European Journal of Control (2001)0:1–18

European Journal of Control

Discrete-time Hybrid Modeling and Verification of the Batch Evaporator Process Benchmark

A. Bemporad[1,2], F.D. Torrisi[1,*] and M. Morari[1,†]

[1]Automatic Control Laboratory, ETH Zentrum – ETL, CH-8092 Zürich, Switzerland; [2]Dip. Ingegneria dell'Informazione, Università di Siena, Via Roma 56, I-53100 Siena, Italy

For hybrid systems described by interconnections of linear discrete-time dynamical systems, automata, and propositional logic rules, we recently proposed the Mixed Logical Dynamical (MLD) systems formalism and the language HYSDEL (Hybrid System Description Language) as a modeling tool. For MLD models, we developed a reachability analysis algorithm which combines forward reach set computation and feasibility analysis of trajectories by linear and mixed-integer linear programming. In this paper the versatility of the overall analysis tool is illustrated on the batch evaporator benchmark process.

Keywords: Discrete-time; Hybrid systems; Reachability analysis; Verification

1. Introduction

Hybrid models describe processes which evolve according to dynamic equations and logic rules [2]. The interest in hybrid systems has grown over the last few years not only because of the theoretical challenges, but also because of their impact on applications, for instance in the automotive industry [6]. Although many physical phenomena are hybrid in nature, the main interest here is directed to real-time systems, where physical processes are controlled by embedded controllers. For this reason, it is important to have available tools to guarantee that this combination behaves as desired. Formal verification is aimed at providing such a certification. This amounts to solving the following reachability problem: For a given set of initial conditions and disturbances, guarantee that all possible trajectories never enter a set of unsafe states, or possibly provide a counterexample. Unfortunately, it is well known that formal verification is an undecidable problem [1,26], as many other system theoretical problems such as stability [15] and observability [7] analysis of hybrid systems. In spite of this complexity, several tools for formal verification of hybrid systems have been proposed in the literature [4,14,20,22,35].

Any verification algorithm needs an abstraction of the real process, namely a mathematical model of the process. The model should not be too complicated in order to efficiently define and run a verification algorithm, but also not too simple, otherwise the results are either too conservative or wrong.

Timed automata and hybrid automata have proved to be successful modeling formalisms for verification, and have been widely used in the literature. In the theory of *timed automata* [5], the dynamic part is the continuous-time flow $\dot{x} = 1$. Timed automata were extended to *linear hybrid automata* [1], where the dynamics is modeled by the differential inclusion $a \leq \dot{x} \leq b$. On the other side, the control community

*torrisi@aut.ee.ethz.ch.
†morari@aut.ee.ethz.ch.
Correspondence and offprint requests to: A. Bemporad, Automatic Control Laboratory, ETH Zentrum–ETL, CH-8092, Zürich, Switzerland. Tel.: +41-1-632 6679; Fax: +41-1-632 1211; Email: bemporad@aut.ee.ethz.ch.

AQ: please check the correspondence adress

Received 4 February 2001; Accepted in revised form 21 March 2001. Recommended by O. Maler and B. Benveniste.

started studying the so-called *hybrid dynamical systems* [2,18,34] where the switching between different dynamics is governed by a finite state machine. A special case where dynamic equations and switching rules are linear functions of the state are the so-called Piecewise Affine (PWA) systems [38]. These are defined by partitioning the state space into polyhedral regions, and associating with each region a different linear state-update equation. PWA systems can model a large number of physical processes, such as systems with static nonlinearities (for instance actuator saturation), and can approximate nonlinear dynamics via multiple linearizations at different operating points. The study of PWA systems is also motivated by the stability and performance analysis of high-performance controllers [32]. In particular, in [10] the authors showed that a model predictive controller (MPC) for constrained linear systems can be expressed explicitly in closed-form as a continuous and piecewise affine state-feedback law. The resulting closed-loop system is therefore PWA, and the criteria for proving stability and robust stability against disturbances and model uncertainties are of fundamental importance [13].

PWA systems are equivalent to interconnections of linear systems and finite automata, as pointed out by Sontag [38]. Based on different arguments, a similar result was proved constructively for discrete-time hybrid models in [7] and extended in [21], where the authors show that PWA systems, linear complementarity (LC) and extended linear complementarity (ELC) systems, max-min-plus-scaling (MMPS) systems, and mixed logical dynamical (MLD) systems are equivalent.

In particular, the MLD framework [9] allows specifying the evolution of continuous variables through linear dynamic equations, of discrete variables through propositional logic statements and automata, and the mutual interaction between the two. The key idea of the approach consists of embedding the logic part in the state equations by transforming Boolean variables into 0–1 integers, and by expressing the relations as mixed-integer linear inequalities. Such a translation procedure is the core of the tool HYSDEL (Hybrid Systems Description Language), which automatically generates an MLD model from a high-level textual description of the system [40].

MLD systems are formulated in discrete time. Despite the fact that the effects of sampling can be neglected in most applications, subtle phenomena such as Zeno behaviors [25] cannot be captured in discrete time. Although reformulating MLD systems in continuous-time would be quite easy from a theoretical point of view, a discrete-time formulation allows one to develop numerically tractable schemes

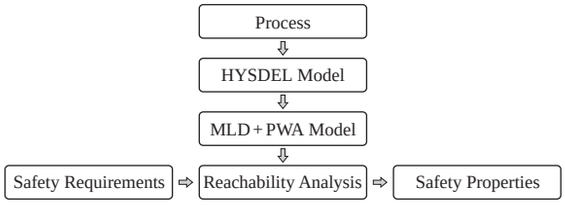

Fig. 1. The overall tool for verification of hybrid systems: The hybrid dynamic model is specified in HYSDEL, translated into MLD and PWA form, and used by the reachability analysis algorithm for proving safety properties.

for solving complex analysis and synthesis problems. Several questions related to MLD systems can indeed be suitably formulated as mixed-integer linear/quadratic optimization problems. For feedback control, in [9] the authors propose a model predictive control scheme which is capable of stabilizing MLD systems on desired reference trajectories while fulfilling operating constraints, and possibly take into account previous qualitative knowledge in the form of heuristic rules. Formal verification algorithms were developed in [12] for MLD hybrid systems, extended in [8] for solving scheduling problems using combined reachability analysis and quadratic optimization, and in [13] for stability and performance analysis of hybrid control systems.

In this paper, we review the basics of MLD systems, the specification language HYSDEL, the reachability analysis algorithm described in [12], and we address the formal verification problem of the batch evaporator benchmark [23,28,30]. We model the process in HYSDEL (in particular, the subsystem defined in [29]) to obtain an MLD system. This consists of three continuous states, and of a finite state automaton representing a programmable logic controller (PLC) control sequence. For formal verification, we adopt the algorithm in [12], where safety tests and reach set computation are done via linear programming (LP), switching detection via mixed-integer linear programming (MILP), and where the reach sets are approximated by using tools from computational geometry. The overall modeling and verification procedure is depicted in Fig. 1.

2. Mixed Logical Dynamical and Piecewise Affine Systems

Several modeling frameworks were proposed in the control literature. Two main categories were successfully adopted for analysis and synthesis [16]: *hybrid control systems* [1,3,9,33,34], which consist of the interaction between continuous dynamical systems

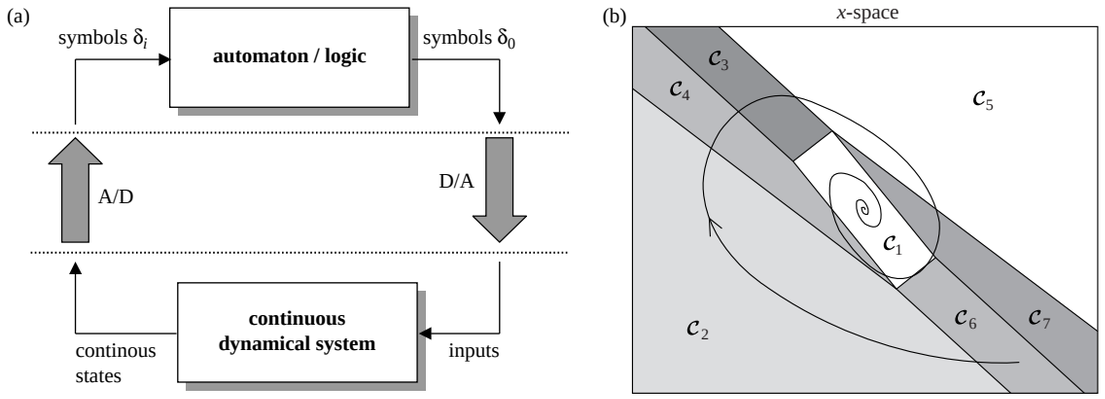

Fig. 2. Hybrid models. (a) Hybrid control systems. Finite state machines and continuous dynamics interact through analog-to-digital (A/D) and D/A interfaces. (b) Switched systems. For each region of the partition of the state space there is a different set of dynamic equations.

and discrete/logic automata (Fig. 2a), and *switched systems* [17,24,37], where the state space is partitioned into regions, each one associated with a different continuous dynamics (Fig. 2b).

