Modeling and Safety Verification of Discrete/ Continuous Processing Systems

V. D. Dimitriadis, N. Shah, and C. C. Pantelides
Centre for Process Systems Engineering, Imperial College of Science, Technology and Medicine, London SW7 2BY, U.K.

A quantitative, model-based approach to the safety verification problem for general processing systems operating in the discrete time domain is presented. It is recognized that the operation of most of these systems involves both discrete and continuous characteristics. Therefore, an appropriate modeling framework is proposed, within which models of purely discrete, purely continuous and hybrid systems of arbitrary complexity can be constructed consistently. The models developed can then be incorporated into a safety verification formulation, which allows the identification of potential hazards that may occur while operating such systems, together with the combinations of events that lead to them. Apart from the dynamic process model, the data required for carrying out the analysis include the space of possible disturbances and the set of operating regimes that are considered to be unsafe or undesirable from the operability point of view. The formulation results in a mixed-integer optimization problem. A number of simple example problems presented illustrate the main ideas of the proposed technique, and the solution of an industrial-scale case study demonstrates its applicability.

Introduction

The problem of ensuring the safety of chemical processes is usually addressed in three steps:

1. *Hazard Identification.* The process is examined to determine if there is any combination of events that can lead to a dangerous situation or compromise normal operation.
2. *Hazard Evaluation.* Each of the identified hazards is examined in detail to determine the probability of its occurrence and to quantify its likely consequences on the process and its surroundings in general.
3. *Hazard Limitation.* If the risk is considered unacceptable, action is taken in order to reduce the probability of the hazard occurring and/or to limit its consequences.

The first step in this procedure is clearly critical. In order to evaluate a hazard and find ways of avoiding it, it has to be identified in the first place. Unfortunately, there have been situations where a hazard was not identified until after the occurrence of an accident (Kletz, 1985). Therefore, the identification methodology should be thorough enough not to overlook any possible dangers, no matter how improbable they might appear to be.

Correspondence concerning this article should be addressed to N. Shah.

For many years, engineers have relied on qualitative techniques for identifying potential process hazards (Crowl and Louvar, 1990). *Checklists* are often used for this purpose. The designer or operator consults a list of potential problems and examines the affected areas in the process, making any necessary modifications to eliminate dangers. Checklists are satisfactory when there is little or no innovation in the process and the hazards have been met before. However, they are of little use when the design is new and no prior experience on it exists. Additionally, they are usually difficult to construct from scratch and may encourage a narrow-minded approach according to which any items that are not on the list are not examined at all.

A more creative approach is followed during a *hazards and operability study* (HAZOP) (Kletz, 1992). Here, a study is undertaken by a group of people who are directly involved in the design, construction, operation, and maintenance of the process under examination. A number of "guide words" (e.g., NONE, MORE OF, LESS OF, etc.) are applied systematically to each connecting pipeline and piece of equipment in the process flow diagram, considering possible deviations from normal operating conditions (e.g., MORE temperature,

LESS pressure, etc.). All causes and consequences of such deviations are identified and measures are taken to prevent possible problems. HAZOPs represent a systematic way of addressing the problem of hazard identification and have been used with success in the process industries for some time. There has also been interest concerning the automation of the method, mostly using knowledge-based expert systems (Venkatasubramanian and Vaidhyanathan, 1994). However, the nature of the method is still inherently qualitative and much expertise on the process under investigation is required. Furthermore, the complete application of the procedure may be problematic for large processes and tends to be used only for very hazardous operations involving larger units. Finally, this technique has been criticized for "goldplating" chemical installations (i.e., resulting in the installation of a complicated network of protective equipment designed to avoid very unlikely hazards), especially when not used in combination with a rigorous hazard evaluation method (Crowl and Louvar, 1990; Kletz, 1992).

