# Hybrid System Analysis and Control via Mixed Integer Optimization

Manfred Morari*
Automatic Control Laboratory,
ETH—Swiss Federal Institute of Technology,
CH-8092 Zurich, Switzerland

**Abstract**

The paper discusses a framework for modeling, analyzing and controlling systems whose behavior is governed by interdependent physical laws, logic rules, and operating constraints, denoted as *Mixed Logical Dynamical* (MLD) systems. They are described by linear dynamic equations subject to linear inequalities involving real and integer variables. MLD models are equivalent to various other system descriptions like Piece Wise Affine (PWA) systems and Linear Complementarity (LC) systems. They have the advantage, however, that all problems of system analysis (like controllability, observability, stability and verification) and all problems of synthesis (like controller design and filter design) can be readily expressed as mixed integer linear or quadratic programs, for which many commercial software packages exist.

In this paper we first recall how to derive MLD models and then illustrate their use in predictive control. Subsequently we define "verification" and show how verification algorithms can be used to solve a variety of practical problems like checking the correctness of an emergency shutdown procedure implemented on a PLC, or assessing the performance of a constrained MPC controller. The eventual practical success of these methods will depend on progress in the development of the various optimization packages so that problems of realistic size can be tackled.

**Keywords**

Hybrid systems, Predictive control, Dynamic models, Mixed-integer programming, Optimization problems

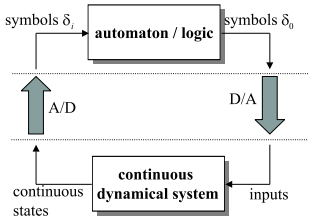
## Introduction

The concept of a *model* of a system is traditionally associated with differential or difference equations, typically derived from physical laws governing the *dynamics* of the system under consideration. Consequently, most of the control theory and tools have been developed for such systems, in particular for systems whose evolution is described by smooth linear or nonlinear state transition functions. On the other hand, in many applications the system to be controlled is also constituted by parts described by *logic*, such as for instance on/off switches or valves, gears or speed selectors, evolutions dependent on if-then-else rules. Often, the control of these systems is left to schemes based on heuristic rules inferred by practical plant operation.

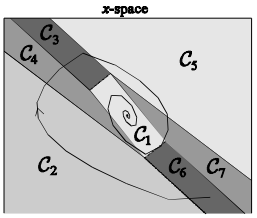
Recently, in the literature researchers started dealing with *hybrid systems*, namely hierarchical systems composed of dynamical components at the lower level, governed by upper level logical/discrete components (Grossmann et al., 1993; Branicky et al., 1998; Labinaz et al., 1997; Branicky, 1995). Hybrid systems arise in a large number of application areas, and are attracting increasing attention in both academic theory-oriented circles as well as in industry. Our interest 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 hybrid systems and to analyze their performance.

This paper discusses a framework for modeling, analyzing and controlling models of systems described by interacting physical laws, logical rules, and operating constraints. We will focus exclusively on discrete time models. We note, however, that interesting mathematical phenomena occurring in hybrid systems, such as Zeno behaviors (Johansson et al., 1999) do not exist in discrete-time. On the other hand, most of these phenomena are usually a consequence of the continuous-time switching model, rather than the real natural behavior. Our main motivation for concentrating on discrete-time models stems from the need to analyze these systems and to solve optimization problems, such as optimal control or scheduling problems, for which the continuous-time counterpart would not be easily computable.

Two main categories of hybrid systems were successfully adopted for analysis and synthesis purposes (Branicky, 1995): *hybrid control systems* (Alur et al., 1993;




(a) Hybrid control systems. Finite state machines and continuous dynamics interact through analog-to-digital (A/D) and D/A interfaces




(b) Piecewise affine systems

**Figure 1:** Hybrid models.

*phone +41-1-632 7626, fax +41-1-632 1211, email: morari@aut.ee.ethz.ch. This paper is based on the work of the Hybrid Systems Group at ETH currently including A. Bemporad, F. Borrelli, F. Cuzzola, G. Ferrari-Trecate, T. Geyer, D. Mignone and F. D. Torrisi

Asarin et al., 1995; Bemporad and Morari, 1999; Lygeros et al., 1996, 1999), which consist of the interaction between continuous dynamical systems and discrete/logic automata (Figure 1(a)), and *switched systems* (Branicky, 1998; Johansson and Rantzer, 1998; Sontag, 1981), where the state-space is partitioned into regions, each one being associated with a different continuous dynamics (Figure 1(b)).

Several modeling frameworks have been introduced for discrete-time hybrid systems. We will refer frequently to the *piecewise affine* (PWA) systems (Sontag, 1981, 1996).

$$\begin{aligned} x(t+1) &= A_i x(t) + B_i u(t) + f_i \ , \\ &\text{for } x(t) \in \mathcal{C}_i \triangleq \{x : \ H_i x \leq K_i\} \end{aligned} \quad (1)$$

where $x \in X \subseteq \mathbb{R}^n$, $u \in \mathbb{R}^m$, $\{\mathcal{C}_i\}_{i=0}^{s-1}$ is a polyhedral partition of the sets of states $X$ and $f_i$ is a constant vector. Furthermore, we mention here linear complementarity (LC) systems (Heemels, 1999; van der Schaft and Schumacher, 1998; Heemels et al., 2000) and extended linear complementarity (ELC) systems (De Schutter and De Moor, 1999), max-min-plus-scaling (MMPS) systems (De Schutter and van den Boom, 2000), and mixed logical dynamical (MLD) systems (Bemporad and Morari, 1999). Each modeling framework has its advantages. For instance, stability criteria were formulated for PWA systems (Johansson and Rantzer, 1998), and control and verification techniques were proposed for MLD discrete-time hybrid models (Bemporad and Morari, 1999; Bemporad et al., 2000). In particular, MLD models were proven successful for recasting hybrid dynamical optimization problems into mixed-integer linear and quadratic programs, solvable via branch and bound techniques (Nemhauser and Wolsey, 1988). Recently, the equivalence of PWA, LC, ELC, MMPS, and MLD hybrid dynamical systems was proven constructively in (Heemels et al., 2001; Bemporad et al., 2000a). Thus the theoretical properties and tools can be easily transferred from one class to another.

## Mixed Logical Dynamical (MLD) Systems

We briefly recall the *mixed logical dynamical* (MLD) system framework introduced in (Bemporad and Morari, 1999). MLD systems are hybrid (control) systems defined by the interaction of logic, finite state machines, and linear discrete-time systems, as shown in Figure 1(a). The ability to include constraints, constraint prioritization, and heuristics augment the expressiveness and generality of the MLD framework.

The MLD modeling framework relies on the idea of translating logic relations into mixed-integer linear inequalities (Bemporad and Morari, 1999; Cavalier et al., 1990; Raman and Grossmann, 1991; Tyler and Morari, 1999; Williams, 1977, 1993). By following standard notation, we adopt capital letters $X_i$ to represent statements, e.g. "$x \geq 0$" or "Temperature is hot". $X_i$ is commonly referred to as a *literal*, and has a *truth value* of either TRUE or FALSE. Boolean algebra enables statements to be combined in compound statements by means of *connectives*: "$\wedge$" (and), "$\vee$" (or), "$\sim$" (not), "$\rightarrow$" (implies), "$\leftrightarrow$" (if and only if), "$\oplus$" (exclusive or). Connectives satisfy several properties (see e.g. (Christiansen, 1997)), which can be used to transform compound statements into equivalent statements involving different connectives, and simplify complex statements. Correspondingly one can associate with a literal $X_i$ a *logical variable* $\delta_i \in \{0,1\}$, which has a value of either 1 if $X_i =$TRUE, or 0 otherwise. A propositional logic problem, where a statement $X_1$ must be proved to be true given a set of (compound) statements involving literals $X_1, \ldots, X_n$, can be solved by means of a linear integer program by suitably translating the original compound statements into linear inequalities involving logical variables $\delta_i$. In fact, the propositions and linear constraints reported in Table 1 can easily be seen to be equivalent.

These translation techniques can be adopted to model logical parts of processes and heuristic knowledge about plant operation as integer linear inequalities. The link between logic statements and continuous dynamical variables, in the form of logic statements derived from conditions on physical dynamics, is provided by properties (AD1), (DA1) in Table 1, and leads to *mixed-integer linear inequalities*, i.e., linear inequalities involving both *continuous variables* of $\mathbb{R}^n$ and logical (*indicator*) variables in $\{0,1\}$. Consider for instance the statement $X \triangleq [f_1(x) \leq 0]$ where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a linear (affine) function, assume that $x \in \mathcal{X}$, where $\mathcal{X} \subset \mathbb{R}^n$ is a given bounded set, and define

$$M_i \triangleq \max_{x \in \mathcal{X}} f_i(x), \ m_i \triangleq \min_{x \in \mathcal{X}} f_i(x), \ i = 1,2$$

Theoretically, an over[under]-estimate of $M_i$ $[m_i]$ suffices for our purpose. By associating a binary variable $\delta$ with the literal $X$, one can transform $X \triangleq [f_1(x) \leq 0]$ into mixed integer inequalities as described in (AD1), Table 1, where $\epsilon$ is a small tolerance (typically the machine precision), beyond which the constraint is regarded as violated. Note that sometimes translations require the introduction of *auxiliary variables* (Williams, 1993, p. 178), for instance according to (DA1) a quantity which is either $f_1(x)$ is $X$ is true, or $f_2(x)$, requires the introduction of a real variable $z$.