Switched systems defined by a polyhedral partition of the state-space and linear dynamic equations are the so-called PWA systems

$$\begin{aligned} x(t+1) &= A_{i(t)}x(t) + B_{i(t)}u(t) + f_{i(t)}, \\ i(t) \quad &\text{s.t. } x(t) \in \mathcal{C}_i \triangleq \{x : H_i x \leq K_i\}, \end{aligned} \tag{1}$$

where $x \in X \subseteq \mathbb{R}^n$, $u \in U \subseteq \mathbb{R}^m$, $\{\mathcal{C}_i\}_{i=0}^{s-1}$ is a polyhedral partition of the sets of states[1] X, the matrices A_i, B_i, f_i, H_i, K_i are constant and have suitable dimensions, and the inequality $H_i x \leq K_i$ should be interpreted componentwise. Without additional hypotheses on the continuity of the piecewise affine state-update mapping, definition (1) is not well posed, as the state-update function is twice (or more times) defined over common boundaries of sets $\mathcal{C}_i$ (the boundaries will also be referred to as *guardlines*). This is a technical issue which can be avoided as in [37] by dealing with open polyhedra and boundaries separately, but it is not of practical interest for the numerical approach taken in this paper.

In this paper we will adopt the MLD system framework introduced in [9]. MLD systems are hybrid systems defined by the interaction of logic, finite state machines, and linear discrete-time systems, as shown in Fig. 2a. The ability to include constraints, constraint prioritization, and heuristics adds to the expressiveness and generality of the MLD framework.

[1] More generally, $\mathcal{C}_i$ are subsets the of state + input space $\mathbb{R}^{n+m}$, for instance when the PWA system is obtained as an equivalent representation of an MLD system [7].

The MLD modeling framework relies on the idea of translating logic relations into mixed-integer linear inequalities [9,19,36,41,43,44]. By following standard notation, we adopt capital letters X_i to represent propositions, 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 propositions to be combined by means of *connectives*: "$\wedge$" (AND), "$\vee$" (OR), "$\sim$" (NOT), "$\rightarrow$" (implies), "$\leftrightarrow$" (if and only if, IFF), "$\oplus$" (exclusive or, XOR). A literal X_i can be associated with a *logical variable* $d_i \in \{0,1\}$, which has a value of either 1 if X_i is TRUE, or 0 otherwise. A propositional logic problem, where a proposition X_1 must be proved to be true given a set of propositions involving literals $X_1, \ldots, X_n$, can be solved by means of a linear integer program by suitably translating the original propositions into linear inequalities involving logical variables d_i [19]. Some basic translations are reported in Table 1.

Such 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 propositions and continuous dynamical variables, in the form of logic propositions derived from conditions on physical dynamics, is provided by properties (analog-to-digital interface (ADI)), (digital-to-analog interface (DAI)) in Table 1, and leads to *mixed-integer linear inequalities*, i.e. linear inequalities involving both *continuous variables* of $\mathbb{R}^n$ and *logical* (or *binary*) variables in $\{0,1\}$. Consider, for instance, the proposition $X \triangleq [a_1' x \leq b_1]$, where $x \in \mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{X}$ is a given bounded set, $a \in \mathbb{R}^n$, $b \in \mathbb{R}$, and let

$$M_i \geq \sup_{x \in \mathcal{X}} (a_i' x - b_i), \; m_i \leq \inf_{x \in \mathcal{X}} (a_i' x - b_i), \quad i = 1, 2.$$

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

	Relation	Logic	(In)equalities
L1	AND ($\wedge$)	$X_1 \wedge X_2$	$d_1 = 1,\ d_2 = 1$
L2	OR ($\vee$)	$X_1 \vee X_2$	$d_1 + d_2 \geq 1$
L3	NOT ($\sim$)	$\sim X_1$	$d_1 = 0$
L4	XOR ($\oplus$)	$X_1 \oplus X_2$	$d_1 + d_2 = 1$
L5	IMPLY ($\rightarrow$)	$X_1 \rightarrow X_2$	$d_1 - d_2 \leq 0$
L6	IFF ($\leftrightarrow$)	$X_1 \leftrightarrow X_2$	$d_1 - d_2 = 0$
L7	ASSIGNMENT	$X_3 = X_1 \wedge X_2$	$d_1 + (1 - d_3) \geq 1$
	($=, \leftrightarrow$)	$X_3 \leftrightarrow X_1 \wedge X_2$	$d_2 + (1 - d_3) \geq 1$
			$(1 - d_1) + (1 - d_2) + d_3 \geq 1$
AD1	EVENT	$[a'x \leq b] \leftrightarrow [d = 1]$	$a'x - b \leq M - Md$
		$x,\ a \in \mathbb{R}^n,\ b \in \mathbb{R}$	$a'x - b \geq \epsilon + (m - \epsilon)d$
DA1	IF-THEN-ELSE	IF X THEN $z = a_1'x - b_1$ ELSE $z = a_2'x - b_2$	$(m_2 - M_1)d + z \leq a_2'x - b_2$
	(= Product)	$(z = d(a_1'x - b_1) + (1 - d)(a_2'x - b_2))$	$(m_1 - M_2)d - z \leq -a_2'x + b_2$
			$(m_1 - M_2)(1 - d) + z \leq a_1'x - b_1$
			$(m_2 - M_1)(1 - d) - z \leq -a_1'x + b_1$

By associating a binary variable d with the literal X, one can transform $X \triangleq [a_1'x \leq b_1]$ into the pair of mixed-integer inequalities described in (AD1), Table 1, where ϵ is a small tolerance (typically the machine precision), beyond which the constraint is regarded as violated. Note that tolerances could be avoided by defining the second inequality in (AD1) as $a'x - b > md$. On the other hand, as we are interested in developing numerical tools for MLD models, strict inequalities $a'x > b$ must be approximated by $a'x \geq b + \epsilon$ for some $\epsilon > 0$, with the assumption that $0 < a'x - b < \epsilon$ cannot occur due to the finite precision of the computer. Note also that sometimes translations require the introduction of *auxiliary variables* [44, p. 178]. For instance, according to (DA1), a quantity which is either $a_1'x - b_1$ if X is true, or $a_2'x - b_2$, requires the introduction of a real variable z.

The rules of Table 1 can be generalized for relations involving n literals $X_1, \ldots, X_n$ combined by arbitrary connectives. Any Boolean expression of logical variables defined by the grammar

$$\begin{aligned} \phi ::= X | \sim \phi_1 | \phi_1 \vee \phi_2 | \phi_1 \oplus \phi_2 | \phi_1 \wedge \phi_2 | \phi_1 \\ \leftarrow \phi_2 | \phi_1 \rightarrow \phi_2 | \phi_1 \leftrightarrow \phi_2 | (\phi_1), \end{aligned} \tag{2}$$

where X is a Boolean variable, can be translated into the conjunctive normal form (CNF)

$$\bigwedge_{j=1}^{k} \left(\bigvee_{i \in P_j} X_i \bigvee_{i \in N_j} \sim X_i \right), N_j, P_j \subseteq \{1, \ldots, n\},$$

by using standard methods [40]. As an example, the proposition (L7) $X_3 \leftrightarrow X_1 \wedge X_2$ is equivalent to

$$(\sim X_1 \vee \sim X_2 \vee X_3) \wedge (X_1 \vee \sim X_3) \wedge (X_2 \vee \sim X_3). \tag{3}$$

Subsequently, the CNF can be translated into the set of integer linear inequalities

$$\begin{cases} 1 \leq \sum_{i \in P_1} d_i + \sum_{i \in N_1} (1 - d_i) \\ \vdots \\ 1 \leq \sum_{i \in P_k} d_i + \sum_{i \in N_k} (1 - d_i). \end{cases} \tag{4}$$

For instance, one can verify that the CNF (3) maps to the inequalities reported in Table 1 (L7). The state-update law of finite state machines can be described by logic propositions involving binary states, their time updates, and binary signals. 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 in Section 5.

By collecting the equalities and inequalities generated by the translation of the automata, ADI, DAI, and by including the linear dynamic difference equations, we can model the hybrid system depicted in Fig. 2a as the MLD system

$$x(t+1) = \Phi x(t) + \mathcal{G}_1 u(t) + \mathcal{G}_2 d(t) + \mathcal{G}_3 z(t), \tag{5a}$$

$$y(t) = \mathcal{H} x(t) + \mathcal{D}_1 u(t) + \mathcal{D}_2 d(t) + \mathcal{D}_3 z(t), \tag{5b}$$

$$\mathcal{E}_2 d(t) + \mathcal{E}_3 z(t) \leq \mathcal{E}_1 u(t) + \mathcal{E}_4 x(t) + \mathcal{E}_5. \qquad (5c)$$

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, $d \in \{0,1\}^{r_\ell}$, $z \in \mathbb{R}^{r_c}$ represent auxiliary binary and continuous variables respectively, which are introduced when transforming logic relations into mixed-integer linear inequalities, and Φ, $\mathcal{G}_{1-3}$, $\mathcal{H}$, $\mathcal{D}_{1-3}$, $\mathcal{E}_{1-5}$ are matrices of suitable dimensions. The inequalities (5c) include the constraints obtained by the D/A and A/D parts of the system, logic, and automata, as well as possible physical constraints on states and inputs. The description (5) seems to be linear, because the nonlinearity is concentrated in the integrality constraints over binary variables.

We assume that system (5) is *completely well-posed* [9], which means that for all x, u within a bounded set the variables d, z are uniquely determined, i.e. there exist functions F, G such that, at each time t, $d(t) = F(x(t), u(t))$, $z(t) = G(x(t), u(t))$[2]. This allows assuming that $x(t+1)$ and $y(t)$ are uniquely defined once $x(t)$, $u(t)$ are given, and therefore that 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 (5c), and therefore this hypothesis is typically fulfilled by MLD relations derived from modeling real-world plants. Nevertheless, a numerical test for well-posedness is reported in [9, Appendix 1].

