its value over the horizon $[t-T,t]$.

The tree exploring strategy proposed in [5] chooses to solve those relaxed problems first, that have few switches in the binary variables. The switches are determined via the vector ξ_j assigned to each subproblem.

6.2 Suboptimal Solutions

The available time for the online computations is often limited by hard bounds. For our computations it is therefore necessary to consider the possibility that we have to deal with suboptimal solutions of the mixed integer optimization problems. For control purposes it is not critical to find the global optimum to guarantee stability, a feasible suboptimal solution suffices [7]. However, we would like to use the available time to solve those subproblems that are most promising in giving the optimal solution. Indeed, it can occur, that the optimal solution is found quite early, but the branch and bound algorithm keeps on doing many computations just to verify that the currently best solution is actually the global optimum.

The method described in [5] and briefly outlined in the previous section has proven its usefulness also in the case, where for each MIQP we limited the number of QPs the algorithm is allowed to solve. If the assumption is fulfilled that the optimum is likely to have few switches of the binary variables over the horizon, the method will already have gone through those candidates at the time when we are stopping the computations. This amounts to cutting off the verification process, which simply confirms that the currently best solution is the global optimum.

6.3 Move Computations Offline

One obvious approach to deal with the problem complexity is to move as many computations as possible offline and to restrict the online computations to a minimum. This goal can be achieved to some extent in the modeling phase, by using as few binary variables as possible (see section 2.2).

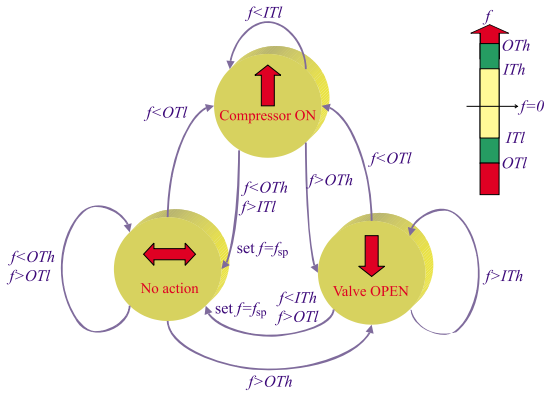

Bild 3: Hybrid automaton for the controller

6.4 Some Remarks about the Solvers

CPLEX [18] is the most widely used commercial code for solving MILPs. We have used the research codes by Fletcher and Leyffer [14] and by Sahinidis [26] to solve the MIQPs arising in the optimal control and estimation problems. There the key is to pair an efficient sparse QP code for the relaxed problems with a good tree exploring strategy.

7 Examples

The following examples are chosen to communicate the power and versatility of the proposed framework. More details and examples can be found in [7].

7.1 Verification of an Automotive Electronic Height Control System

The chassis level of a car is controlled by a pneumatic suspension system. The level is raised by pumping air into the system, and lowered by opening an escape valve. For the sake of simplicity, as in [27, 12], we consider an abstract model including only one wheel. The suspension system is commanded by a logic controller, whose behavior is represented in Fig. 3. In short, the controller switches the compressor on when the level of the chassis is below a certain outer tolerance OTl, off when it reaches again an inner tolerance ITl. It opens the valve when the level is above OTh, and closes it again when the level decreases below ITh. Because of high-frequency disturbances due to irregularities of the road surface, the controller switches based on a filtered version $f(t) = \frac{1}{1+as}h(t)$ of the measured level h of the chassis. The filter is reset to $f=0$ each time f returns within the inner range [ITl, ITh]. The compressor can lift the chassis at a rate $cp(t) \in [cp_{\min}\ cp_{\max}]$, and the escape valve can lower it at a rate $ev(t) \in [ev_{\min}\ ev_{\max}]$.

The study of this system was first proposed in [27], and reconsidered in [12] and [13]. The aim is to verify

that the automotive control system satisfies certain driving comfort requirements. A verification algorithm for MLD systems was proposed in [8] and subsequently implemented in Matlab. The algorithm considers the time-evolution of polyhedral subsets in the state space. Using MILPs it determines whether some unsafe region can be reached for some inputs or disturbances.

The algorithm uses interpreted m-code and terminates in 25 min on a Sun SPARCstation 20 with 64 Mb RAM. The research standard HyTech [2] required 62 min on a Sun SparcStation 20 with 128 Mb RAM. In [13], the author reports computation times of up to one day. These data establish the technique based on the MLD formulation as a credible alternative.

7.2 Predictive Control

The three tank system represented in Fig. 4 has been adopted recently as a standard benchmark problem for fault detection and reconfigurable control [21, 9]. Here we used a simplified physical description of the system (more details can be found in [11, 6]). The equations describing the system are simple material balances. The switched nature arises from the fact that flow through the upper horizontal pipes occurs only when the level has reached the level of the pipe. The MLD description

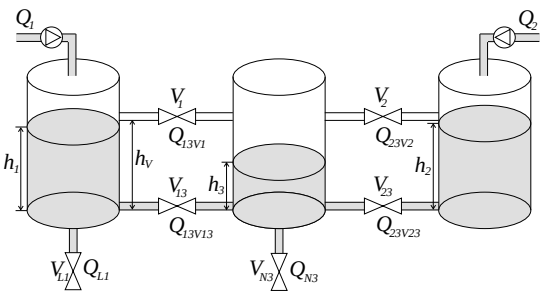

Bild 4: COSY Three-Tank Benchmark.

can be readily derived according to [6].