The rules of Table 1 can be generalized for relations involving an arbitrary number of discrete variables combined by arbitrary connectives. *Any* combinational relation of logical variables can in fact be translated to the conjunctive normal form (CNF)

$$P_1 \wedge P_2 \wedge \ldots \wedge P_m, \ P_i = Y_1^i \vee Y_2^i \vee \ldots \vee Y_{n_i}^i$$

where $Y_j^i$ is a literal $X$ or its inverse $\sim X$, by using standard methods (Bemporad et al., 2000b). As an example,

| | Relation | Logic | (In)equalities |
|---|---|---|---|
| **L1** | **AND** ($\wedge$) | $X_1 \wedge X_2$ | $\delta_1 = 1$, $\delta_2 = 1$ |
| **L2** | **OR** ($\vee$) | $X_1 \vee X_2$ | $\delta_1 + \delta_2 \geq 1$ |
| **L3** | **NOT** ($\sim$) | $\sim X_1$ | $\delta_1 = 0$ |
| **L4** | **XOR** ($\oplus$) | $X_1 \oplus X_2$ | $\delta_1 + \delta_2 = 1$ |
| **L5** | **IMPLY** ($\rightarrow$) | $X_1 \rightarrow X_2$ | $\delta_1 - \delta_2 \leq 0$ |
| **L6** | **IFF** ($\leftrightarrow$) | $X_1 \leftrightarrow X_2$ | $\delta_1 - \delta_2 = 0$ |
| **L7** | **ASSIGNMENT** ($=, \leftrightarrow$) | $X_3 = X_1 \wedge X_2$<br><br>$X_3 \leftrightarrow X_1 \wedge X_2$ | $\delta_1 + (1 - \delta_3) \geq 1$<br>$\delta_2 + (1 - \delta_3) \geq 1$<br>$(1 - \delta_1) + (1 - \delta_2) + \delta_3 \geq 1$ |
| **AD1** | **EVENT** | $[f(x) \leq 0] \leftrightarrow [\delta = 1]$ | $f(x) \leq M - M\delta$<br>$f(x) \geq \epsilon + (m - \epsilon)\delta$ |
| **DA1** | **IF-THEN-ELSE** (**=Product**) | **IF** $X$ **THEN** $z = f_1(x)$ **ELSE** $z = f_2(x)$ ($z = \delta f_1(x) + (1 - \delta) f_2(x)$) | $(m_2 - M_1)\delta + z \leq f_2(x)$<br>$(m_1 - M_2)\delta - z \leq -f_2(x)$<br>$(m_1 - M_2)(1 - \delta) + z \leq f_1(x)$<br>$(m_2 - M_1)(1 - \delta) - z \leq -f_1(x)$ |

**Table 1:** Basic conversion of logic relations into mixed-integer inequalities. Relations involving the inverted literals $\sim X$ can be obtained by substituting $(1 - \delta)$ for $\delta$ in the corresponding inequalities.

the statement (L7) $X_3 \leftrightarrow X_1 \wedge X_2$ is equivalent to

$$(\sim X_1 \vee \sim X_2 \vee X_3) \wedge (X_1 \vee \sim X_3) \wedge (X_2 \vee \sim X_3) \quad (2)$$

Subsequently, the CNF form can be translated (again, automatically and without introducing additional integer variables) into the set of integer linear inequalities

$$y_1^i + y_2^i + \ldots + y_{n_i}^i \geq 1, \ i = 1, \ldots, m,$$

where $y_j^i = \delta_j^i$ if $Y_j^i = X$ or $(1 - \delta_j^i)$ if $Y_j^i = \sim X$. For instance, it is immediate to check that the CNF (2) maps to the inequalities reported in Table 1(L7).

The state update law of finite state machines can be described by logic statements involving binary states, their time updates, and indicator variables. It is clear that the automatized translation mentioned above can be directly applied to translate automata into a set of linear integer inequalities. An example will be provided when modeling the PLC control code of the batch evaporator process benchmark.

By collecting the equalities and inequalities generated by the translation of the automata, analog-to-digital interface (AD1), digital-to-analog interface (DA2), and by including the linear dynamic difference equations, we can model the hybrid system depicted in Figure 1(a) as the Mixed Logical Dynamical (MLD) system

$$x(t+1) = \Phi x(t) + \mathcal{G}_1 u(t) + \mathcal{G}_2 \delta(t) + \mathcal{G}_3 z(t) \quad (3a)$$
$$y(t) = \mathcal{H} x(t) + \mathcal{D}_1 u(t) + \mathcal{D}_2 \delta(t) + \mathcal{D}_3 z(t) \quad (3b)$$
$$\mathcal{E}_2 \delta(t) + \mathcal{E}_3 z(t) \leq \mathcal{E}_1 u(t) + \mathcal{E}_4 x(t) + \mathcal{E}_5 \quad (3c)$$

where $x \in \mathbb{R}^{n_c} \times \{0,1\}^{n_\ell}$ is a vector of continuous and binary states, $u \in \mathbb{R}^{m_c} \times \{0,1\}^{m_\ell}$ are the inputs, $y \in \mathbb{R}^{p_c} \times \{0,1\}^{p_\ell}$ the outputs, $\delta \in \{0,1\}^{r_\ell}$, $z \in \mathbb{R}^{r_c}$ represent auxiliary binary and continuous variables respectively, which are introduced when transforming logic relations into mixed-integer linear inequalities, and $\Phi$, $\mathcal{G}_1$, $\mathcal{G}_2$, $\mathcal{G}_3$, $\mathcal{H}$, $\mathcal{E}_1, \ldots, \mathcal{E}_5$ are matrices of suitable dimensions. The inequalities (3c) include the constraints obtained by the D/A, A/D, and automaton part of the system, as well as possible physical constraints on states and inputs. Despite the fact that the description (3) seems to be linear, clearly the nonlinearity is concentrated in the integrality constraints over binary variables. The following simple example illustrates the technique.

**Example 1**

Consider the following switched system

$$x(t+1) = \begin{cases} 0.8x(t) + u(t) & \text{if} \quad x(t) \geq 0 \\ -0.8x(t) + u(t) & \text{if} \quad x(t) < 0 \end{cases} \quad (4)$$

where $x(t) \in [-10, 10]$, and $u(t) \in [-1, 1]$. The condition $x(t) \geq 0$ can be associated with a binary variable $\delta(t)$ such that

$$[\delta(t) = 1] \ \leftrightarrow \ [x(t) \geq 0] \quad (5)$$

By using the transformation (AD1) in Table 1, Equation 5 can be expressed by the inequalities

$$-m\delta(t) \leq x(t) - m$$
$$-(M + \epsilon)\delta \leq -x - \epsilon$$

where $M = -m = 10$, and $\epsilon$ is a small positive scalar. Then (4) can be rewritten as

$$x(t+1) = 1.6\delta(t)x(t) - 0.8x(t) + u(t) \quad (6)$$

By defining a new variable $z(t) = \delta(t)x(t)$ which, by (DA1) in Table 1, can be expressed as

$$\begin{aligned} z(t) &\leq M\delta(t) \\ z(t) &\geq m\delta(t) \\ z(t) &\leq x(t) - m(1-\delta(t)) \\ z(t) &\geq x(t) - M(1-\delta(t)), \end{aligned}$$

the evolution of system (4) is ruled by the linear equation

$$x(t+1) = 1.6z(t) - 0.8x(t) + u(t)$$

subject to the linear constraints above.

We assume that system (3) is *completely well-posed* (Bemporad and Morari, 1999), which means that for all $x, u$ within a bounded set the variables $\delta, z$ are uniquely determined, i.e., there exist functions $F$, $G$ such that, at each time $t$, $\delta(t) = F(x(t), u(t))$, $z(t) = G(x(t), u(t))$[1]. Therefore the $x$- and $y$-trajectories exist and are uniquely determined by the initial state $x(0)$ and input signal $u$. In light of the transformations of Table 1, it is clear that the well-posedness assumption stated above is usually guaranteed by the procedure used to generate the linear inequalities (3c), and therefore this hypothesis is typically verified by MLD relations derived from modeling real-world plants. Nevertheless, a numerical test for well-posedness is reported in (Bemporad and Morari, 1999, Appendix 1).

Recently, in (Bemporad et al., 2000b) a declarative language for specifying hybrid control systems has been developed which fully automatizes the construction of the MLD matrices. Such a language (called HYSDEL, HYbrid System DEscription Language) will be used later to model the batch evaporator process benchmark. Throughout the paper, we will assume that both the PWA and the MLD forms are available. Their role is complementary and in our algorithms we use whichever one is more appropriate for a specific task.