3. HYSDEL

In the previous sections we have described a technique for transforming the hybrid system into the set of linear mixed-integer equalities and inequalities (5), denoted as MLD system. The language HYSDEL fully automatizes the translation procedure. Given a textual description of the logic and dynamic parts of the hybrid system, HYSDEL returns the matrices Φ, $\mathcal{G}_{1-3}$, $\mathcal{H}$, $\mathcal{D}_{1-3}$, $\mathcal{E}_{1-5}$ of the corresponding MLD form (5). A full description of HYSDEL is reported in [40]. The compiler is available at
http://control.ethz.ch/~hybrid/hysdel.

A HYSDEL description is composed of two parts: The first one, called `INTERFACE`, contains the declaration of all the continuous and binary state, input, and output variables, and parameters. The second part, `IMPLEMENTATION`, is composed of six specialized sections where the relations among variables are described, as detailed below. An example HYSDEL code is reported in Appendix A.

3.1. `AD` and `DA` Sections

These two sections define the interactions among the continuous and discrete variables. The `AD` section defines the Boolean variables from linear-threshold conditions over the continuous variables. For instance, $[d = 1] \leftrightarrow [a_1 x_1 + a_2 x_2 \leq b]$, $x_1, x_2, a_1, a_2 \in \mathbb{R}^n$, $b \in \mathbb{R}$, $d \in \{0,1\}$, with $(a_1 x_1 + a_2 x_2 - b) \in [\min, \max]$. In HYSDEL this condition is represented as:

```
AD{d = a1 * x1 + a2 * x2 - b.
     <= 0[max, min, e];}.
```

The expression `[max, min, e]` denotes the upper and the lower bounds of the function for which the threshold is specified, and `e` is the tolerance mentioned in Section 2. HYSDEL translates the `AD` section into mixed-integer inequalities as in (AD1), Table 1. The symmetric HYSDEL construct `DA` specifies rules of the type "if $[d = 1]$ then $z = a_1 x + b_1$ else $z = a_2 x + b_2$", where $(a_1 x + b_1) \in [\min_1, \max_1]$ and $(a_2 x + b_2) \in [\min_2, \max_2]$:

```
DA{z = {IF d THEN a1 * x - b1 [max1, min1, 0]
          ELSE a2 * x - b2 [max2,min2,0]};}.
```

HYSDEL translates the `DA` section into mixed-integer inequalities according to the equivalence (AD1), Table 1.

3.2. `LOGIC` Section

This section specifies arbitrary functions $X_n = f(X_1, X_2, \ldots, X_{n-1})$ of Boolean variables, where f is a Boolean expression defined by the grammar (2). For example, the formula $d_1 = d_2 \wedge (d_3 \vee \sim d_4)$ is represented in HYSDEL as:

```
LOGIC{d1 = d2 & (d3|~d4); }
```

and translated into the mixed-integer inequalities by computing the associated CNF and then using (4).

3.3. `CONTINUOUS` Section

This section describes linear dynamics as difference equations. Consider, for instance, the first-order linear dynamic equation $\dot{x}(t) = -ax(t) + bu(t)$ and its

[2] For a more general definition of well-posedness, see [9].

equivalent discrete-time counterpart $x(k+1) = e^{-aT}x(k) - (1-e^{-aT})a/bu(k)$ where T is the sampling time. In HYSDEL, this is described as:

```
CONTINUOUS{ x = exp(-a * T) * x
            + (1 - exp(-a * T))/a * b * u; }
```

3.4. AUTOMATA Section

This section specifies the state transition equations of automata, under the assumptions that transitions are clocked and synchronous with the sampling time of the continuous dynamical equations, and that the automaton is well-posed according to the definition recalled in Section 2. (Fig. 3).

The first step is to associate a Boolean representation with each of the n_s logic states of the automata (see Fig. 3). This can be done by defining a logic state vector x_ℓ of size $\lceil \log_2 n_s \rceil$ ($x_\ell = \begin{bmatrix} x_{\ell 1} \\ x_{\ell 2} \end{bmatrix}$ in the example). Using standard tools from digital circuit design, one can derive the state transition function for the automaton:

$$x_\ell(t+1) = F(x_\ell(t), u_\ell(t)),$$

where u_l is the vector of Boolean signals defining the transitions of the automaton. Therefore the automaton is equivalent to a nonlinear discrete time system where F is a purely Boolean function. The state transition function is inserted in the HYSDEL section AUTOMATA. The Boolean values of the states x_ℓ that are not associated with any logic state of the automaton can be explicitly ruled out in the MUST section of the HYSDEL code.

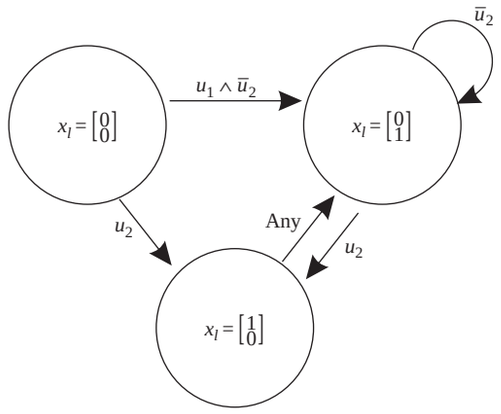

Fig. 3. Example of automaton with $n_s = 3$ logic states, represented with $2 = \lceil \log_2 n_s \rceil$ logic variables and $u_\ell = [u_1\ u_2]' \in \{0,1\}^2$.

An example is reported later in Section 5, where the PLC code that handles the exceptions in the batch evaporator system is modeled in HYSDEL.

3.5. MUST Section

This section allows to specify linear constraints on continuous variables, and logic constraints of the type "$F(X_1, \ldots, X_N)$ is TRUE", where F is a Boolean expression. For example: The variable x must remain in the range $[x_{\min}, x_{\max}]$, and the binary signals u_1, u_2 are mutually exclusive. In HYSDEL these conditions are represented as:

```
MUST { x - xmax <= 0; -x + xmin <= 0;
       (u1 & ~u2) | (~u1 & u2);}.
```

The textual description of the hybrid model is processed by the HYSDEL compiler, which generates the matrices of the MLD system (5). Once the MLD model is available, its equivalent PWA form (1) is obtained by using the procedure suggested in [7]. In the next section, we will assume that both the PWA and the MLD forms are available and discuss their complementary role in the reachability analysis algorithm.

4. Reachability Analysis

The reachability analysis of hybrid dynamical systems allows the verification of *safety properties*: For a given set of initial conditions and exogenous signals, verify that the set of unsafe states cannot be entered, or provide a counterexample. More precisely, we define the following:

Reachability Analysis Problem Given a hybrid system Σ (either in PWA form (1) or MLD (5)), a set of initial conditions $\mathcal{X}(0)$, a collection of disjoint target sets $\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_L$, a bounded set of inputs $\mathcal{U}$, and a time horizon $t \le T_{\max}$, determine (i) if $\mathcal{Z}_j$ is reachable from $\mathcal{X}(0)$ within $t \le T_{\max}$ steps for some sequence $\{u(0), \ldots, u(t-1)\} \subseteq \mathcal{U}$ of exogenous inputs; (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 $\{u(0), \ldots, u(t-1)\} \subseteq \mathcal{U}$, $t \le T_{\max}$, which drives x_1 to x_2 (Fig 4).

From now on, we will assume that $\mathcal{X}(0)$, $\mathcal{Z}_j$, $\mathcal{U}$ are polyhedral sets, and, without loss of generality, that they are also convex. We will also 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 $u(k) \in \mathcal{U}$, $0 \le k \le t-1$. The reason for focusing on finite-time reachability is that the time-horizon $T_{\max}$ has a clear meaning, namely that states which are not reachable in less than $T_{\max}$ steps are in practice unreachable. Although finite time reachability analysis cannot guarantee certain liveness properties (for instance, if $\mathcal{Z}_i$ will be ever reached), the reachability problem stated above is clearly decidable [31,42].

Nevertheless, the problem is $\mathcal{NP}$-hard. To see this, for simplicity consider that system Σ is piecewise linear ($f_i = 0$, for all $i = 0, \ldots, s-1$), and autonomous ($B_i = 0$ for all $i = 0, \ldots, s-1$). Its evolution is

$$x(T) = A_{i(T-1)} A_{i(T-2)} \cdots A_{i(0)} x(0), \tag{6}$$

where $i(k) \in \{0, \ldots, s-1\}$ is the index such that $H_{i(k)} x(k) \le K_{i(k)}$, $k = 0, \ldots, T-1$. The previous questions of reachability can be answered once all the switching sequences $I(T) \stackrel{\Delta}{=} \{i(0), \ldots, i(T-1)\}, \forall T \le T_{\max}$ leading to $\mathcal{Z}_1$, or $\mathcal{Z}_2, \ldots$, or $\mathcal{Z}_L$ from $\mathcal{X}(0)$ are known. In fact, it is enough to check that the reach set at time T, $\mathcal{X}(T, \mathcal{X}(0)) \stackrel{\Delta}{=} A_{i(T-1)} A_{i(T-2)} \cdots A_{i(0)} \mathcal{X}(0)$, satisfies $\mathcal{X}(T, \mathcal{X}(0)) \cap \mathcal{Z}_j \neq \emptyset$ for all admissible switching sequences $I(T)$. However, the number of all possible switching sequences $I(T)$ is combinatorial with respect to T and the number of partitions s, and any enumeration method would be impractical. We recall the verification algorithm proposed in [12] that will be used to avoid such an enumeration (see Table 2).