Other methods, such as *hazard surveys, safety reviews*, and *safety audits* have also been applied to the problem of hazard identification. In general, all of these techniques are based on engineering judgment and intuition, and require a substantial degree of expertise on the specific process examined. They are easy to understand and follow, and usually succeed in identifying the most obvious hazards. They are also appropriate when no detailed model of the process under investigation exists or existing models are highly uncertain. Their main disadvantage is that they rely on a qualitative model for the system under examination and thus fail to quantify the identified hazards. Moreover, for processes with many units, it is very difficult to track down the consequences of an event as it propagates throughout the process. This is very serious because an initiating event might appear of little interest in itself and still have severe consequences at other stages of the process. Finally, the methods just described very often fail to provide direct indications as to what modifications should be made in order to make the process safer. This is left for the engineers to decide using their knowledge on similar processes and safety in general.

Dynamic process simulation, both qualitative (Waters and Ponton, 1989; Catino and Ungar, 1995) and quantitative (Park and Barton, 1994), has also been used to investigate the performance of chemical processes with respect to safety. Typically, a number of simulations are performed for a set of —what are usually perceived as "worst-case"—test inputs. If the behavior of the system satisfies the safety specifications, the designer can have some confidence that the system is safe or error-free. However, exhaustive examination of all possible inputs is rarely feasible and thus, although simulations are helpful, they cannot be used as a rigorous safety verification tool.

Formal verification methods have been extensively studied by researchers in the field of computer science. A number of modeling frameworks and analysis methods have been proposed to aid in the verification of safety-critical computer software and hardware. Due to the inherently discrete nature of these systems, early attempts in this area have concentrated on purely discrete systems (Cassandras, 1993).

Some of these discrete analysis methods have been applied to chemical processing systems. Moon et al. (1992) proposed a verification procedure for discrete event dynamic systems (DEDS). They use a finite state machine (FSM) representation to model DEDS, temporal logic statements to represent specifications imposed on their operation, and the model checking verification (MCV) method of Clarke et al. (1986) to test if given relationships between the model and the specifications hold. The method has been successfully applied to the verification of programmable logic controllers (PLC) (Moon, 1992) and control procedures for batch plants (Moon and Macchietto, 1994). Its application to chemical processes with continuous characteristics has also been reported (Moon, 1992; Probst and Powers, 1994). In the latter case, the continuous variables are discretized in order to construct the FSM representation, and thus the number of system states increases significantly, making the subsequent analysis computationally intensive.

However, even if the automated control system is purely discrete, the continuous time element and the continuous characteristics of the underlying process give rise to hybrid behavior. Moreover, modern control systems frequently exhibit inherent hybrid characteristics (e.g., multimode continuous controllers, etc.). This motivated the development of modeling frameworks (e.g., timed automata, timed Petri nets, hybrid automata, etc.) and verification methods for hybrid systems (Grossman and Nerode, 1993).

Among these, the *hybrid automata* modeling framework (Alur et al., 1993; Nicollin et al., 1993; Alur et al., 1995) is both the most recent and the most general. A hybrid automaton (HA) is essentially an FSM where each state is characterized additionally by a set of variables and a set of equations describing the system when in that state. Transitions between states are triggered either by actions or when certain invariant conditions associated with each state are satisfied. Each transition is labeled with a guarded command to be executed when the transition takes place. The verification problem for HA is treated as a reachability problem and is solved by computing the forward and backward reachable regions of the automaton through an iterative semidecision procedure. Although the modeling framework can, in principle, be generalized to include nonlinear systems, the same is not true for the verification method. Furthermore, even for linear systems, no theoretical guarantee can be given that the verification algorithm will terminate and, more importantly, the state explosion problem encountered in finite-state based methods reoccurs in the form of computationally expensive infinite state space searches.

The objective of this article is to describe a quantitative, model-based approach to the safety verification problem for general processing systems operating in the discrete time domain. It is recognized that the operation of most of these systems involves both discrete and continuous characteristics and an appropriate *modeling framework* is proposed to describe it. The system representation used closely resembles an HA and can be readily transformed into a discrete/continuous mathematical model for systems of arbitrary complexity. The models developed can then be used for the purposes of *safety verification*, which is posed as a mixed-integer optimization problem.