## Predictive Control of MLD Systems

The predictive control problem for MLD sytems can be defined formally as follows. Consider an equilibrium pair $(x_e, u_e)$ Let the components $\delta_{e,i}$, $z_{e,j}$ correspond to desired steady-state values for the indefinite auxiliary variables. Let $t$ be the current time, and $x(t)$ the current state. Consider the following optimal control problem

$$\begin{aligned} \min_{\{v_0^{T-1}\}} \; & J(v_0^{T-1}, x(t)) \triangleq \sum_{k=0}^{T-1} \|v(k) - u_e\|_{Q_1}^2 \\ & + \|\delta(k|t) - \delta_e\|_{Q_2}^2 + \|z(k|t) - z_e\|_{Q_3}^2 \\ & + \|x(k|t) - x_e\|_{Q_4}^2 + \|y(k|t) - y_e\|_{Q_5}^2 \end{aligned} \quad (7)$$

subject to

$$\begin{cases} x(T|t) & = x_e \\ x(k+1|t) & = Ax(k|t) + B_1 v(k) + B_2\delta(k|t) \\ & \quad + B_3 z(k|t) \\ y(k|t) & = Cx(k|t) + D_1 v(k) + D_2\delta(k|t) \\ & \quad + D_3 z(k|t) \\ E_2\delta(k|t) & + E_3 z(k|t) \leq E_1 v(k) + E_4 x(k|t) + E_5 \end{cases} \quad (8)$$

where $Q_1 = Q_1' > 0$, $Q_2 = Q_2' \geq 0$, $Q_3 = Q_3' \geq 0$, $Q_4 = Q_4' > 0$, $Q_5 = Q_5' \geq 0$, $x(k|t) \triangleq x(t+k, x(t), v_0^{k-1})$, and $\delta(k|t)$, $z(k|t)$, $y(k|t)$ are similarly defined. Assume for the moment that the optimal solution $\{v_t^*(k)\}_{k=0,\ldots,T-1}$ exists. According to the *receding horizon* philosophy, set

$$u(t) = v_t^*(0), \quad (9)$$

disregard the subsequent optimal inputs $v_t^*(1), \ldots, v_t^*(T-1)$, and repeat the whole optimization procedure at time $t+1$. The control law (7)–(9) will be referred to as the *Mixed Integer Predictive Control* (MIPC) law.

In principle all the design rules about parameter choices and theoretical results regarding stability developed for MPC over the last two decades can be applied here to MIPC after some adjustments. For instance, the number of control degrees of freedom can be reduced to $N_u < T$, by setting $u(k) \equiv u(N_u - 1)$, $\forall k = N_u, \ldots, T$. While this choice usually reduces the size of the optimization problem dramatically at the price of inferior performance, here the computational gain is only partial, since all the $T$ $\delta(k|t)$ and $z(k|t)$ variables remain in the optimization.

Infinite horizon formulations are inappropriate for both practical and theoretical reasons. In fact, approximating the infinite horizon with a large $T$ is computationally prohibitive, as the number of 0-1 variables involved in the MIQP depends linearly on $T$. Moreover, the quadratic term in $\delta$ might oscillate (for example, for a system which approaches the origin in a "switching" manner), and hence "good" (i.e., asymptotically stabilizing) input sequences might be ruled out by a corresponding infinite value of the performance index; it could even happen that no input sequence has finite cost.

Using an appropriate stability definition (Bemporad and Morari, 1999) have proven asymptotic stability of the MIPC scheme. In the typical fashion, the end point constraint was invoked in the Lyapunov arguments.

Despite the fact that very effective methods exist to compute the (global) optimal solution of the MIQP problem (7)–(9) (see Section  below), in the worst-case the solution time depends exponentially on the number of integer variables. In principle, this might limit the scope of application of the proposed method to very slow systems, since for real-time implementation the sampling time should be large enough to allow the worst-case computation. However, the stability proof does not require that the evaluated control sequences $\{\mathcal{U}_t^*\}_{t=0}^{\infty}$ are global

[1] A more general definition of well-posedness, where only the components of $\delta$ and $z$ entering (3a)–(3b) are required to be unique, is given in (Bemporad and Morari, 1999).

optima, but only that they lead to a decrease in the objective function. Thus the solver can be interrupted at any intermediate step to obtain a suboptimal solution $\mathcal{U}_{t+1}^*$ which satisfies the decrease condition. For instance, when Branch & Bound methods are used to solve the MIQP problem, the new control sequence $\mathcal{U}_t^*$ can be selected as the solution to a QP subproblem which is integer-feasible and has the lowest value. Obviously in this case the performance deteriorates.

**Example 2**

Consider the following system

$$\begin{cases} x(t+1) = 0.8 \begin{bmatrix} \cos\alpha(t) & -\sin\alpha(t) \\ \sin\alpha(t) & \cos\alpha(t) \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \\ y(t) = [1\ 0]x(t) \\ \alpha(t) = \begin{cases} \frac{\pi}{3} & \text{if} \quad [1\ 0]x(t) \geq 0 \\ -\frac{\pi}{3} & \text{if} \quad [1\ 0]x(t) < 0 \end{cases} \\ x(t) \in [-10, 10] \times [-10, 10] \\ u(t) \in [-1, 1] \end{cases} \tag{10}$$

By using auxiliary variables $z(t) \in \mathbb{R}^4$ and $\delta(t) \in \{0, 1\}$ such that $[\delta(t) = 1] \leftrightarrow [[1\ 0]x(t) \geq 0]$, Equation 10 can be rewritten in the form (3) as

$$\begin{aligned} x(t+1) = &\begin{bmatrix} I & I \end{bmatrix} z(t) \\ &\begin{bmatrix} 10 \\ -10-\epsilon \\ -M \\ -M \\ M \\ M \\ M \\ M \\ -M \\ -M \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \delta(t) + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ I & 0 \\ -I & 0 \\ 0 & I \\ 0 & -I \\ I & 0 \\ -I & 0 \\ 0 & I \\ 0 & -I \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} z(t) \leq \\ &\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ B \\ -B \\ B \\ -B \\ 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} u(t) + \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & \\ 0 & \\ 0 & \\ 0 & \\ A_1 & \\ -A_1 & \\ A_2 & \\ -A_2 & \\ I & \\ -I & \\ 0 & \\ 0 & \end{bmatrix} x(t) + \begin{bmatrix} 10 \\ -\epsilon \\ 0 \\ 0 \\ M \\ M \\ M \\ M \\ 0 \\ 0 \\ N \\ N \\ 1 \\ 1 \end{bmatrix} \end{aligned}$$

where $B = [0\ 1]'$, $A_1, A_2$ are obtained by (10) by setting respectively $\alpha = \frac{\pi}{3}, -\frac{\pi}{3}$, $M = 4(1+\sqrt{3})[1\ 1]' + B$, $N \triangleq 10[1\ 1]'$, and $\epsilon$ is a properly small positive scalar.

In order to stabilize the system to the origin, the feedback control law (7)–(9) is adopted, along with the parameters $T = 3$, $u_e = 0$, $\delta_e = 0$, $z_e = [0\ 0\ 0\ 0]'$,

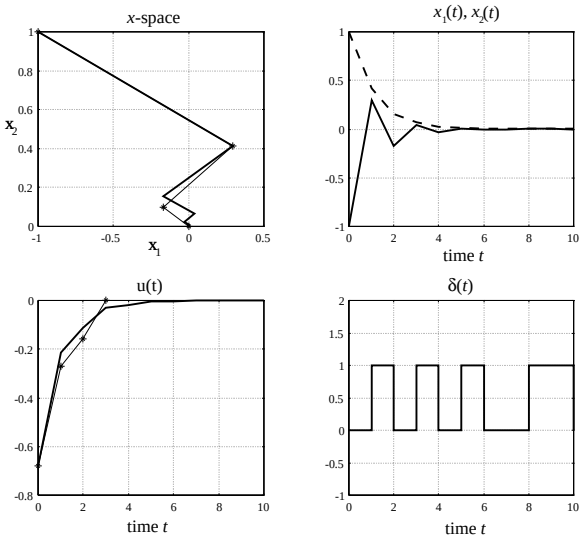



**Figure 2:** Closed-loop regulation problem for system (10). Closed-loop trajectories (thick lines) and optimal solution at $t = 0$ (thin lines, right plots).

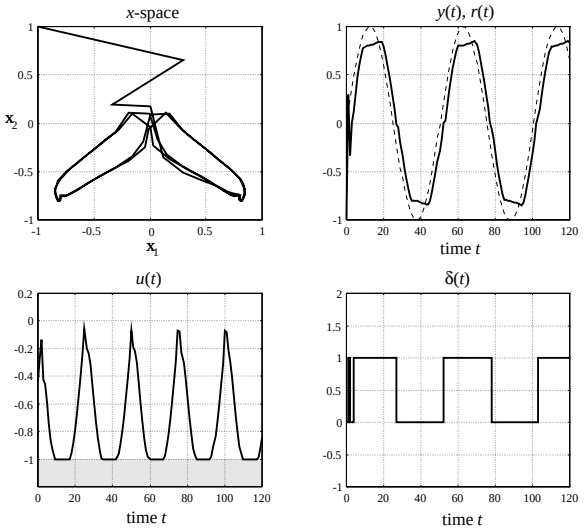



**Figure 3:** Closed-loop tracking problem for system (10), with $y(t) = x_1(t)$.

$x_e = [0\ 0]'$, $y_e = 0$, and the weights $Q_1 = 1$, $Q_2 = 0.01$, $Q_3 = 0.01 I_4$, $Q_4 = I_2$, $Q_5 = 0$. Figure 2 shows the resulting trajectories. The trajectories obtained at time $t = 0$ by solving the optimal control problem (7)–(8) are also reported in the right plots (thin lines). Consider now a desired reference $r(t) = \sin(t/8)$ for the output $y(t)$. We apply the same MIPC controller, with the exception of $Q_4 = 10^{-8} I_2$, $Q_5 = 1$. The steady-state parameters are selected as $y_e = r(t)$, and $u_e$, $x_e$, $\delta_e$, $z_e$ consistently. Figure 3 shows the resulting closed-loop trajectories. Notice that the constraint $-1 \leq u(t) \leq 1$ prevents the system from tracking the peaks of the sinusoid, and therefore

the output trajectory is chopped. □

**MIQP Solvers**

With the exception of particular structures, mixed-integer programming problems involving 0-1 variables are classified as $\mathcal{NP}$-complete, which means that in the worst case, the solution time grows exponentially with the problem size (Raman and Grossmann, 1991). Despite this combinatorial nature, several algorithmic approaches have been proposed and applied successfully to medium and large size application problems (Floudas, 1995), the four major ones being

- *Cutting plane methods*, where new constraints (or "cuts") are generated and added to reduce the feasible domain until a 0-1 optimal solution is found.
- *Decomposition methods*, where the mathematical structure of the models is exploited via variable partitioning, duality, and relaxation methods.
- *Logic-based methods*, where disjunctive constraints or symbolic inference techniques are utilized which can be expressed in terms of binary variables.
- *Branch and bound methods*, where the 0-1 combinations are explored through a binary tree, the feasible region is partitioned into sub-domains systematically, and valid upper and lower bounds are generated at different levels of the binary tree.