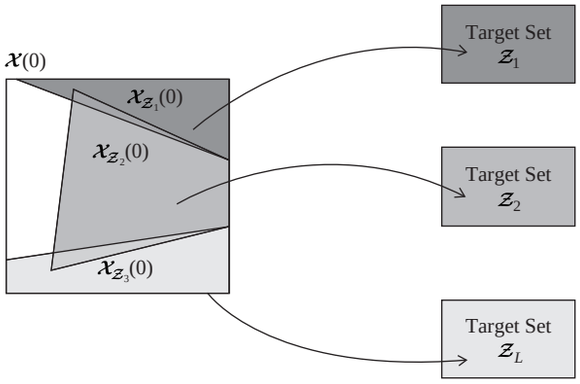

Fig. 4. Reachability analysis problem.

4.1. Verification Algorithm

In order to determine admissible switching sequences $I(t)$, we need to exploit the special structure of the PWA system (1). This allows an easy computation of the reach set, as long as the evolution remains within a single region $\mathcal{C}_i$ of the polyhedral partition. Whenever the reach set crosses a guardline and enters a new region $\mathcal{C}_j$, a new reach set computation based on the j-th linear dynamics is computed, as shown in Fig. 5a. The Algorithm proposed in [12] is summarized by the pseudo-code reported in Table 2, where we assume that $\mathcal{X}(0) \subset \mathcal{C}_i$ is a convex polyhedral set (more generally, we can consider all nonempty intersections $\mathcal{X}(0) \cap \mathcal{C}_i$, for all $i = 1, \ldots, s$). The algorithm determines the reach set $\mathcal{X}(T, \mathcal{X}(0))$ for all $T \le T_{\max}$ (i.e., the *reachable set*).

Table 2 Reachability analysis algoritham.

Algorithm 4.1

```
1   Given: X(0) (to be explored), target sets Z_j, j = 1,...,L;
2   push X(0, X(0)) in STACK ;
3   while STACK not empty do
4      pop X(t, R_j) from STACK, let i such that R_j ⊆ C_i;
5      while not terminate(X(t, R_j))
6         if X(t, R_j) ∩ Z_k ≠ ∅ then Z_k can be reached from R_j
7         t ← t + 1; X(t, R_j) ← A_i X(t − 1, R_j) + B_i U + {f_i};   % set
8         for all h ≠ i such that R_h ≜ C_h ⋂ X(t, R_j) ≠ ∅   % R_j
9            R̄_h ← outer approximation of R_h;
10           Push R̄_h on STACK;
11        X(t, R_j) ← X(t, R_j) ⋂ C_j;
12     end while ;
13  end while ;
14  end .
```

4.1.1. Reach Set Computation

The reach set $\mathcal{X}(t, R_j) \cap \mathcal{C}_i$ is computed iteratively in steps 4, 7, and 11. Note that the subset $S_i(t, \mathcal{X}(0))$ of $\mathcal{X}(0)$ whose evolution lies in $\mathcal{C}_i$ for t steps is given by the explicit representation

$$S(t, \mathcal{X}(0)) = \Bigg\{ x \in \mathbb{R}^n : \; S_0 x \le T_0, H_i A_i^k x \le S_i - H_i \sum_{j=0}^{k-1} A_i^j [B_i u(k-1-j) + f_i], k = 0, \ldots, t \Bigg\}. \tag{7}$$

As $S(t, \mathcal{X}(0))$ is a polyhedral set in the augmented space of tuples $(x, u(0), \ldots, u(t-1))$, the reach set $\mathcal{X}(t, \mathcal{X}(0)) \cap \mathcal{C}_i$ is a polyhedral set as well. A compact representation of $\mathcal{X}(t, \mathcal{X}(0)) \cap \mathcal{C}_i$ (as inequalities over the final state $x(t)$) can be computed by projecting $S(t, \mathcal{X}(0))$ onto the affine subspace $A_i^t x + \sum_{k=0}^{t-1} A_i^k [B_i u(t-1-k) + f_i]$, although this is not required by Algorithm 4.1. Note that the set $\mathcal{X}(t, \mathcal{X}(0)) \bigcap \mathcal{C}_i$ is a polyhedral set.

4.1.2. Guardline Crossing Detection

Step 8 requires a switching detection, namely finding all possible new regions $\mathcal{C}_h$ entered by the reach set at the next time step, or, in other words, all the nonempty sets $R_h \stackrel{\Delta}{=} \mathcal{X}(t, \mathcal{X}(0)) \bigcap \mathcal{C}_h, \; h \neq i$. Rather than enumerating and checking nonemptiness for all $h = 0, \ldots, i-1, i+1, \ldots, s-1$, we can exploit the equivalence between PWA and MLD, and solve the switching detection problem via mixed-integer linear programming. In fact, switching detection amounts to finding all feasible vectors $d(t) \in \{0,1\}^{r_\ell}$ which are compatible with the constraints in (5) plus the constraint $x(t-1) \in \mathcal{X}(t-1, \mathcal{X}(0)) \cap \mathcal{C}_i$. Such a problem is a mixed-integer linear feasibility test, and can be efficiently solved through standard recursive branch and bound procedures. In the average case, the MLD form (through the branch and bound algorithm) requires a number of feasibility tests which is much smaller than enumerating and solving a feasibility test for all the possible regions of the PWA model.

4.1.3. Approximation of Intersection

The computation of the reach set proceeds in each region $\mathcal{C}_h$ from each new intersection R_h. A new reach set computation is started from R_h, unless R_h is contained in some larger subset of $\mathcal{C}_h$ which was already explored. As in principle the number of facets of R_h grows linearly with time, we need to approximate R_h in step 9, so that its complexity is bounded (and therefore the reach set computation from R_h has a limited complexity with respect to the initial region). Hyper-rectangular approximations $\bar{R}_h$ of R_h are the best candidates, as checking for set inclusion between hyper-rectangles reduces to a simple comparison of the coordinates of the vertices. On the other hand, a crude rectangular outer approximation of R_h might lead to explore large regions which are not reachable from the initial set $\mathcal{X}(0)$, as they are just introduced by the approximation itself. In [11] the authors propose an iterative method for inner and outer approximation which is based on linear programming, and approximates with arbitrary precision polytopes by a collection of hyper-rectangles, as depicted in Fig. 5b.

4.1.4. Termination of Explorations

In Section 4.1.1 we showed how to compute the evolution of the reach set $\mathcal{X}(t, R_j)$ inside a region $\mathcal{C}_i$. The computation is stopped (step 5) once one of the following condition happens:

1. The set $\mathcal{X}(t, R_j)$ is empty. This means that the whole evolution has left region $\mathcal{C}_i$.
2. $\mathcal{X}(t, R_j) \subseteq \mathcal{Z}_j, j = 1, \ldots, L$, the target set $\mathcal{Z}_j$ has been reached by all possible evolutions from R_j.
3. $t > T_{\max}$.

These conditions can be easily checked through LP. Note that the termination condition 2 implies that once a target set has been reached no further exploration is performed.

4.1.5. Post-processing

The result of the exploration algorithm detailed in the previous sections can be conveniently stored in a graph G. The nodes of G represent sets from which a reach set evolution is computed, and an oriented arc of G connects two nodes if a transition exists between the two corresponding sets. Each arc has an associated weight which represents the time-steps needed for the transition. The graph has initially no arc, and the nonempty initial set $\mathcal{X}(0)$ and $\mathcal{Z}_j$, $j = 1, \ldots, L$ as nodes. When a new intersection $\mathcal{X}(t, \mathcal{X}(0)) \cap \mathcal{C}_h$ is detected, it is approximated by a collection of hyper-rectangles, as described in Section 4.1.3. Each hyper-rectangle becomes a new node in G, and is connected by a

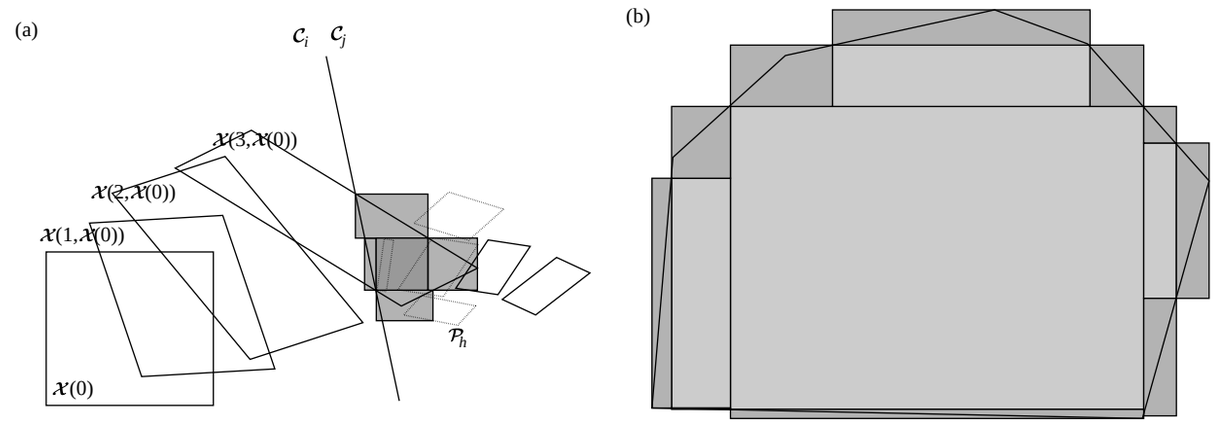

Fig. 5. Reachability analysis. (a) Reach set evolution, guardline crossing, outer approximation of a new intersection, (b) Outer rectangular approximation of a new region intersection.

weighted arc from $\mathcal{X}(0)$. In addition, each hyperrectangle is pushed on a stack of sets to be explored.