We begin by describing the mixed discrete/continuous modeling framework and showing how it can be used to construct mathematical models of purely discrete, purely contin-

uous, and hybrid processing systems in a consistent way. Then, the basic concepts on which our approach to safety verification is based are introduced and the problem is formulated as a mixed-integer optimization problem. The use of the proposed techniques is illustrated using a number of example problems as well as a larger-scale industrial case study taken from the literature. We conclude with some remarks on the applicability and limitations of this work.

Modeling of Processing Systems in the Discrete Time Domain

The operation of most processing systems exhibits both continuous and significant discrete characteristics, the latter typically arising both from discontinuities in the physical behavior of the process (e.g., phase transitions and irregularities in the system geometry) and/or from discrete external actions imposed on the process (e.g., batch operations and digital control). Therefore, any model-based method aimed at the design and/or analysis of such systems has to make use of models that can capture this hybrid nature in a consistent way.

Recently, Barton and Pantelides (1994) proposed a general modeling framework for discrete/continuous processing systems operating in the continuous time domain. In the present article, this framework is modified to describe processes operating in the discrete time domain, and extended to support the development of model-based applications other than detailed dynamic simulation. As will be seen later, safety verification can be viewed as one such application.

A mathematical model used to describe a dynamic system comprises a set of describing variables and a set of equations that relate these variables. At any given time, t, the composition of these two sets will characterize the current *state* of the system model. Many models have a unique state, that is, the composition of the variable and equation sets remains unchanged throughout the operation of the system. On the other hand, discontinuities in the behavior of a system will, in general, result in changes to either or both of these sets, giving rise to different states for the corresponding model. A transition from one state to another is triggered by logical conditions involving the variables that describe the system in the former state.

For example, consider the storage tank depicted in Figure 1a. The dynamic mass balance equation that describes the behavior of this system can be written in a discrete time domain as follows:

$$V_{t+1} = \begin{cases} V_t + F_t^{\text{in}} - F_t^{\text{out}}, & \text{if } V_t + F_t^{\text{in}} - F_t^{\text{out}} > 0 \\ 0, & \text{if } V_t + F_t^{\text{in}} - F_t^{\text{out}} \le 0. \end{cases}$$

The discontinuity in the system behavior is caused by the fact that the volume of liquid in the tank is a nonnegative quantity and results in two different states for the model, as shown in Figure 1b. In this case, the same set of variables describes both states, but the corresponding sets of equations differ.

States and transitions

To formalize the preceding ideas, consider a dynamic system operating in a discrete time domain $t = 0 \ldots \mathcal{H}$, where

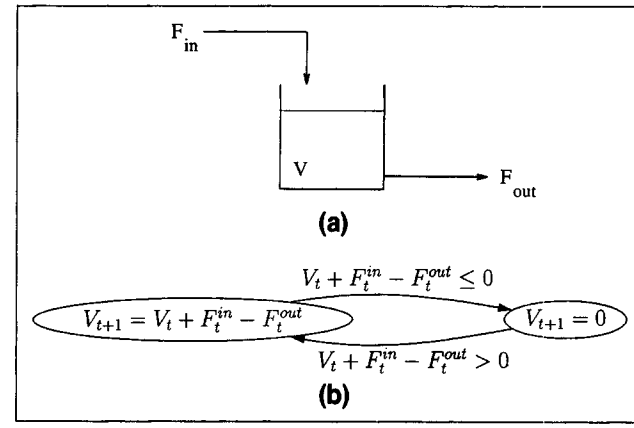

Figure 1. Storage tank: (a) system; (b) modeling representation.

$\mathcal{H}$ is the time horizon considered. At any given time, t, the system is subject to external inputs $\boldsymbol{u}_t \in \mathfrak{U}$, while a set of output variables $\boldsymbol{z}_t$ convey information about the system to its environment (see Figure 2). Both $\boldsymbol{u}_t$ and $\boldsymbol{z}_t$ may be either continuous or discrete valued or a combination thereof.