For MIQP problems, (Fletcher and Leyffer, 1998) indicate Generalized Benders' Decomposition (GBD) (Lazimy, 1985), Outer Approximation (OA), LP/QP based branch and bound, and Branch and Bound as the major solvers. See (Roschchin et al., 1987) for a review of these methods.

Several authors agree on the fact that branch and bound methods are the most successful for mixed integer quadratic programs. (Fletcher and Leyffer, 1998) report a numerical study which compares different approaches, and Branch and Bound is shown to be superior by an order of magnitude. While OA and GBD techniques can be attractive for general Mixed-Integer Nonlinear Problems (MINLP), for MIQP at each node the relaxed QP problem can be solved without approximations and reasonably quickly (for instance, the Hessian matrix of each relaxed QP is constant).

As described by (Fletcher and Leyffer, 1998), the Branch and Bound algorithm for MIQP consists of solving and generating new QP problems in accordance with a tree search, where the nodes of the tree correspond to QP subproblems. Branching is obtained by generating child-nodes from parent-nodes according to branching rules, which can be based for instance on a-priori specified priorities on integer variables, or on the amount by which the integer constraints are violated. Nodes are labeled as either pending, if the corresponding QP problem has not yet been solved, or fathomed, if the node has already been fully explored. The algorithm stops when all nodes have been fathomed. The success of the branch and bound algorithm relies on the fact that whole subtrees can be excluded from further exploration by fathoming the corresponding root nodes. This happens if the corresponding QP subproblem is either infeasible or an integer solution is obtained. In the second case, the corresponding value of the cost function serves as an upper bound on the optimal solution of the MIQP problem, and is used to further fathom other nodes having greater optimal value or lower bound.

Some of the simulation results reported in this paper have been obtained in Matlab by using the commercial Fortran package (Fletcher and Leyffer, 1994) as a MIQP solver. This package can handle both dense and sparse MIQP problems. The latter has proven to be particularly effective to solve most of the optimal control problems for MLD systems. In fact, the constraints have a triangular structure, and in addition most of the constraints generated by representation of logic facts involve only a few variables, which often leads to sparse matrices.

**Explicit Computation of MPC Control Law**

In (Bemporad et al., 2002) the authors presented a new approach to implement MPC, where all the computation effort is moved off-line. The idea is based on the observation that in (8) the state $x(0|t)$ can be considered a vector of parameters, and (7)–(9) as a multi-parametric Mixed Integer Quadratic Program (mp-MIQP). If the 1-norm is used in (7) instead of the 2-norm a multi-parametric Mixed Integer Linear Program (mp-MILP) results. An algorithm to solve mp-MILP problems was presented in (Acevedo and Pistikopoulos, 1997). Once the multi-parametric problem (7,8) has been solved off line, i.e., the solution $U_t^* = f(x(t))$ of (7,8) has been found, the model predictive controller is available explicitly. In (Acevedo and Pistikopoulos, 1997) the authors also show that the solution $U^* = f(x)$ of the mp-MILP problem is piecewise affine. Clearly, the same properties are inherited by the controller, i.e.,

$$\begin{array}{rl} u(t) = & F_i x(t) + g_i, \text{ for} \\ & x(t) \in \mathcal{C}_i \triangleq \{x:\ H_i x \le S_i\},\ i = 1, \ldots, s \end{array} \tag{11}$$

where $\cup_{i=1}^{s} \mathcal{C}_i$ is the set of states for which a feasible solution to (7,8) exists. Therefore, the closed MPC loop is of the form (1), where $A_i = A + BF_i$, $f_i = Bg_i$, $B_i = 0$. Note that the form of the closed-loop MPC system remains PWA also when (i) the matrices $A$, $B$ of the plant model are different from those used in the prediction model, and (ii) the plant model has a PWA form. Typically, the MPC law is designed based on a linear model obtained by linearizing the nonlinear model of the plant around some operating condition. When the nonlinear model can be approximated by a PWA system (e.g., through multiple linearizations at different

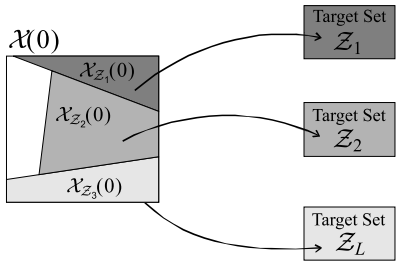



**Figure 4:** Reachability analysis problem.

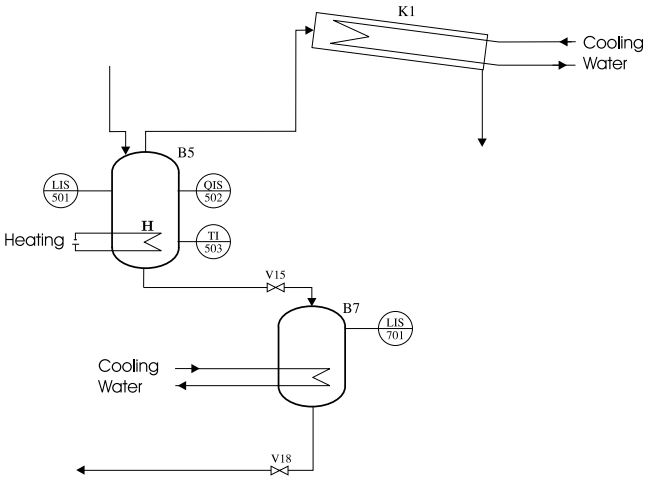



**Figure 5:** Flowchart of the benchmark evaporator system.

operating points or by approximating nonlinear static mappings by piecewise linear functions), the closed-loop consisting of the nonlinear plant model and the MPC controller can be approximated by a PWA system as well.

The explicit representation of the MPC controller discussed above is significant for several reasons. First of all, it gives some insight into the mathematical structure of the controller which is otherwise hidden behind the optimization formalism. Furthermore it offers an alternative route to an efficient controller implementation, opening up the route to use MPC in "fast" and "cheap" systems where the on-line solution of a QP and especially an MILP is prohibitive. Finally, the fact that we can represent the closed loop system in a PWA form allows us to apply new tools for performance analysis as discussed below.

## Verification of MLD and PWA Systems

In this section we will briefly review the topic of verification and sketch the available algorithms. There are numerous practical applications of verification, two of which (checking the correctness of an emergency shut down control system and performance analysis of MPC) will be discussed in some detail.

The problem of verification of hybrid systems, or, in system theoretical words, the *reachability analysis* of hybrid systems can be defined as follows:

**Reachability Analysis Problem**. Given a hybrid system $\Sigma$ (either in PWA form (1) or MLD (3)), a set of initial conditions $\mathcal{X}(0)$, a collection of disjoint target sets $\mathcal{Z}_1$, $\mathcal{Z}_2$, ..., $\mathcal{Z}_L$, a bounded set of inputs $\mathcal{U}$, and a time horizon $t \leq T_{\max}$, determine (*i*) if $\mathcal{Z}_j$ is reachable from $\mathcal{X}(0)$ within $t \leq T_{\max}$ steps for some input sequence $\{u(0), \ldots, u(t-1)\} \subseteq \mathcal{U}$; (*ii*) if yes, the subset of initial conditions $\mathcal{X}_{\mathcal{Z}_j}(0)$ of $\mathcal{X}(0)$ from which $\mathcal{Z}_j$ can be reached within $T_{\max}$ steps; (*iii*) for any $x_1 \in \mathcal{X}_{\mathcal{Z}_j}(0)$ and $x_2 \in \mathcal{Z}_j$, the input sequence $\{u(0), \ldots, u(t-1)\} \subseteq \mathcal{U}$, $t \leq T_{\max}$, which drives $x_1$ to $x_2$.

We will denote by $\mathcal{X}(t, \mathcal{X}(0))$ the *reach set* at time $t$ starting from any $x \in \mathcal{X}(0)$ and by applying any input $u(k) \in \mathcal{U}$, $k \leq t-1$.

Although finite time reachability analysis cannot answer certain "liveness" questions (for instance, if $\mathcal{Z}_i$ will be ever reached), the reachability problem stated above is decidable. The reason for focusing on finite-time reachability is that the time-horizon $T_{\max}$ has a clear meaning, namely that states which are reachable in more than $T_{\max}$ steps are in practice unreachable. Many undecidable problems can be approximated by decidable ones which are equivalent from a practical point of view. The decidable algorithm shown in (Bemporad et al., 2000a) for observability analysis, and the decidable stability analysis proposed in (Bemporad et al., 2000) are other examples of such a philosophy. Nevertheless, the problem is $\mathcal{N}P$-hard.