Before starting a new reach set computation from a set R_j extracted from the stack, we check for inclusion of R_j in other nodes of G. If this happens, say $R_j \subseteq R_1$ and $R_j \subseteq R_2$, the node associated with R_j is removed from G, and the arcs are redirected.

After Algorithm 4.1 terminates, the oriented paths on G from initial node $\mathcal{X}(0)$ to terminal nodes $\mathcal{Z}_j, j = 1, \ldots, L$ determine a superset of feasible switching sequences $I(t) = \{i(0), \ldots, i(t-1)\}$. In fact, because of the outer approximation $\bar{R}_h$ of new intersections R_h (step *9*), not all switching sequences are feasible. Feasibility can be simply tested via LP over the sets of linear inequalities generated by an explicit reach set representation as in (7).

5. Verification of the Batch Evaporator Process Benchmark

In this section we apply the tools proposed in the previous sections to the benchmark problem proposed within the ESPRIT-LTR Project 26270 VHS (Verification of Hybrid Systems)[3] as Case Study 1. The full system consists of an experimental batch plant [28]. In this paper we will focus only on the *evaporator system*, as proposed in [29], which is schematically depicted in Fig. 6.

The considered subsystem is composed of three parts: the upper tank 1 (labeled as `B5` in Fig. 6), 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 empty when its level is below 1 cm.

During normal operation of the plant, an aqueous solution with a low concentration of salt enters tank 1. The heating is turned on, and when the production of steam begins, the concentration of salt in tank 1 starts to rise. The outgoing steam flows through the condenser and is drained away from the system. 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.

In this paper we focus on the exception handling required 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 in tank 1 rises. The heating system should be switched off to prevent damage 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.

A PLC controls the plant according to the finite state machine depicted in Fig. 7, 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 malfunction of the condenser is detected, the

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

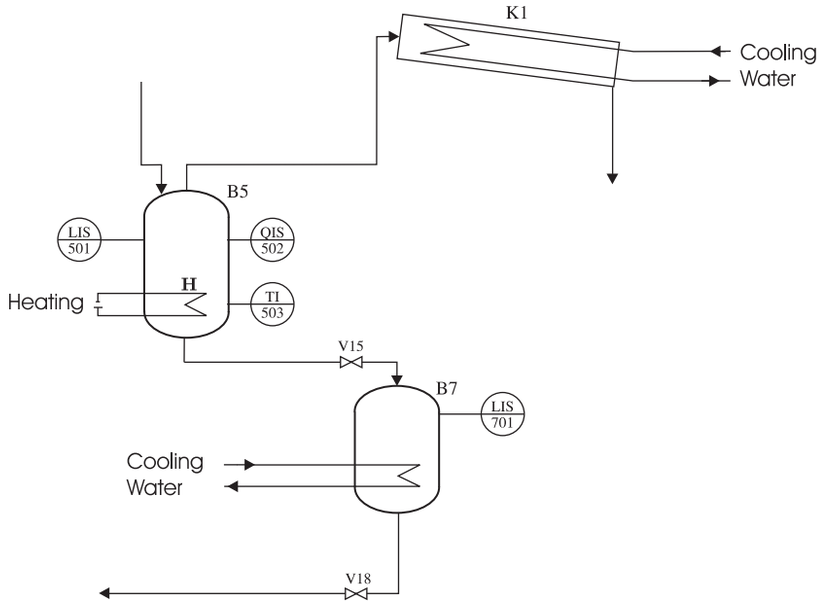

Fig. 6. Flowchart of the benchmark evaporator system.

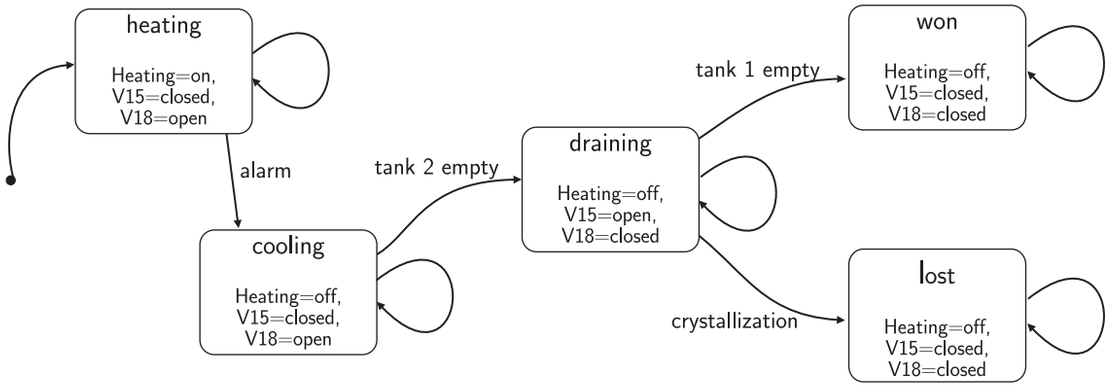

Fig. 7. State transition diagram of the controller.

controller enters the alarm mode and immediately opens valve V18 to empty tank 2. During this phase, the heater is still in the 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 tank. From draining, the controller can either switch to the state won if tank 1 becomes empty, or to lost if the temperature in tank 1 becomes lower than the critical value T_c.

The aim 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 solidifies, (iii) when the valve V15 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.

6. Modeling CS1 in MLD Form

In order to use the verification tools described in Section 4, we need to obtain a hybrid model of the batch evaporator process in MLD form. We consider the model described in [39], which only takes into account the dynamics of the levels h_1, h_2, and of the temperature T of tank 1. The model developed in [39]

Table 3. Continuous dynamics associated with the different states of the FSM in Fig. 7.

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

is summarized in Table 3 (dynamic equations associated with each logical state of the controller of Fig. 7) together with the parameters reported in Appendix B.1, 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 $T \gtrsim 373\,\mathrm{K}$, (iii) average constants replace ranges of physical parameters.

6.1. Continuous Dynamics

The dynamic equations described in Table 3 are nonlinear and continuous-time. As MLD models only deal with hybrid linear discrete-time dynamics, we first approximate the static nonlinearity (square root) with a piecewise linear function, and then sample the resulting system.

The nonlinearity arises from Torricelli's law, which models the liquid flow through the outlet of the tank, namely $\dot{h} = -k\sqrt{h}$ where k depends on the geometry of the tank and outlet pipe, and h is the height of the liquid in the tank.

We partition the operating range of h and approximate the square root by a linear function in three intervals: [0, 0.01], (0.01, h_t] and (h_t, $h_{\max}$], as in Fig. 8. In the first interval [0, 0.01] the approximation is such that h goes instantaneously to zero when the level is lower that 0.01 (consistent with the PLC, which assumes that the tank is empty if $h < 0.01$). In the other two intervals, a straight line interpolates the extreme points. The parameter h_t can be chosen $\approx h_{\max}/2$. Figure 8 shows the square root and its piecewise linear approximation for $h_t = 0.9$ (left plot), and compares the evolution of the original continuous-time system and its sampled ($T_s = 60\,\mathrm{s}$) piecewise linear counterpart (right plot).

The piecewise linear approximation of the nonlinearity can be represented by introducing two logical auxiliary variables l_{i1}, l_{i0}, $i = 1, 2$, defined by

$$l_{i1}(t) \leftrightarrow [h_i(t) \le h_{it}], \quad l_{i0}(t) \leftrightarrow [h_i(t) \le 0.01], \quad (8)$$

where we use the shorthand notation l_{i1} for $[l_{i1} = 1]$, and $\bar{l}_{i1}$ for $[l_{i1} = 0]$. As $h_{it} > 0.01$, the logic relation $\bar{l}_{i1}(t) \vee \bar{l}_{i0}(t) = \mathsf{TRUE}$ holds $\forall t$, $i = 1, 2$. Then, the piecewise linear approximation of the square root is defined by

$$s_{h_i} = \begin{cases} \infty & \text{if } l_{i0}, \\ a_{1h_i} h_i + b_{1h_i} & \text{if } \bar{l}_{i0} \wedge l_{i1} \quad i = 1, 2, \\ a_{2h_i} T + b_{2h_i} & \text{otherwise,} \end{cases} \quad (9)$$

where the constants a_{1h_i}, a_{2h_i}, b_{1h_i}, b_{2h_i} are defined in Appendix B.2.