We applied the model predictive control scheme described in section 4 to solve a regulation problem. In order to stabilize the system to a desired setpoint, the feedback control law resulting from the optimization (14) is adopted with a horizon of $T = 5$. Fig. 5 shows the resulting trajectories for the states $[h_1, h_3]$. The hysteresis of the switching valve V_1 was removed for simplicity.

7.3 Fault Detection

For the tank system described in the last section we applied the fault detection scheme described in section 5 and in [6]. A switching controller for valve V_1 is used to keep the liquid level in the middle tank at some desired value. The following two types of faults are considered: The fault ϕ_1 denotes a leak in tank 1 and the fault ϕ_2 implies that valve V_1 is blocked closed. Note that the failure of valve V_1 is modeled as a multiplicative fault. In Fig. 6 we simulated the occurrence of the faults at different times.

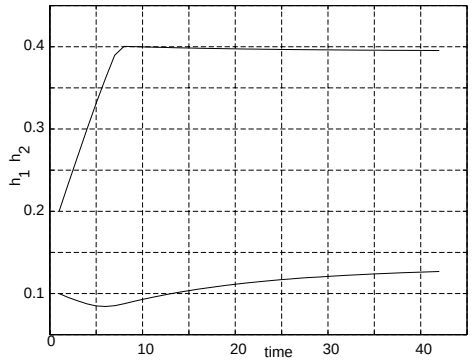

Bild 5: Closed-loop regulation problem for the control of the tank system

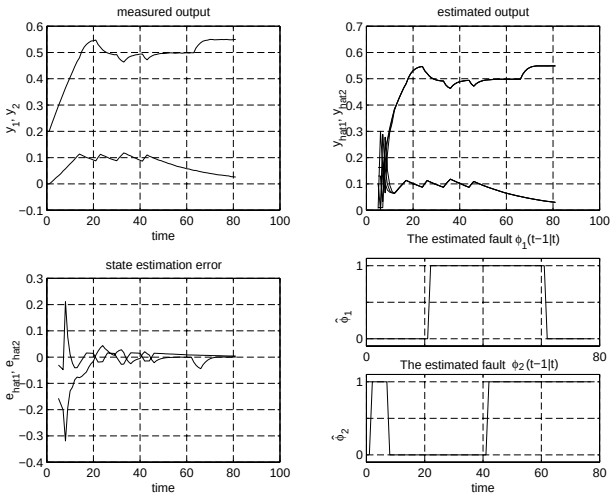

Bild 6: Simulation of a leak in tank 1 (ϕ_1) from $t = 20$ until $t = 60$, and a blocking valve (ϕ_2) from $t = 40$ until $t = 80$.

Both faults are detected correctly with a few time steps of delay. Note however that during the startup there are a few false alarms of fault ϕ_2, i.e. blocking of valve V_1. These wrongly detected faults are due to the fact, that the level in tank 1 has not yet reached the height of valve V_1. Therefore no liquid can pass through V_1, which is indistinguishable from a blocked valve V_1. To avoid this problem it is very natural to formulate the clause $[h_1 \leq h_v] \Rightarrow \phi_2 = 0$. This is just an additional constraint that can be added to the other constraints of the optimization problem. With this correction, the fault estimates are free of any errors.

8 Conclusions

Via the Mixed Logic Dynamical (MLD) System formulation we are capable of describing a wide range of practical control problems, involving, for example, linear hybrid systems; sequential logical systems (finite state machines, automata); nonlinear dynamical systems, where the nonlinearity can be expressed through

combinational logic; some classes of discrete event systems; constrained linear systems, etc. The solution of these problems can be posed in terms of Mixed-Integer Linear (or Quadratic) Programs (MILPs or MIQPs). Because efficient techniques not only for MILPs but also for MIQPs are becoming available, this new approach holds much promise for tackling realistic size problems.

Literatur

[1] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In A.P. Ravn R.L. Grossman, A. Nerode and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture notes in computer Science*, pages 209–229. Springer Verlag, 1993.

[2] R. Alur, T.A. Henzinger, and P.H. Ho. Automatic Symbolic Verification of Embedded Systems. *IEEE Trans. on Software Engineering*, 22:181–201, 1996.

[3] M. Anlauff, A. Bemporad, A. Chakraborty, P. Kutter, D. Mignone, M. Morari, A. Pierantonio, and L. Thiele. From Ease in Programming to Easy Maintenance: Extending DSL Usability with Montages. In *submitted to Usenix, DSL 99, 2nd conference on domain specific languages, Austin Texas*, 1999.