### Verification

**Algorithm.** Assume $\mathcal{X}(0) \subset C_i$ is a convex polyhedral set. With the algorithm introduced in (Bemporad et al., 2000) computing the evolution $\mathcal{X}(T_{\max}, \mathcal{X}(0))$ requires: (*i*) the reach set $\mathcal{X}(t, \mathcal{X}(0)) \bigcap \mathcal{C}_i$, i.e., the set of evolutions at time $t$ in $\mathcal{C}_i$ from $\mathcal{X}(0)$; (*ii*) crossing detection of the guardlines $\mathcal{P}_h \triangleq \mathcal{X}(t, \mathcal{X}(0)) \bigcap \mathcal{C}_h \neq \emptyset$, $\forall h = 0, \ldots, i-1, i+1, \ldots, s-1$; (*iii*) elimination of redundant constraints and approximation of the polyhedral representation of the new regions $\mathcal{P}_h$ (approximation is desirable, as the number of facets of $\mathcal{P}_h$ can grow linearly with time); (*iv*) detection of emptiness of $\mathcal{X}(t, \mathcal{P}_h)$ (emptiness happens when all the evolutions have crossed the guardlines) and detection of $\mathcal{X}(t, \mathcal{P}_h) \subseteq \mathcal{Z}_j$, $j = 1, \ldots, L$ (these will be referred to as *fathoming* conditions), (*v*) detection of $\mathcal{X}(t, \mathcal{P}_h) \cap \mathcal{Z}_j \neq \emptyset$, $j = 1, \ldots, L$ (reachability detection).

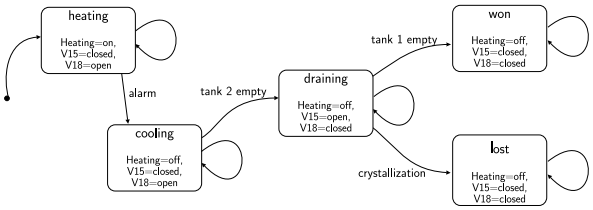



**Figure 6:** Model of the controller as finite state machine.

**Batch Evaporator Process Benchmark.** In this section we report on an application of the verification algorithm outlined in the previous section to the benchmark problem proposed within the ESPRIT-LTR Project 26270 VHS (Verification of Hybrid Systems)[2] as case study 1. The full system consists of an experimental batch plant (Kowalewski, 1998). In this paper we will focus only on the *evaporator system*, as proposed in (Kowalewski and Stursberg, 1998), which is schematically depicted in Figure 5. The considered subsystem consists of three parts: the upper tank 1 (labeled as B5 in Figure 5), the lower tank 2 (B7) and the condenser (K1). Tank 1 is equipped with a heating system (H), and is connected to tank 2 by a pipe and a valve (V15), while the outlet of tank 2 is controlled by valve V18. Both the heating system and the valves can only have two configurations: on (open) and off (closed). The levels $h_1$, $h_2$ of the solution in the two tanks, and the temperature $T$ of tank 1 are detected by sensors. These provide four logic signals: tank 1 empty, tank 2 empty, alarm, and crystallization. A tank is considered as empty when its level is lower than 0.01 m.

During normal operation of the plant, an aqueous solution of salt enters tank 1 to be concentrated. The exiting steam flows through a condenser. When the concentration of salt has reached a certain level, the heating system is switched off, valve V15 is opened, and the solution flows to tank 2 for post-processing operations. The plant is designed in such a way that more than one batch can be produced at the same time, so that tank 1 and tank 2 can process different batches in parallel.

Here we want to analyze the exception handling needed when the condenser does not work properly. Suppose that for some reason (e.g. lack of cooling agent) the condenser malfunctions. In this case, the steam cannot be cooled down and the pressure rises in tank 1. The heating system should be switched off to prevent damages to the plant due to over-pressure. On the other hand, the temperature in tank 1 should not get lower than a critical temperature $T_c$, otherwise the salt may crystallize and expensive procedures would be needed to restore the original functionality.

The plant is controlled by a PLC (*programmable logic controller*). The finite state machine underlying the control code is described in Figure 6, where the event alarm occurs when $T \geq T_a$ and crystallization when $T \leq T_c$.

The control strategy can be explained as follows. When a malfunctioning of the condenser is detected, the controller enters the alarm mode and immediately opens valve V18 to empty tank 2. During this phase, the heating is still on (heating state). When the temperature in tank 1 reaches the alarm level $T_a$ (alarm), the heating is switched off and the controller enters the state cooling. Finally, when tank 2 is empty, the controller gets in the state draining, where valve V18 is closed and valve V15 is opened, and the solution flows from the upper tank to the lower one. From draining, the controller can either switch to the state won if tank 1 gets empty, or to lost if the temperature in tank 1 gets lower than the critical value $T_c$.

The goal is to verify that the controller satisfies the following safety requirements: (i) if a malfunctioning in the cooling system of the condenser occurs, the heater must be turned off quickly enough to prevent damages to the condenser, (ii) the solution in tank 1 is drained to tank 2 before it eventually solidifies, (iii) when the valve 15 is open tank 2 is empty.

Certifying that the PLC code satisfies these specifications amounts to verify that from all the initial states in a given set $\mathcal{X}_0$ the system never reaches the state lost, or, equivalently, that the system always reaches the state won

**Modeling the Evaporator Benchmark in MLD Form.** In order to use the verification tools outlined above, we need to obtain a hybrid model of the batch evaporator process in MLD form. We consider the model described in (Stursberg, 1999), which only takes into account the heights $h_1$, $h_2$ and the temperature $T$ of tank 1. The model is summarized in Table 2 (dynamic equations associated with each logical state), and is based on the following simplifying assumptions: (i) the pressure increase during the evaporation in the heating phase is neglected, (ii) the dynamics during the cooling phase is the same for $T \geqslant 373\ K$, (iii) average constants replace ranges of physical parameters.

After the piece-wise linear approximation of the square root relation (three sections) the model can be readily expressed in the HYSDEL language (see Appendix ). The MLD model generated by the compiler includes three continuous states $x_c$, three logic states $x_\ell$, 19 Boolean inputs $\delta$ and eight auxiliary variables $z$.

**Verification Results.** We aim at verifying that after an exception occurs, the PLC code based on the control logic of Figure 6 safely shuts down the plant to the won state for any initial condition $x(0) = \left[\begin{smallmatrix} x_c(0) \\ x_\ell(0) \end{smallmatrix}\right] \in \mathcal{X}_0 = \{T, h_1, h_2, x_\ell : T = 373, 0.2 \leq h_1 \leq 0.22, 0.28 \leq h_2 \leq 0.3, x_\ell = \left[\begin{smallmatrix} 0 \\ 0 \\ 0 \end{smallmatrix}\right]\}$.

[2] http://www-verimag.imag.fr/VHS/

| Logic state | Heating | V15 | V18 | Dynamic behavior |
|---|---|---|---|---|
| Heating | on | closed | open | $\begin{cases} \frac{\partial T}{\partial t} = k_3(q - k_4(T - t_e)) \\ \frac{\partial h_1}{\partial t} = 0 \\ \frac{\partial h_2}{\partial t} = -k2\sqrt{h_2} \end{cases}$ |
| Cooling | off | closed | open | $\begin{cases} \frac{\partial T}{\partial t} = k_5(T - t_e) \\ \frac{\partial h_1}{\partial t} = 0 \\ \frac{\partial h_2}{\partial t} = -k2\sqrt{h_2} \end{cases}$ |
| Draining | off | open | closed | $\begin{cases} \frac{\partial T}{\partial t} = k_5(T - t_e) \\ \frac{\partial h_1}{\partial t} = -k1\sqrt{h_1} \\ \frac{\partial h_2}{\partial t} = k2\sqrt{h_1} \end{cases}$ |
| Won | off | closed | closed | $\begin{cases} \frac{\partial T}{\partial t} = k_5(T - t_e) \\ \frac{\partial h_1}{\partial t} = 0 \\ \frac{\partial h_2}{\partial t} = 0 \end{cases}$ |
| Lost | off | closed | closed | $\begin{cases} \frac{\partial T}{\partial t} = k_5(T - t_e) \\ \frac{\partial h_1}{\partial t} = 0 \\ \frac{\partial h_2}{\partial t} = 0 \end{cases}$ |

**Table 2:** Hybrid model of the evaporator process.

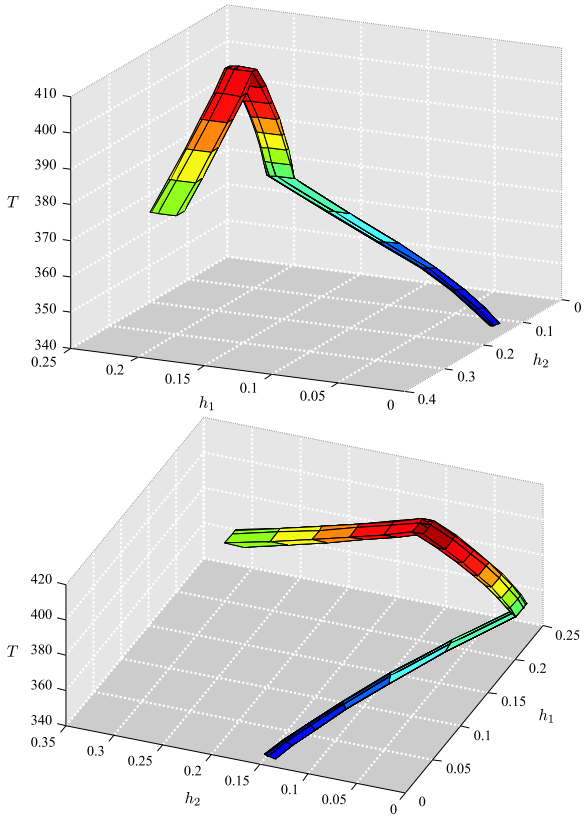



**Figure 7:** Set evolution from $\mathcal{X}_0$ to the target set $\mathcal{Z}_1$ (won) for $T_a = 391$ (same evolution, different perspectives).