6.2 A/D

As mentioned above, the level and temperature sensors provide four logic signals: tank 1 empty, tank 2 empty, alarm, crystallization. We associate a 0/1 variable with each signal, namely l_{10}, l_{20} introduced in (8) for the tank levels, and t_1, t_2, for the temperature,

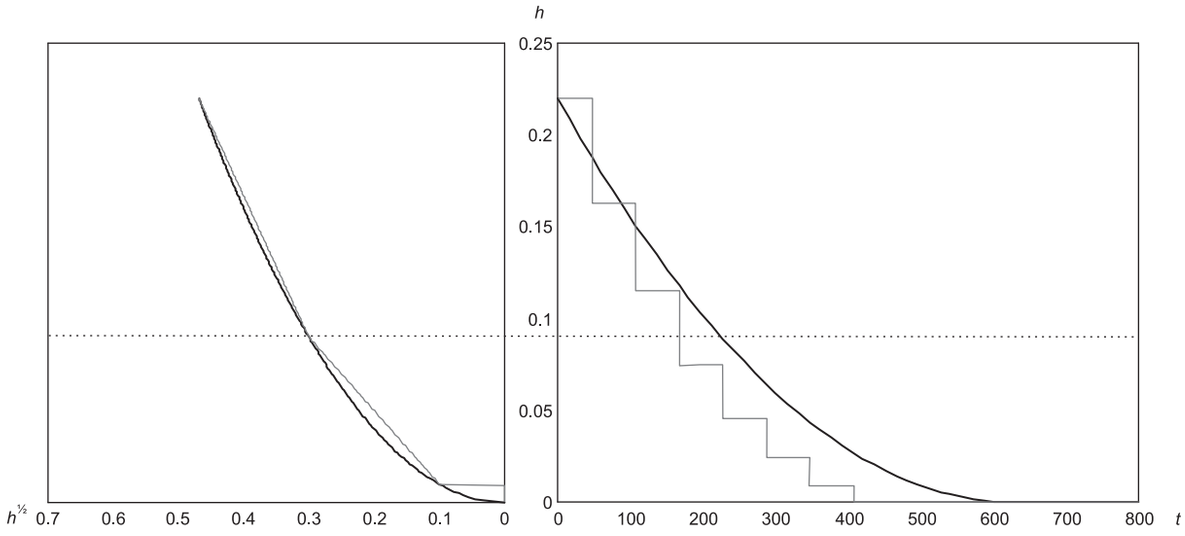

Fig. 8. Piecewise linear approximation of the square root (left), evolution of the approximated sampled system versus the continuous-time nonlinear model (right).

where

$$t_1(t) \leftrightarrow [T(t) \geq T_a], t_2(t) \leftrightarrow [T(t) \leq T_c], \qquad (10)$$

and $T(t)$ is the temperature in tank 1. As $T_a > T_c$, the logic condition $\bar{t}_1(t) \vee \bar{t}_2(t) = \mathsf{TRUE}$ must be satisfied for all t.

6.3. Automaton

The five states of the automaton in Fig. 7 representing the PLC code for exception handling can be represented by the combination of three 0/1 logical states, grouped in the binary vector x_ℓ, according to the convention

$$x_\ell = \begin{bmatrix}0\\0\\0\end{bmatrix} = \mathsf{heating}, \begin{bmatrix}0\\0\\1\end{bmatrix} = \mathsf{cooling}, \begin{bmatrix}0\\1\\0\end{bmatrix} = \mathsf{won}, \quad \begin{bmatrix}0\\1\\1\end{bmatrix} = \mathsf{lost},$$

and

$$\begin{bmatrix}1\\0\\0\end{bmatrix} = \mathsf{draining}$$

As the combinations

$$\begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}$$

are not used, the logic constraint $x_{\ell 1}(t)(x_{\ell 2}(t) \vee x_{\ell 3}(t)) = \mathsf{FALSE}$ must hold $\forall t$. The transition map of the automaton is expressed by 0/1 variables in Table 4. Each update $x_{\ell i}(t+1)$, $i = 1, 2, 3$, can be rewritten as a *sum of products* (SOP)[4]

$$x_{\ell 1}(t+1) = (\bar{x}_{\ell 1}\bar{x}_{\ell 2}x_{\ell 3}l_{20}) \vee (x_{\ell 1}\bar{x}_{\ell 2}\bar{x}_{\ell 3}\bar{t}_{i2}\bar{l}_{10}); \qquad (11)$$

$$x_{\ell 2}(t+1) = (x_{\ell 1}\bar{x}_{\ell 2}\bar{x}_{\ell 3}l_{10}) \vee (x_{\ell 1}\bar{x}_{\ell 2}\bar{x}_{\ell 3}t_2) \vee (\bar{x}_{\ell 1}x_{\ell 2}\bar{x}_{\ell 3}) \vee (\bar{x}_{\ell 1}x_{\ell 2}x_{\ell 3}); \qquad (12)$$

$$x_{\ell 3}(t+1) = (\bar{x}_{\ell 1}\bar{x}_{\ell 2}\bar{x}_{\ell 3}t_1) \vee (\bar{x}_{\ell 1}\bar{x}_{\ell 2}x_{\ell 3}\bar{l}_{20}) \vee (x_{\ell 1}\bar{x}_{\ell 2}\bar{x}_{\ell 3}t_2) \vee (\bar{x}_{\ell 1}x_{\ell 2}x_{\ell 3}), \qquad (13)$$

by simply collecting all the rows of Table 4 where $x_{\ell i}(t+1) = 1$ (in (11)–(13) the temporal index (t) on the right hand side has been omitted for brevity). For instance, the update law for $x_{\ell 1}$ is obtained by collecting the rows where $x_{\ell 1}(t+1)$ is 1 (rows 3 and 5), and rewriting the conditions on lines 3 and 5 as in (11).

The control action produced in each logical state of the controller is

$$h(t) = \bar{x}_{\ell 1}(t)\bar{x}_{\ell 2}(t)\bar{x}_{\ell 3}(t), \qquad (14)$$

$$v_{15}(t) = x_{\ell 1}(t)\bar{x}_{\ell 2}(t)\bar{x}_{\ell 3}(t), \qquad (15)$$

$$v_{18}(t) = (\bar{x}_{\ell 1}(t)\bar{x}_{\ell 2}(t)\bar{x}_{\ell 3}(t)) \vee (\bar{x}_{\ell 1}(t)\bar{x}_{\ell 2}(t)x_{\ell 3}(t)) \qquad (16)$$

and is obtained in a similar way from Table 5.

[4] This is equivalent to the *disjunctive normal form* (DNF).

Table 4. State transition function of the FSM.

$x_{\ell 1}$	$x_{\ell 2}$	$x_{\ell 3}$	t_1	t_2	l_{10}	l_{20}		$x_{\ell 1}(t+1)$	$x_{\ell 2}(t+1)$	$x_{\ell 3}(t+1)$
0	0	0	1	*	*	*	$\rightarrow$	0	0	1
0	0	0	0	*	*	*	$\rightarrow$	0	0	0
0	0	1	*	*	*	1	$\rightarrow$	1	0	0
0	0	1	*	*	*	0	$\rightarrow$	0	0	1
1	0	0	*	*	1	*	$\rightarrow$	0	1	0
1	0	0	*	1	*	*	$\rightarrow$	0	1	1
1	0	0	*	0	0	*	$\rightarrow$	1	0	0
0	1	0	*	*	*	*	$\rightarrow$	0	1	0
0	1	1	*	*	*	*	$\rightarrow$	0	1	1

6.4. Sampling

Each linear continuous dynamics of the piecewise linear approximations is sampled with sampling time $T_s = 60\,\text{s}$. Because of the resulting discrete-time nature of the model, switches can only occur at sampling steps. Although this clearly introduces a model mismatch, the sampling time is short enough to render the approximation negligible, or at least comparable with the other approximations of the physics of the model.

6.5. D/A

The continuous state-update law can be rewritten as the linear set of difference equations

$$T(k+1) = z_T(k), \tag{17}$$

$$h_1(k+1) = z_{h_{11}}(k) + z_{h_{12}}(k) + z_{h_{13}}(k), \tag{18}$$

$$h_2(k+1) = z_{h_{21}}(k) + z_{h_{22}}(k) + z_{h_{23}}(k) + z_{h_{24}}(k), \tag{19}$$

where z_T and $z_{h_{ij}}$ are auxiliary continuous variables defined in Table 6, together with the parameters specified in Appendix B.3.

In summary, the state of the MLD system modeling the batch evaporator process is

$$x_c = \begin{bmatrix} T \\ h_1 \\ h_2 \end{bmatrix} \in \mathbb{R}^3 \quad x_\ell = \begin{bmatrix} x_{\ell 1} \\ x_{\ell 2} \\ x_{\ell 3} \end{bmatrix} \in \{0,1\}^3, \tag{20}$$

$d \in \{0,1\}^{18}$, $z \in \mathbb{R}^8$ and the corresponding MLD and PWA models are automatically obtained by processing the HYSDEL system description reported in Appendix A. The total number of MLD inequalities is 127. The equivalent PWA system has 13 regions.

Table 5. Output function of the automaton.