State Characterization. The system can exist in one of a finite number of distinct states. Each of the states, $s \in \mathcal{S}$, is characterized by:

1. A set of continuous *variables* $\boldsymbol{x}_t^{(s)} \in \mathfrak{X}^{(s)}$ and $\boldsymbol{y}_t^{(s)} \in \mathfrak{Y}^{(s)}$, that are used to describe the system when it is in that state.
2. A set of *equations*

$$\boldsymbol{f}^{(s)}(\boldsymbol{x}_{t+1}^{(s)}, \boldsymbol{x}_t^{(s)}, \boldsymbol{y}_t^{(s)}, \boldsymbol{u}_t) = 0 \tag{1}$$

that determine the behavior of the system when it is in state s. It is assumed that $\boldsymbol{f}^{(s)}(\cdot) = 0$ determine unique values of $\boldsymbol{x}_{t+1}^{(s)}$ and $\boldsymbol{y}_t^{(s)}$ for any given values $\boldsymbol{x}_t^{(x)} \in \mathfrak{X}^{(s)}$ and $\boldsymbol{u}_t \in \mathfrak{U}$.

3. A (possibly empty) set of *transitions* to other states, $\mathfrak{I}_s$.

Transition Characterization. Each of the transitions comprises:

1. An initial state, s.
2. A final state, s'.
3. A logical condition, involving the describing variables in s, that must be satisfied for the transition to occur.
4. A set of relationships

$$\boldsymbol{I}^{(ss')}(\boldsymbol{x}_{t+1}^{(s')}, \boldsymbol{y}_{t+1}^{(s')}, \boldsymbol{x}_t^{(s)}, \boldsymbol{y}_t^{(s)}, \boldsymbol{u}_t) = 0 \tag{2}$$

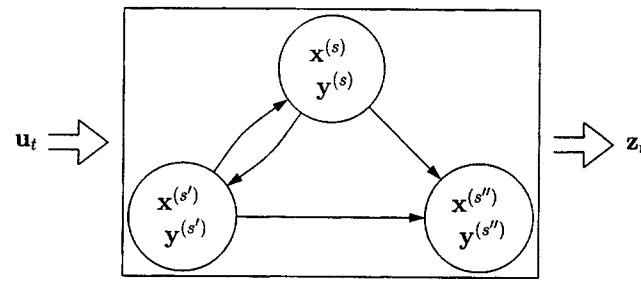

Figure 2. System states, transitions, and describing variables.

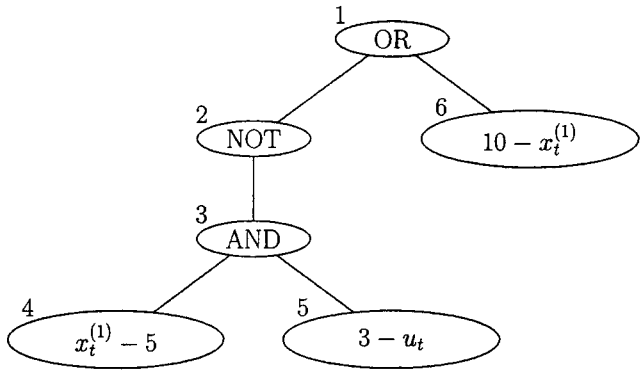

Figure 3. Tree representation for logical conditions.

that allow the condition of the system in state s' at time $t+1$ to be determined from the condition of the system in state s at time t. More specifically, it is assumed that the system of equations $f^{(s')}(x_{t+2}^{(s')}, x_{t+1}^{(s')}, y_{t+1}^{(s')}, u_{t+1}) = 0$ and $I^{(ss')}(\cdot) = 0$ determine unique values for $x_{t+2}^{(s')}$, $x_{t+1}^{(s')}$ and $y_{t+1}^{(s')}$ for any set of values of $x_t^{(s)} \in \mathfrak{X}^{(s)}$, $y_t^{(s)} \in \mathfrak{Y}^{(s)}$, $u_t \in \mathfrak{U}$, and $u_{t+1} \in \mathfrak{U}$.