To this end, we apply the verification algorithm presented above, and label as target set $\mathcal{Z}_1$ the set of *safe* states $\{x : x_\ell = \left[\begin{smallmatrix}0\\1\\0\end{smallmatrix}\right]\}$ (won), and as target set $\mathcal{Z}_2$ the set of *unsafe* states $\{x : x_\ell = \left[\begin{smallmatrix}0\\1\\0\end{smallmatrix}\right]\}$ (lost). The results of the algorithm are plotted in Figure 7, where the *set evolution* in the three-dimensional continuous state space $h_1$, $h_2$, $T$ from the initial conditions $\mathcal{X}(0)$ is depicted from different view angles.

The tool can also easily perform parametric verification if the vector of parameters $\theta$ enters the model linearly, and its range is a polyhedral set $\Theta$ (e.g. $\Theta$ is an interval). Constant parameters can in fact be taken into account by augmenting the state vector to $\left[\begin{smallmatrix}x\\\theta\end{smallmatrix}\right]$, adding a constant dynamics $\theta(t+1) = \theta(t)$ for the additional state $\theta$, and setting the set of initial conditions to $\mathcal{X}(0) \times \Theta$. Vice versa, varying parameters with unknown dynamics can be modeled as additional inputs to the system, i.e., as disturbances.

We use parametric verification for checking against variations of the alarm temperature $T_a$ in the range 383 K$\leq T_a \leq$ 393 K. As $T_a$ is a constant parameter of the PLC code, it is treated as an additional state.

The parametric verification produces the following result: for $T_a \geq 390.4902$ the controller drives the plant to the terminal state $\mathcal{Z}_1$ (won) for all the initial heights and temperature in $\mathcal{X}_0$. The parametric verification requires 82 s to build the graph of evolution on a PC Pentium II 400 MHz running interpreted Matlab code.

**Performance Characterization**

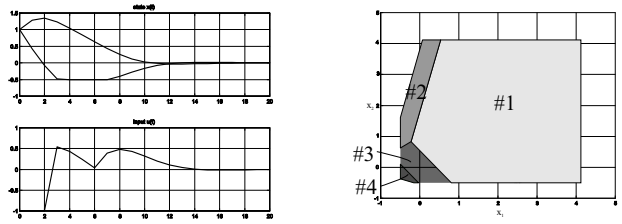
While there are many performance measures for linear systems (ranging from the traditional Integral-Square-Error to the modern $H_{infty}$ criterion) the performance of systems with constraints under MPC is more difficult to characterize in a compact meaningful manner. Obviously, as a minimum requirement the closed loop MPC system must be stable. All the available MPC stability results hold when the associated optimization problem is feasible. Thus, a possible performance charcterization would be to determine that region of the state space for which all emanating trajectories lead to feasible optimization problems as they evolve. To arrive at a more quantitative measure we can define a region $\mathcal{C}_0$ around the origin and determine that region $\mathcal{D}_T(0)$ of the state space for which all emanating trajectories lead into $\mathcal{C}_0$ in $T$ time steps. This problem can also be solved by the proposed verification algorithm as detailed below.

We aim at estimating the domain of attraction of the origin, and the set of initial conditions from which the state trajectory remains feasible for the constraints. As mentioned in the previous section, the nominal MPC closed-loop system is an autonomous PWA system. The origin belongs to the interior of one of the sets of the partition, namely the region where the LQ gain $K$ is asymptotically stabilizing while fulfilling the constraints, which by convention will be referred to as $\mathcal{C}_0$. Denote by $\mathcal{D}_\infty(0) \subseteq \mathbb{R}^n$ the (unknown) domain of attraction of the origin. Given a bounded set $\mathcal{X}(0)$ of initial conditions, we want to characterize $\mathcal{D}_\infty(0) \bigcap \mathcal{X}(0)$.

By construction, the matrix $A_0$, associated with the region $\mathcal{C}_0$, is strictly Hurwitz and $f_0 = 0$ (in fact, in $\mathcal{C}_0$ the feedback gain is the unconstrained LQR gain $F_0 = K$, $g_0 = 0$ (Bemporad et al., 2002)). Then we can compute an invariant set in $\mathcal{C}_0$. In particular, we compute the *maximum output admissible set* (MOAS) $\mathcal{X}_\infty \subseteq \mathcal{C}_0$. $\mathcal{X}_\infty$ is the largest invariant set contained in $\mathcal{C}_0$, which by construction of $\mathcal{C}_0$ is compatible with the constraints $u_{\min} \leq Kx(t) \leq u_{\max}$, $x_{\min} \leq x(t) \leq x_{\max}$. By (Gilbert and Tan, 1991, Th.4.1), the MOAS is a polyhedron with a finite number of facets, and is computed through a finite number of linear programs (LP's) (Gilbert and Tan, 1991).

In order to circumvent the undecidability of stability, we give the following

**Definition 1** *Consider the PWA system (1), and let the origin* $0 \in \mathring{\mathcal{C}}_0 \triangleq \{x : \ H_0 x < S_0\}$*, and* $A_0$ *be strictly Hurwitz. Let* $\mathcal{X}_\infty$ *be the maximum output admissible set (MOAS) in* $\mathcal{C}_0$*, which is an invariant for the linear system* $x(t+1) = A_0 x(t)$*. Let* $T$ *be a finite time horizon. Then, the set* $\mathcal{X}(0) \subseteq \mathbb{R}^n$ *of initial conditions is said to belong to the* domain of attraction in $T$ steps $\mathcal{D}_T(0)$ *of the origin if* $\forall x(0) \in \mathcal{X}(0)$ *the corresponding final state* $x(T) \in \mathcal{X}_\infty$.




(a) Closed-loop MPC. (b) Explicit solution.

**Figure 8:** Example 12.

Note that $\mathcal{D}_T(0) \subseteq \mathcal{D}_{T+1}(0) \subseteq \mathcal{D}_\infty(0)$, and $\mathcal{D}_T(0) \to \mathcal{D}_\infty(0)$ as $T \to \infty$. The horizon $T$ is a practical information about the speed of convergence of the PWA system to the origin and thus about its dynamic performance.

**Definition 2** *Consider the PWA system (1), and let* $\mathcal{X}_{\text{infeas}} \triangleq \mathbb{R}^n \setminus \cup_{i=1}^s \mathcal{C}_i$*. The set* $\mathcal{X}(0) \subseteq \mathbb{R}^n$ *of initial conditions is said to belong to the* domain of infeasibility in $T$ steps $\mathcal{I}_T(0)$ *if* $\forall x(0) \in \mathcal{X}(0)$ *there exists* $t$, $0 \leq t \leq T$ *such that* $x(t) \in \mathcal{X}_{\text{infeas}}$.

In Definition (2), the set $\mathcal{X}_{\text{inf}}$ must be interpreted as a set of "very large" states. Although instability in $T$ steps does not guarantee instability (for any finite $T$, a trajectory might reach $\mathcal{X}_{\text{inf}}$ and converge back to the origin), it has the practical meaning of labeling as "unstable" the trajectories whose magnitude is unacceptable, for instance because the PWA system is no longer valid as a model of the real system.

Given a set of initial conditions $\mathcal{X}(0)$, we aim at finding subsets of $\mathcal{X}(0)$ which are safely asymptotically stable $(\mathcal{X}(0) \bigcap \mathcal{D}_T(0))$, and subsets which lead to infeasibility in $T$ steps $(\mathcal{X}(0) \bigcap \mathcal{I}_T(0))$. Subsets of $\mathcal{X}(0)$ leading to none of the two previous cases are labeled as *non-classifiable in $T$ steps*. As we will use linear optimization tools, we assume that $\mathcal{X}(0)$ is a convex polyhedral set (or the union of convex polyhedral sets). Typically, non-classifiable subsets shrink and eventually disappear for increasing $T$.

**An Example.** Consider the system $y(t) = \frac{s+1}{s^2+s+2}u(t)$, and sample the dynamics with $T = 0.2$ s. The task is to regulate the system to the origin while fulfilling the constraints $-1 \leq u(t) \leq 1$ and $x(t) \geq \left[\begin{smallmatrix}-0.5\\-0.5\end{smallmatrix}\right]$. To this aim, we design an MPC controller based on the optimization problem

$$\begin{aligned} \min_{u_t, u_{t+1}} \quad & ||x_{t+2|t}||_P^2 + \sum_{k=0}^{1} ||x_{t+k|t}||^2 + .1||u_{t+k}||^2 \\ \text{subj. to} \quad & -2 \leq u_{t+k} \leq 2, \ k = 0, 1 \\ & x_{t+k|t} \geq x_{\min}, \ x_{\min} \triangleq \left[\begin{smallmatrix}-0.5\\-0.5\end{smallmatrix}\right], \ k = 0, 1 \end{aligned} \tag{12}$$

where $P$ is the solution to the Riccati equation (in this example $Q = \left[\begin{smallmatrix}1 & 0\\0 & 1\end{smallmatrix}\right]$, $R = 0.1$, $N_u = N_y = N_c = 2$).