$x_{\ell 1}(t)$	$x_{\ell 2}(t)$	$x_{\ell 3}(t)$		$h(t)$	$v_{15}(t)$	$v_{18}(t)$
0	0	0	$\rightarrow$	1	0	1
0	0	1	$\rightarrow$	0	0	1
1	0	0	$\rightarrow$	0	1	0
0	1	0	$\rightarrow$	0	0	0
0	1	1	$\rightarrow$	0	0	0

7. Verification Results

We aim to verify that after an exception occurs, the PLC code based on the control logic of Fig. 7 safely shuts down the plant to the won state for any initial condition

$$x(0) = \begin{bmatrix} x_c(0) \\ x_\ell(0) \end{bmatrix} \in \mathcal{X}_0 = \left\{ T, h_1, h_2, x_\ell : T = 373, 0.2 \leq h_1 \leq 0.22, 0.28 \leq h_2 \leq 0.3, x_\ell = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right\}.$$

To this end, we apply the verification algorithm presented in Section 4, and label as target set $\mathcal{Z}_1$ the set of *safe* states

$$\left\{ x : x_\ell = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right\}$$

(won), and as target set $\mathcal{Z}_2$ the set of *unsafe* states

$$\left\{ x : x_\ell = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \right\}$$

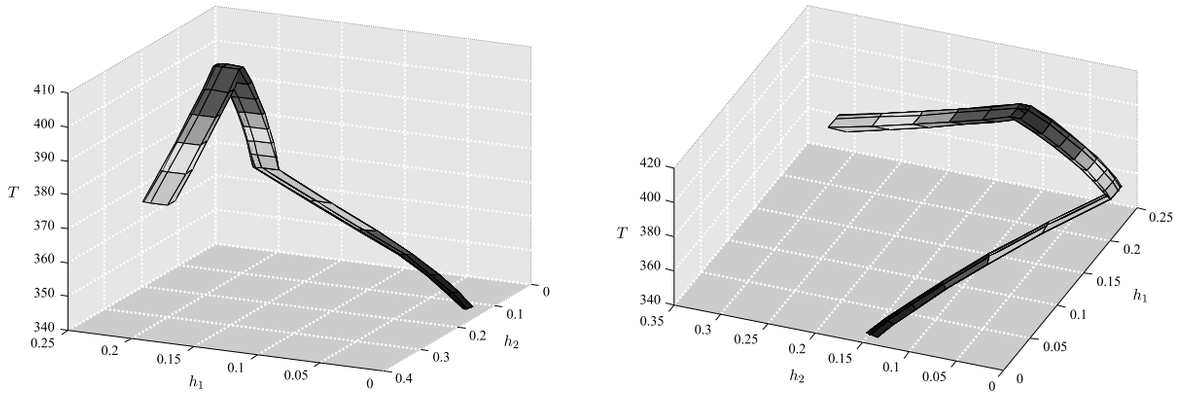

Fig. 9. Set evolution from $\mathcal{X}_0$ to the target set $\mathcal{Z}_1$ (won) for $T_a = 391$ (same evolution, different perspectives).

Table 6. D/A converters.

Symbol	Definition
z_T	$\begin{cases} a_{T1}T + b_{T1} & \text{if } h \\ a_{T2}T + b_{T2} & \text{otherwise} \end{cases}$
z_{h11}	$\begin{cases} a_{h_{11}}h_1 + b_{h_{11}} & \text{if } \bar{h}\bar{v}_{18}\bar{l}_{10}h_{1t} \\ 0 & \text{otherwise} \end{cases}$
z_{h12}	$\begin{cases} a_{h_{12}}h_1 + b_{h_{12}} & \text{if } \bar{h}\bar{v}_{18}\bar{l}_{10}h_{1t} \\ 0 & \text{otherwise} \end{cases}$
z_{h13}	$\begin{cases} a_{h_{13}}h_1 + b_{h_{13}} & \text{if } (h \vee v_{18})\bar{l}_{10} \\ 0 & \text{otherwise} \end{cases}$
z_{h21}	$\begin{cases} a_{h_{21}}h_1 + h_2 + b_{h_{21}} & \text{if } \bar{h}\bar{v}_{18}h_{2t} \\ 0 & \text{otherwise} \end{cases}$
z_{h22}	$\begin{cases} a_{h_{22}}h_2 + b_{h_{22}} & \text{if } (h \vee v_{18})\bar{l}_{20}h_{2t} \\ 0 & \text{otherwise} \end{cases}$
z_{h23}	$\begin{cases} a_{h_{23}}h_1 + h_2 + b_{h_{23}} & \text{if } \bar{h}\bar{v}_{18}\bar{l}_{20}\bar{h}_{2t} \\ 0 & \text{otherwise} \end{cases}$
z_{h24}	$\begin{cases} a_{h_{24}}h_2 + b_{h_{24}} & \text{if } (h \vee v_{18})\bar{l}_{20}\bar{h}_{2t} \\ 0 & \text{otherwise} \end{cases}$

(lost). The results of the algorithm are plotted in Fig. 9, 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. Algorithm 4.1 was executed in 79 s on a PC Pentium II running interpreted Matlab code.

The tool can also easily perform parametric verification if the vector of parameters θ enters the model linearly, and its range is a polyhedral set Θ (e.g. Θ is an interval). Constant parameters can in fact be taken into account by augmenting the state vector to $\begin{bmatrix} x \\ \theta \end{bmatrix}$, adding constant dynamics $\theta(t+1) = \theta(t)$ for the additional state θ, 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 variations of the alarm temperature T_a in the range $383\,\text{K} < T_a \leq 393\,\text{K}$. 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 temperatures in $\mathcal{X}_0$. The parametric verification requires 82 s of CPU time.

8. Conclusions

In this paper we have proposed a modeling framework for hybrid systems described by the interaction of discrete-time linear dynamic equations, logic rules, and automata, and a general algorithm for safety analysis based on LP and MILP. The versatility and the computation feasibility of the approach have been tested on the batch evaporator benchmark process.

Acknowledgements

The authors thank the partners of the Esprit Project 26270 for stimulating discussions, and in particular Stefan Kowalewski and Olav Stursberg for providing useful information about the batch evaporator example, and Domenico Mignone for fruitful discussions. This research was supported by the Swiss National Science Foundation and the Esprit Project 26270. (Hybrid Systems).

References

1. Alur R, Courcoubetis C, Henzinger TA, Ho P-H. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In: Ravn AP, Grossman A, Nerode A, Rischel H

(ed). Hybrid Systems, vol 736 of Lecture notes in computer science, Springer-Verlag, 1993, pp 209–229
2. Antsaklis PJ. A brief introduction to the theory and applications of hybrid systems. In: Proceedings of the IEEE, Special Issue on hybrid systems: Theory and applications, 88(7) July 2000, pp 879–886
3. Asarin A, Maler O, Pnueli A. On the analysis of dynamical systems having piecewise-constant derivatives. Theor Computer Sci 1995; 138: 35–65
4. Asarin E, Bournez O, Dang T, Maler O. Reachability analysis of piecewise-linear dynamical systems. In: Krogh B, Lynch N (eds). Hybrid Systems: Computation and Control, vol 1790 of Lecture notes in computer science, Springer-Verlag, 2000, pp 20–31
5. Asarin E, Bozga M, Kerbrat A, Maler O, Pnueli A, Rasse A. Data-structures for the verification of timed automata. In: Maler O (ed). Hybrid and real-time systems, vol 1201 of Lecture notes in computer science, Springer-Verlag, 1997, pp 346–360
6. Balluchi A, Benvenuti L, Di Benedetto M, Pinello C, Sangiovanni-Vincentelli A. Automotive engine control and hybrid systems: Challenges and opportunities. Proceedings of the IEEE, 88(7) July 2000, 888–912
7. Bemporad A, Ferrari-Trecate G, Morari M. Observability and controllability of piecewise affine and hybrid systems. IEEE Trans Autom Control 2000; 45(10): 1864–1876
8. Bemporad A, Giovanardi L, Torrisi FD. Performance driven reachability analysis for optimal scheduling and control of hybrid systems. In: Proceedings of the 39th IEEE Conference on decision and dontrol, Sydney, Australia. December 2000
9. Bemporad A, Morari M. Control of systems integrating logic, dynamics, and constraints. Automatica 1999; 35(3): 407–427
10. Bemporad A, Morari M, Dua V, Pistikopoulos EN. The explicit linear quadratic regulator for constrained systems. In: Proceedings of the American contr Conference, 2000
11. Bemporad A, Torrisi FD. Inner and outer approximation of polytopes using hyper-rectangles. Technical Report AUT00-02, Automatic Control Lab, ETH Zurich. 2000
12. Bemporad A, Torrisi FD. Morari M. Optimization-based verification and stability characterization of piecewise affine and hybrid systems. In: Krogh B, Lynch N (eds), Hybrid systems: computation and control, vol 1790 of Lecture notes in computer science, Springer-Verlag, 2000, pp 45–58
13. Bemporad A, Torrisi FD, Morari M. Performance analysis of piecewise linear systems and model predictive control systems. In: Proceedings of the 39th IEEE Conference on decision and control, Sydney, Australia. December 2000
14. Bengtsson J, Larsen KG, Larsson F, Pettersson P, Wang Y, Weise C. New Generation of UPPAAL. In: International Workshop on Software Tools for Technology Transfer, June 1998
15. Blondel VD, Tsitsiklis JN. Complexity of stability and controllability of elementary hybrid systems. Automatica 1999; 35: 479–489
16. Branicky MS. Studies in hybrid systems: Modeling, analysis and control. PhD thesis, LIDS-TH 2304, Massachusetts Institute of Technology, Cambridge, MA, 1995
17. Branicky MS. Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. IEEE Trans Autom Control 1998; 43(4): 475–482
18. Branicky MS, Borkar VS, Mitter SK. A unified framework for hybrid control: Model and optimal control theory. IEEE Trans Autom Control 1998; 43(1): 31–45
19. Cavalier TM, Pardalos PM, Soyster AL. Modeling and integer programming techniques applied to propositional calculus. Computers Operations Research 1990; 17(6): 561–570
20. Chutinan A, Krogh BH. Verification of polyhedral invariant hybrid automata using polygonal flow pipe approximations. In: Vaandrager F, van Schuppen J, (eds). Hybrid systems: Computation and control, vol 1569 of Lecture notes in computer science, Springer-Verlag 1999, pp 76–90
21. Heemels WPMH, De Schutter B, Bemporad A. Equivalence of hybrid dynamical models. Automatica (to appear).
22. Henzinger TA, Ho P-H, Wong-Toi H. HYTECH: A model checker for hybrid systems. Software tools for technology transfer 1997; 1: 110–122
23. Huuck R. Verifying timing aspects of VHS case study 1. Technical Report http://www-verimag.imag.fr/VHS/year1/cs11i.ps, Christian-Albrechts-Universität zu Kiel, 1999
24. Johannson M, Rantzer A. Computation of piece-wise quadratic Lyapunov functions for hybrid systems. IEEE Trans Autom Control 1998; 43(4): 555–559
25. Johansson KH, Egerstedt M, Lygeros J, Sastry S. On the regularization of Zeno hybrid automata. System & Control Letters 1999; 38: 141–150
26. Kesten Y, Pnueli A, Sifakis J, Yovine S. Integration graphs: A class of decidable hybrid systems. In: Grossman RL, Nerode A, Ravn AP, Rischel H (eds). Hybrid systems, vol 736 of lecture notes in computer science, Springer-Verlag, 1993, pp 179–208
27. Kfoury AJ, Moll RN, Arbib MA. A programming approach to computability. 3rd edn. Springer-Verlag, 1982
28. Kowalewski S. Description of VHS case study 1. Experimental batch plant, http://astwww.chemietechnik.uni-dortmund.de/~vhs/cs1descr.zip. Draft. University of Dortmund, Germany, July 1998
29. Kowalewski S, Stursberg O. The batch evaporator: A benchmark example for safety analysis of processing systems under logic control. In: 4th International Workshop on discrete event systems (WODES98), Cagliaro, Italy 1998
30. Kristoffersen K, Larsen K, Petterson P, Weise C. Experimental batch plant – VHS case study 1 using time automata and UPPAAL. http://www-verimag.imag.fr/VHS/year1/cs11f.ps, May 1999
31. Lafferiere G, Pappas G, Yovine S. A new class of decidable hybrid systems. In: Vaandrager FW, van Schuppen JH (eds). Hybrid systems: Computation and Control, vol 1569 of Lecture notes in computer science, Springer-Verlag, 1999, pp 137–151
32. Liberzon D, Morse AS. Basic problems in stability and design of switched systems. IEEE Control Systems Magazine 1999; 19(5): 59–70
33. Lygeros J, Godbole DN, Sastry S. A game theoretic approach to hybrid system design. In: Alur R, Henzinger T (eds). Hybrid systems III, vol 1066 of