[4] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and Controllability of Piecewise Affine and Hybrid Systems. *submitted to CDC*, 1999.

[5] A. Bemporad, D. Mignone, and M. Morari. An Efficient Branch and Bound Algorithm for State Estimation and Control of Hybrid Systems. *European Control Conference*, 1999.

[6] A. Bemporad, D. Mignone, and M. Morari. Moving Horizon Estimation for Hybrid Systems and Fault Detection. In *Proceedings of the American Control Conference*, 1999.

[7] A. Bemporad and M. Morari. Control of Systems Integrating Logic, Dynamics, and Constraints. *Automatica*, 35(3):407–427, March 1999. `ftp://control.ethz.ch/pub/reports/postscript/AUT98-04.ps`.

[8] A. Bemporad and M. Morari. Verification of Hybrid Systems via Mathematical Programming. In *Hybrid Systems: Computation and Control; Second International Workshop*, 1999.

[9] L. Berec and L. Tesař. Testing fault detection methods via three-tank system. Technical Report Issue A, Copernicus Project CT94-0237, 1997.

[10] T.M. Cavalier, P.M. Pardalos, and A.L. Soyster. Modeling and integer programming techniques applied to propositional calculus. *Computers Opns Res.*, 17(6):561–570, 1990.

[11] G. Dolanc, D. Juricic, A. Rakar, J. Petrovcic, and D. Vrancic. Three-tank Benchmark Test. Technical Report Copernicus Project Report CT94-02337, Jozef Stefan Institute, 1997. `ftp://ftp.utia.cas.cz/pub/staff/tesar/fault/latest/`.

[12] N. Elia and B. Brandin. Verification of an automotive active leveler. In *Proceedings of the American Control Conference*, 1999.

[13] A. Fehnker. Automotive control revised - linear inequalities as approximation of reachable sets. In *Hybrid Systems: Computation and Control*, volume 1386 of *Lecture Notes in Computer Science*, pages 110–125. Springer Verlag, 1998.

[14] R. Fletcher and S. Leyffer. Numerical experience with lower bounds for MIQP branch-and-bound. Technical report, University of Dundee, Dept. of Mathematics, Scotland, U.K., 1995. submitted to SIAM Journal on Optimization, `http://www.mcs.dundee.ac.uk:8080/~sleyffer/miqp_art.ps.Z`.

[15] C.A. Floudas. *Nonlinear and Mixed-Integer Optimization.* Oxford University Press, 1995.

[16] C.E. Garcia, D.M. Prett, and M. Morari. Model Predictive Control: Theory and Practice – a Survey. *Automatica*, 25(3):335–348, 1989.

[17] M. Heymann, F. Lin, and G. Meyer. Viability of Controllers for Hybrid Machines. *Proceedings of the 36th Conference on Decision and Control*, 1997.

[18] ILOG. *Using the CPLEX Callable Library*, 5.0 edition, 1997.

[19] S.S. Keerthi and E.G. Gilbert. Optimal Infinite-Horizon Feedback Laws for a General Class of Constrained Discrete-Time Systems: Stability and Moving-Horizon Approximations. *Journal of Optimization Theory and Applications*, 57(2):265–293, 1988.

[20] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs: a class of decidable hybrid systems. In A.P. Ravn R.L. Grossman, A. Nerode and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture notes in computer Science*, pages 179–208. Springer Verlag, 1993.

[21] J. Lunze. Laboratory Three Tanks System — Benchmark for the Reconfiguration Problem. Technical report, Tech. Univ. of Hamburg-Harburg, Inst. of Control. Eng., Germany, 1998. `http://www.tu-harburg.de/rts/software/cosy/`.

[22] D. Mignone, A. Bemporad, and M. Morari. A Framework for Control, Fault Detection, State Estimation and Verification of Hybrid Systems. *Proceedings of the American Control Conference*, 1999.

[23] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.

[24] R. Raman and I.E. Grossmann. Integration of logic and heuristic knowledge in MINLP optimization for process synthesis. *Computers and Chemical Engineering*, 16(3):155–171, 1992.

[25] C. V. Rao and J. B. Rawlings. Nonlinear Moving Horizon Estimation. In *International Symposium on Nonlinear Model Predictive Control: Assessment and Future Directions (Ascona, Switzerland)*, 1998.

[26] H. S. Ryoo and N.V. Sahinidis. A Branch-and-Reduce Approach to Global Optimization. *Journal of Global Optimization*, 8:107–138, 1996.

[27] T. Stauner, O. Müller, and M. Fuchs. Using HYTECH to verify an automotive control system. In O. Maler, editor, *Hybrid and Real-Time Systems*, volume 1201 of *Lecture Notes in Computer Science*, pages 139–153. Springer Verlag, 1997.

[28] H.P. Williams. *Model Building in Mathematical Programming*. John Wiley & Sons, Third Edition, 1993.

Manuskripteingang: 4. Juni 1999