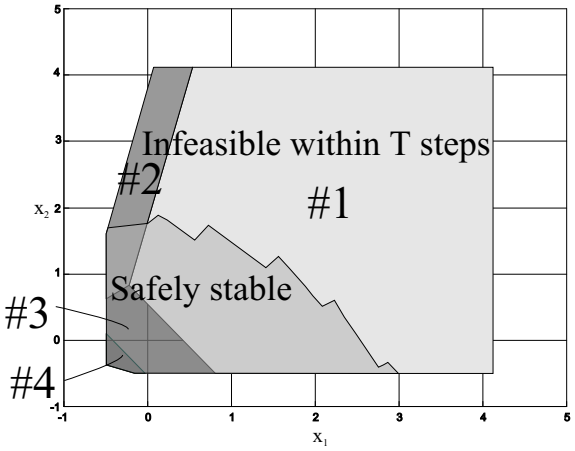



**Figure 9:** Partition of initial states into safely stable, and infeasible in $T = 20$ steps.

Note that this choice of $P$ corresponds to setting $u_{t+k} = Kx_{t+k|t}$ for $k \geq 2$, where $K$ is the LQR gain, and minimizes $\sum_{k=0}^{\infty} x'_{t+k|t}x_{t+k|t} + .01u^2_{t+k}$ with respect to $u_t$, $u_{t+1}$. The closed loop response from the initial condition $x(0) = [1\ 1]'$ is shown in Figure 8(a).

The solution to the mp-QP problem was computed by using the solver in (Bemporad et al., 2002) in 0.66 s on a PC Pentium III 650 MHz running Matlab 5.3, and the corresponding polyhedral partition of the state-space is depicted in Figure 8(b). The MPC law is linear in each one of the four depicted regions.

Region #3 corresponds to the unconstrained LQR controller, #1 and #4 to saturation at $-1$ and $+1$, respectively, and #2 is a transition region between LQR control and the saturation.

Note that the union of the regions depicted in Figure 8(b) should not be confused with the region of attraction of the MPC closed-loop. For instance, by starting at $x(0) = [3.5\ 0]'$ (for which a feasible solution exists), the MPC controller runs into infeasibility after $t = 5$ time steps.

The reachability analysis algorithm described above was applied to determine the set of safely stable initial states and states which are infeasible in $T = 20$ steps (Figure 9). The algorithm computes the graph of evolutions in 115 s on a Pentium II 400 running Matlab 5.3.

## Conclusions

The paper argues that many unsolved problems of practical interest involve systems where dynamics and logic interact. A big subclass of such systems can be modeled in discrete time as Mixed Logic Dynamical (MLD) systems described by linear dynamic equations subject to linear inequalities involving real and integer variables. As an immediate benefit of the MLD description most system analysis and synthesis tasks can be cast as mixed integer optimization problems, for which many commercial solvers exist.

Our group has concentrated on this model paradigm and developed a wide variety of tools and techniques (only a small fraction of which was discussed in this paper), among them: HYSDEL, a modelling language for the specification of MLD systems and a compiler to generate the MLD models; a model predictive controller with an explicit representation where the optimization effort is entirely shifted off-line; an algorithm for the verification of MLD systems which is useful for a variety of tasks ranging from checking the correctness of PLC code to assessing the performance of MPC loops; several algorithms to analyze the observability of MLD systems essential for filter design, process monitoring and fault detection; several filtering algorithms based on the moving horizon idea with rather general convergence guarantees.

In collaboration with different companies we have applied the tools to a range of problems including traction control and gear shift/clutch operation on automotive vehicles, power management for electrical utilities, fault detection on a benchmark multi-tank system, optimal operation of a gas supply system, blood pressure control in anesthesia and analysis of an emergency shutdown system for a pilot batch plant.

All the investigated theoretical problems are "hard" in the mathematical sense (maybe all interesting problems are?), which implies—loosely speaking—that in the worst case the computational effort grows exponentially with the problem size. Thus the future challenge will be to develop approximate methods which provide good, if not optimal answers for problems with specific structures and where the computational effort grows only in a polynomial fashion. Otherwise the applicability of the developed tools will remain limited to small problems.

An extensive set of reports describing all aspects of our work on hybrid systems is available from our web site `http://control.ethz.ch`

## Acknowledgments

The author wishes to thank Domenico Mignone who helped in the preparation of the paper.

## References

Acevedo, J. and E. N. Pistikopoulos, "A Multiparametric Programming approach for linear process engineering problems under uncertainty," *Ind. Eng. Chem. Res.*, **36**, 717–728 (1997).

Alur, R., C. Courcoubetis, T. A. Henzinger, and P.-H. Ho, Hybrid Automata: an algorithmic approach to the specification and verification of hybrid systems, In R. L. Grossman, A. Nerode, A. P. R. and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer Verlag (1993).

Asarin, A., O. Maler, and A. Pnueli, "On the Analysis of Dynam-

ical Systems having Piecewise-Constant Derivatives," *Theoretical Computer Science*, **138**, 35–65 (1995).

Bemporad, A. and M. Morari, "Control of Systems Integrating Logic, Dynamics, and Constraints," *Automatica*, **35**(3), 407–427 (1999).

Bemporad, A., P. Hertach, D. Mignone, M. Morari, and F. D. Torrisi, HYSDEL—Hybrid Systems Description Language, Technical Report AUT00-03, ETH Zurich (2000b).

Bemporad, A., G. Ferrari-Trecate, and M. Morari, "Observability and Controllability of Piecewise Affine and Hybrid Systems," *IEEE Trans. Auto. Cont.*, **45**(10), 1864–1876 (2000a).

Bemporad, A., F. D. Torrisi, and M. Morari, Optimization-Based Verification and Stability Characterization of Piecewise Affine and Hybrid Systems, In Krogh, B. and N. Lynch, editors, *Hybrid Systems: Computation and Control, Proceedings 3rd International Workshop on Hybrid Systems, Pittsburgh, PA, USA*, Lecture Notes in Computer Science. Springer Verlag (2000).

Bemporad, A., M. Morari, V. Dua, and E. N. Pistikopoulos, "The Explicit Linear Quadratic Regulator for Constrained Systems," *Automatica*, **38**(1), 3–20 (2002).

Branicky, M. S., V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control: model and optimal control theory," *IEEE Trans. Auto. Cont.*, **43**(1), 31–45 (1998).

Branicky, M. S., *Studies in Hybrid Systems: Modeling, Analysis, and Control*, PhD thesis, Massachussets Institute of Technology (1995).

Branicky, M. S., "Multiple Lyapunov functions and other analysis tools for switched and hybrid systems," *IEEE Trans. Auto. Cont.*, **43**(4), 475–482 (1998).

Cavalier, T. M., P. M. Pardalos, and A. L. Soyster, "Modeling and integer programming techniques applied to propositional calculus," *Comput. Oper. Res.*, **17**(6), 561–570 (1990).

Christiansen, D., *Electronics Engineers' Handbook, 4th edition.* IEEE Press/ McGraw Hill, Inc. (1997).

De Schutter, B. and B. De Moor, The Extended Linear Complementarity Problem and the Modeling and Analysis of Hybrid Systems, In Antsaklis, P., W. Kohn, M. Lemmon, A. Nerode, and S. Sastry, editors, *Hybrid Systems V*, volume 1567 of *Lecture Notes in Computer Science*, pages 70–85. Springer (1999).

De Schutter, B. and T. van den Boom, Model predictive control for max-plus-linear systems, In *Proc. American Contr. Conf.*, pages 4046–4050 (2000).

Fletcher, R. and S. Leyffer, A mixed integer quadratic programming package, Technical report, University of Dundee, Dept. of Mathematics, Scotland, UK (1994).

Fletcher, R. and S. Leyffer, "Numerical Experience with Lower Bounds for MIQP Branch-And-Bound," *SIAM J. Optim.*, **8**(2), 604–616 (1998). `http://epubs.siam.org/sam-bin/dbq/toclist/SIOPT`.

Floudas, C. A., *Nonlinear and Mixed-Integer Optimization.* Oxford University Press (1995).

Gilbert, E. G. and K. T. Tan, "Linear systems with state and control constraints: the theory and applications of maximal output admissible sets," *IEEE Trans. Auto. Cont.*, **36**(9), 1008–1020 (1991).

Grossmann, R. L., A. Nerode, A. P.Ravn, and H. R. (Eds.), *Hybrid Systems.* Springer Verlag, New York (1993). no. 736 in LCNS.

Heemels, W. P. M. H., J. M. Schumacher, and S. Weiland, "Linear Complementarity Systems," *SIAM J. Appl. Math.*, **60**(4), 1234–1269 (2000).

Heemels, W. P. M. H., B. De Schutter, and A. Bemporad, "Equivalence of Hybrid Dynamical Models," *Automatica*, **37**(7), 1085–1093 (2001).

Heemels, W. P. M. H., *Linear complementarity systems: a study in hybrid dynamics*, PhD thesis, Dept. of Electrical Engineering, Eindhoven University of Technology, The Netherlands (1999).

Johansson, M. and A. Rantzer, "Computation of Piecewise Quadratic Lyapunov Functions for Hybrid Systems," *IEEE Trans. Auto. Cont.*, **43**(4), 555–559 (1998).

Johansson, K. H., M. Egerstedt., J. Lygeros, and S. Sastry, "On the Regularization of Zeno hybrid automata," *Sys. Cont. Let.*, **38**, 141–150 (1999).

Kowalewski, S. and O. Stursberg, The Batch Evaporator: A Benchmark Example For Safety Analysis Of Processing Systems Under Logic Control, In *4th Int. Workshop on Discrete Event Systems (WODES 98)*, Cagliari (Italy) (1998).