Representation of General Logical Conditions. We note that the logical condition associated with a given transition $s \to s'$ can be of arbitrary complexity. For the purposes of the mathematical formulation presented in the next section, we need a formal definition of any such condition. In particular, we choose to describe each logical condition in terms of a tree, each node of which has at most two descendants; an example of such a tree is shown in Figure 3. The set of leaf nodes (i.e., those with no descendants) of the tree associated with the transition $s \to s'$ are denoted by $LF^{(ss')}$. Each node $k \in LF^{(ss')}$ is characterized by an algebraic expression

$$l_k^{(ss')}(x_t^{(s)}, y_t^{(s)}, u_t), \qquad k \in LF^{(ss')}.$$

The node is considered to have a value TRUE if

$$l_k^{(ss')}(x_t^{(s)}, y_t^{(s)}, u_t) \le 0,$$

otherwise it is FALSE.

Each of the nonleaf nodes of the tree is characterized by a logical operator. In particular, we denote the set of nonleaf nodes described by the binary disjunctive (OR) and conjunctive (AND) operators by $DJ^{(ss')}$ and $CJ^{(ss')}$. The left and right descendent nodes of such a binary node, k, are denoted as left(k) and right(k), respectively. Finally, the set of nonleaf nodes described by the unary negation operator (NOT) is denoted by $NG^{(ss')}$; the descendant node of a node $k \in NG^{(ss')}$ is denoted by desc(k).

The value of a nonleaf node is obtained by applying its logical operator to the value(s) of its descendent(s). The value of the logical expression described by the entire tree is simply the value of the root node.

As an illustration, consider the example shown in Figure 3 describing the logical condition associated with a transition from state 1 to state 2 in a given model. It can be verified that the tree corresponds to the logical expression

$$\text{NOT } [(x_t^{(1)} \le 5) \text{ AND } (u_t \ge 3)] \text{ OR } (x_t^{(1)} \ge 10).$$

The various sets of nodes just defined are as follows:

$$LF^{(12)} \equiv \{4,5,6\}, \qquad DJ^{(12)} \equiv \{1\}, \qquad CJ^{(12)} \equiv \{3\},$$
$$NG^{(12)} \equiv \{2\}.$$

The value of the three leaf nodes is associated with the elementary logical expressions:

$$l_4^{(12)} \equiv x_t^{(1)} - 5 \le 0$$
$$l_5^{(12)} \equiv 3 - u_t \le 0$$
$$l_6^{(12)} \equiv 10 - x_t^{(1)} \le 0.$$

The structure of the tree is expressed in terms of the descendent relations:

left(1) = 2; right(1) = 6; desc(2) = 3; left(3) = 4; right(3) = 5.

System model

Given the preceding representation for a system, we proceed to construct a discrete/continuous mathematical model that expresses its behavior in terms of algebraic equality and inequality constraints. The example shown in Figure 4 will be used throughout the rest of this section to illustrate better this procedure. This describes a hybrid system operating in three distinct states ($s = 1, 2, 3$) subject to a single external input u_t. The equations describing the system behavior in each state are shown in the corresponding node in Figure 4. The information shown against each transition $s \to s'$ is in the form:

$$\left| \begin{array}{c} \text{Logical condition triggering transition} \\ I^{(ss')}(\cdot) = 0. \end{array} \right|$$

For instance, the transition from state 1 to state 2 is triggered at time t if:

$$\text{NOT } [(x_t^{(1)} \le 5) \text{ AND } (u_t \ge 3)] \text{ OR } (x_t^{(1)} \ge 10).$$

In such a case, the system finds itself in state 2 at time $t+1$, with the value of the variable $x_{t+1}^{(2)}$ equal to the value $x_t^{(1)}$. It can be verified that this information, together with the equa-

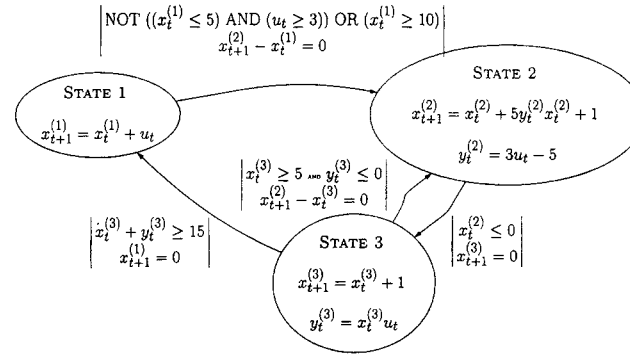

Figure 4. Example hybrid system.

tions describing the system behavior in state 2 and the value of the external input u_{t+1} also allow the variable values $y_{t+1}^{(2)}$ and $x_{t+2}^{(2)}$ to be determined uniquely.