Lecture notes in computer science, Springer-Verlag, 1996, pp 1–12
34. Lygeros J. Tomlin C, Sastry S. Controllers for reachability specifications for hybrid systems. Automatica 1999; 35(3): 349–370
35. Preussig J, Stursberg O, Kowalewski S. Reachability analysis of a class of switched continuous systems by integrating rectangular approximation and rectangular analysis. In: Vanndrager, van Schuppen J (eds). Hybrid systems: Computation and control, vol 1569 of Lecture notes in computer science, Springer-Verlag, 1999, pp 210–222
36. Raman R, Grossmann IE. Relation between MILP modeling and logical inference for chemical process synthesis. Computers Chem Eng 1991; 15(2): 73–84
37. Sontag ED. Nonlinear regulation: The piecewise linear approach. IEEE Trans Autom Control April 1981; 26(2); 346–358
38. Sontag ED. Interconnected automata and linear systems: A theoretical framework in discrete-time. In: Alur R, Henzinger TA, Sontag ED (eds). Hybrid systems III – Verification and control, number 1066 in Lecture notes in computer science, Spinger-Verlag, 1996, pp 436–448
39. Stursberg O. The batch-evaporator benchmark example – A simplified formulation. http;//astwww.chemie-techink.uni-dortmund.de/~olaf/, September 1999
40. Torrisi FD, Bemporad A, Mignone D. HYSDEL – A language for describing hybrid systems. Technical Report AUT00-03, ETH Zurich, 2000. http://control.ethz.ch/~hybrid/hysdel
41. Tyler ML, Morari M. Propositional logic in control and monitoring problems. Automatica 1999; 35(4): 565–583
42. Vidal R, Schaffert S, Lygeros J, Sastry S. Controlled invariance of discrete time systems. In: Krogh B, Lynch N (eds). Hybrid systems: Computation and control, vol 1790 of Lecture notes in computer science Springer-Verlag, 2000, pp 437–450
43. Williams HP. Logical problems and integer programming. Bulletin of the Institute of Mathematics and Its Applications 1977; 13: 18–20
44. Williams HP. Model building in mathematical programming. 3rd edn. John Wiley & Sons, 1993

Appendix A. HYSDEL Code for the Batch Evaporator Process

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

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

(other parameters are omitted for brevity, see Appendix B)

```
    }
} IMPLEMENTATION {
    AUX {
        REAL zT, zh1a, zh1b, zh1c, zh2a, zh2b, zh2c, zh2d;
        BOOL ti1, ti2, l1, l2, h, v18, l1h1, l1h2; }
    LOGIC {
        h = ~p1 & ~p2 & ~p3;
        v18 = ~p1 & ~p2 & ~p3 | ~p1 & ~p2 & p3; }
    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 (~h & ~v18 & ~l1 & l1h1) THEN ah1a * h1 + bh1a
                     [ah1a * hmax  + bh1a, ah1a * hmin + bh1a, 0] };
        zh1b = { IF (~h & ~v18 & ~l1 & ~l1h1) THEN ah1b * h1 + bh1b
                 [ah1b * hmax + bh1b, ah1b * hmin + bh1b, 0] };
        zh1c = { IF (~(h | v18)&~l1) THEN h1
                 [hmax, hmin, 0]};
        zh2a = { IF (~h & ~v18 & ~l2 & l1h2) THEN ahh2a * h1 + h2 + bhh2a
                 [ahh2a * hmax + hmax + bhh2a, ahh2a * hmin + hmin + bhh2a, 0] };
        zh2b = { IF (~(h | v18) & ~l2 & l1h2) THEN ah2a * h2 + bh2a
                 [ah2a * hmax + bh2a, ah2a * hmin + bh2a, 0] };
        zh2c = { IF (~h & ~v18 & ~l2 & ~l1h2) THEN ahh2b * h1 + h2 + bhh2b
                 [ahh2b * hmax + hmax + bhh2b, ahh2b * hmin + hmin + bhh2b, 0] };
        zh2d = { IF (~(h | v18) & ~l2 & ~l1h2) 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 the combination ti1,ti2=11 */
            ~(p1 & (p2 | p3)); /* Excludes logic states 101,110,111 */
        }
    }
    }
```

Appendix B. Parameters

B.1. Model Parameters

$q = 5000\,\mathrm{kW}$, $\mathrm{t_e} = 283\,\mathrm{K}$, $k_1 = 2.2 \cdot 10^{-3}\,\mathrm{m^{(1/2)}s^{-1}}$,
$k_2 = 1.5 \cdot 10^{-3}\,\mathrm{m^{(1/2)}s^{-1}}$, $k_3 = 3.7 \cdot 10^{-6}\,\mathrm{K\ kW^{-1}}$, $k_4 = 24\,\mathrm{K\,kW^{-1}}$,
$k_5 = 8.9 \cdot 10^{-4}\,\mathrm{s^{-1}}$, $T_c = 338\,\mathrm{K}$.

B.2. Linearization Parameters

$h_{1t} = 0.09\,\mathrm{m}$, $h_{1\max} = 0.22\,\mathrm{m}$, $a_{1h_1} = 2.5$, $b_{1h_1} = 0.057\,\mathrm{m}$,
$a_{2h_1} = 1.3003$, $b_{2h_1} = 0.1830\,\mathrm{m}$, $h_{2t} = 0.15\,\mathrm{m}$, $h_{2\max} = 0.3\,\mathrm{m}$,
$a_{1h_2} = 2.0521$, $b_{1h_2} = 0.0795\,\mathrm{m}$, $a_{2h_2} = 1.0695$, $b_{2h_2} = 0.2269\,\mathrm{m}$.

B.3. Sampled Dynamics

$a_{T1} = 0.9481$, $b_{T1} = 25.4931\,\mathrm{K}$, $a_{T2} = 0.9480$, $b_{T2} = 14.7158\,\mathrm{K}$,
$a_{h_{11}} = 0.7189$, $b_{h_{11}} = -0.0084\,\mathrm{m}$, $a_{h_{12}} = 0.8423$, $b_{h_{12}} = -0.0222\,\mathrm{m}$,
$a_{h_{13}} = 1$, $b_{h_{13}} = 0\,\mathrm{m}$, $a_{h_{21}} = 0.1916$, $b_{h_{21}} = 0.0057\,\mathrm{m}$,
$a_{h_{22}} = 0.8314$, $b_{h_{22}} = -0.0065\,\mathrm{m}$, $a_{h_{23}} = 0.1075$, $b_{h_{23}} = 0.0151\,\mathrm{m}$,
$a_{h_{24}} = 0.9082$, $b_{h_{24}} = -0.0195\,\mathrm{m}$.