Kowalewski, S., Description of VHS Case Study 1 "Experimental Batch Plant", `http://astwww.chemietechnik.uni-dortmund.de/\symbol{126}vhs/cs1descr.zi%p` (1998). Draft. University of Dortmund, Germany.

Labinaz, G., M. M. Bayoumi, and K. Rudie, "A Survey of Modeling and Control of Hybrid Systems," *Annual Reviews of Control*, **21**, 79–92 (1997).

Lazimy, R., "Improved algorithm for mixed-integer quadratic programs and a computational study," *Math Prog.*, **32**, 100–113 (1985).

Lygeros, J., D. N. Godbole, and S. Sastry, A game theoretic approach to hybrid system design, In Alur, R. and T. Henzinger, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 1–12. Springer Verlag (1996).

Lygeros, J., C. Tomlin, and S. Sastry, "Controllers for reachability specifications for hybrid systems," *Automatica*, **35**(3), 349–370 (1999).

Nemhauser, G. L. and L. A. Wolsey, *Integer and Combinatorial Optimization.* Wiley (1988).

Raman, R. and I. E. Grossmann, "Relation between MILP modeling and logical inference for chemical process synthesis," *Comput. Chem. Eng.*, **15**(2), 73–84 (1991).

Roschchin, V. A., O. V. Volkovich, and I. V. Sergienko, "Models and methods of solution of quadratic integer programming problems," *Cybernetics*, **23**, 289–305 (1987).

Sontag, E. D., "Nonlinear Regulation: The Piecewise Linear Approach," *IEEE Trans. Auto. Cont.*, **26**(2), 346–358 (1981).

Sontag, E. D., Interconnected automata and linear systems: A theoretical framework in discrete-time, In Alur, R., T. A. Henzinger, and E. D. Sontag, editors, *Hybrid Systems III—Verification and Control*, number 1066 in Lecture Notes in Computer Science, pages 436–448. Springer-Verlag (1996).

Stursberg, O., The Batch-Evaporator Benchmark Example—A simplified formulation (1999). `http://astwww.chemietechnik.uni-dortmund.de/~olaf/`.

Tyler, M. L. and M. Morari, "Propositional logic in control and monitoring problems," *Automatica*, **35**(4), 565–582 (1999).

van der Schaft, A. J. and J. M. Schumacher, "Complementarity Modelling of Hybrid Systems," *IEEE Trans. Auto. Cont.*, **43**, 483–490 (1998).

Williams, H. P., "Logical problems and integer programming," *Bulletin of the Institute of Mathematics and Its Applications*, **13**, 18–20 (1977).

Williams, H. P., *Model Building in Mathematical Programming.* John Wiley & Sons, Third Edition (1993).

## Appendix

### HYSDEL (HYbrid Systems DEscription Language)

The derivation of the MLD model, i.e., a set of linear difference equations and mixed-integer linear inequalities, from an interconnection of components involving continuous systems and logic is an involved tedious task for all but the most simple example problems. Therefore we have developed a HYbrid Systems DEscription Language (HYSDEL) for the specification of such systems and a compiler which readily generates the equivalent MLD form. The HYSDEL compiler is available on-line at `http://control.ethz.ch/~hybrid/hysdel`. Thanks to the equivalence between the various hybrid system descriptions mentioned in the paper, the MLD form can be used as an intermediate step to obtain the corresponding PWA, LC, ELC, or MMPS counterpart.

In HYSDEL systems are viewed as the interconnection of basic objects. Each object admits an equivalent representation as linear mixed-integer equalities and inequalities. The basic objects are:

- **A/D (Analog-to-Digital) block.** Can extract logic facts from the activity level of linear thresholds.A graphical representation is provided in Figure 10.
- **D/A (Digital-to-Analog) block.** Is the counterpart of the A/D block. This block (see Figure 11) is able to associate with the output different linear expressions according to the different logic value of the input.
- **Automaton.** Evolves according to the logic part of the overall system input and the logic signals coming out from the A/D blocks and from other automata. As the schematic representation depicted in Figure 12 shows, the automaton typically makes its internal state available to other components.
- **Continuous dynamics.** Is a *Discrete-time Linear Time Invariant* (D-LTI) system and the different modes of operation are obtained by connecting (see Figure 13) the input to the output of a D/A block.

As an illustration the HYSDEL code for the batch evaporator example is shown in Figure 14.


**Figure 10:** A/D block—Continuous to logic conversion.

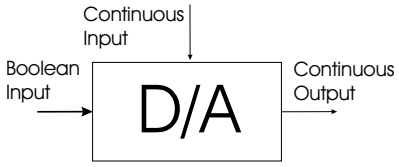



**Figure 11:** D/A block—Logic to continuous conversion.

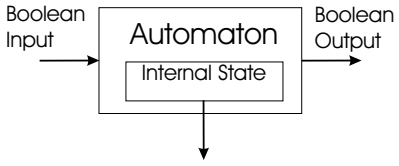



**Figure 12:** Automaton.

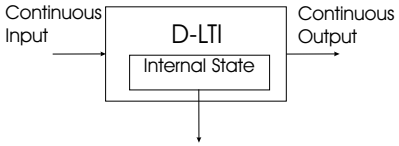



**Figure 13:** Discrete-time-invariant linear continuous dynamics.

```
/* VHS Esprit Project - Case Study 1 */

SYSTEM batchevaporator {
INTERFACE {
    STATE {
        REAL tmp,h1,h2,tal;
        BOOL p1,p2,p3; }
    OUTPUT {
        REAL outreal1;
        BOOL outbool1; }
    PARAMETER {
        REAL q = 5000;    /* kW     */

                (other parameters are omitted for brevity)

    }
}

IMPLEMENTATION {
    AUX {
        REAL zT, zh1a, zh1b, zh1c, zh2a, zh2b, zh2c, zh2d;

        BOOL ti1,ti2,l1,l2,h,v18,v15,d1;
        BOOL l1h1,l1h2; /* Linearization of sqrt */
        BOOL da1,da2,da3,da4,da5,da6,da7;  }
    LOGIC {
        h = ~p1&~p2&~p3;
        v18 = ~p1&~p2&~p3 | ~p1&~p2&p3;
        v15 = p1&~p2&~p3;
        da1=v18&~l1&l1h1;
        da2=v18&~l1&~l1h1;
        da3=~v18&~l1;
        da4=v15&~l2&l1h2;
        da5=~v15&~l2&l1h2;
        da6=v15&~l2&~l1h2;
        da7=~v15&~l2&~l1h2;  }
    AD {
        ti1 = -tmp+tal <= 0 [-Tmin+Talmax,-Tmax+Talmin,1e-2];
        ti2 = tmp-338 <= 0 [Tmax-338,Tmin-338,1e-2];
        l1 = h1-0.01 <= 0 [hmax-0.01,hmin-0.01,1e-6];
        l2 = h2-0.01 <= 0 [hmax-0.01,hmin-0.01,1e-6];
        l1h1 = h1-l1h1t <= 0 [hmax-l1h2t,hmin-l1h2t,1e-6];
        l1h2 = h2-l1h2t <= 0 [hmax-l1h2t,hmin-l1h2t,1e-6];   }
    DA {
        zT =   {IF h THEN atmp1*tmp+btmp1 [atmp1*Tmax+btmp1,atmp1*Tmin+btmp1,0]
                     ELSE atmp2*tmp+btmp2 [atmp2*Tmax+btmp2,atmp2*Tmin+btmp2,0] };
        zh1a = {IF da1 THEN ah1a*h1+bh1a [ah1a*hmax+bh1a,ah1a*hmin+bh1a,0]      };
        zh1b = {IF da2 THEN ah1b*h1+bh1b [ah1b*hmax+bh1b,ah1b*hmin+bh1b,0]      };
        zh1c = {IF da3 THEN h1 [hmax,hmin,0]                                    };
        zh2a = {IF da4 THEN ahh2a*h1+h2+bhh2a [ahh2a*hmax+hmax+bhh2a,ahh2a*hmin+hmin+bhh2a,0]};
        zh2b = {IF da5 THEN ah2a*h2+bh2a [ah2a*hmax+bh2a,ah2a*hmin+bh2a,0]      };
        zh2c = {IF da6 THEN ahh2b*h1+h2+bhh2b [ahh2b*hmax+hmax+bhh2b,ahh2b*hmin+hmin+bhh2b,0]};
        zh2d = {IF da7 THEN ah2b*h2+bh2b [ah2b*hmax+bh2b,ah2b*hmin+bh2b,0]      };   }
    CONTINUOUS {
        tmp =  zT;
        h1 = zh1a + zh1b + zh1c;
        h2 = zh2a + zh2b + zh2c + zh2d;
        tal = tal;  }
    AUTOMATA {
        p1= (~p1&~p2&p3&l2) | (p1&~p2&~p3&~ti2&~l1);
        p2= (p1&~p2&~p3&l1) | (p1&~p2&~p3&ti2) | (~p1&p2&~p3) | (~p1&p2&p3);
        p3= (~p1&~p2&~p3&ti1) | (~p1&~p2&p3&~l2) | (p1&~p2&~p3&ti2) | (~p1&p2&p3);   }
    MUST   {
        ~(ti1 & ti2);      /* Excludes combination ti1,ti2=11 */
        ~(p1 & (p2 | p3)); /* Excludes logical states 101,110,111 */
        ~l1h1|~l1;
        ~l1h2|~l2;  }
}
}
```

**Figure 14:** Example of HYSDEL code for the batch evaporator example.