For the purposes of the mathematical formulation presented in this section, we assume that the domains $\mathfrak{U}$, $\mathfrak{X}^{(s)}$, $\mathfrak{Y}^{(s)}$, $s \in \mathcal{S}$ are finite, and that the functions $f^{(s)}(\cdot)$, $s \in \mathcal{S}$, $I^{(ss')}(\cdot)$, $s \in \mathcal{S}$, $(s,s') \in \mathfrak{I}_s$, and $l_k^{(ss')}(\cdot)$, $k, s \in \mathcal{S}$, $(s,s') \in \mathfrak{I}_s$ are bounded.

Key Variables. To characterize the state of the system and the transitions occurring at any time t within the time horizon, the following key sets of binary variables are defined:

$$X_{st} = \begin{cases} 1, & \text{if the system is in state } s \text{ at time } t, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall s, t$$

$$L_{ss't} = \begin{cases} 1, & \text{if the logical condition corresponding to the transition from state } s \text{ to state } s' \text{ is true at time } t, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall s, (s,s') \in \mathfrak{I}_s, t$$

$$\tilde{L}_{ss't} = \begin{cases} 1, & \text{if a transition from state } s \text{ to state } s' \text{ takes place at time } t, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall s, (s,s') \in \mathfrak{I}_s, t.$$

Other auxiliary variables will be introduced where necessary.

System State Characterization. The state of the system at any time is uniquely defined:

$$\sum_{s \in S} X_{st} = 1, \qquad t = 0 \ldots \mathfrak{K}. \tag{3}$$

For the system shown in Figure 4, this constraint is written as follows:

$$X_{1t} + X_{2t} + X_{3t} = 1, \qquad t = 0 \ldots \mathfrak{K}.$$

System Behavior. When the system is in state s, the corresponding set of describing equations, 1, must hold, that is, $f^{(s)}(\cdot) = 0$. This can be written mathematically as

$$\mathfrak{L}_F^{(s)}(1 - X_{st}) \le f^{(s)}(x_{t+1}^{(s)}, x_t^{(s)}, y_t^{(s)}, u_t) \le \mathfrak{U}_F^{(s)}(1 - X_{st}), \qquad \forall s \in \mathcal{S}, t = 0 \ldots \mathfrak{K} - 1, \tag{4}$$

where $\mathfrak{L}_F^{(s)}$ and $\mathfrak{U}_F^{(s)}$ are suitable lower and upper bounds on $f^{(s)}(\cdot)$. We note that if $X_{st} = 1$, then the preceding expression reduces to $f^{(s)}(\cdot) = 0$, while it is nonconstraining if $X_{st} = 0$.

For instance, for the second state in the example system, Eq. 4 yields the following constraints:

$$\mathfrak{L}_F^{(2)}(1 - X_{2t}) \le x_{t+1}^{(2)} - (x_t^{(2)} + 5y_t^{(2)}x_t^{(2)} + 1) \le \mathfrak{U}_F^{(2)}(1 - X_{2t}), \qquad t = 0 \ldots \mathfrak{K} - 1$$

$$\mathfrak{L}_F^{(2)}(1 - X_{2t}) \le y_t^{(2)} - (3u_t - 5) \le \mathfrak{U}_F^{(2)}(1 - X_{2t}), \qquad t = 0 \ldots \mathfrak{K} - 1.$$

The values of the constants $\mathfrak{L}_F^{(2)}$, $\mathfrak{U}_F^{(2)}$ have to be selected so that the constraints

$$\mathfrak{L}_F^{(2)} \le x_{t+1}^{(2)} - (x_t^{(2)} + 5y_t^{(2)}x_t^{(2)} + 1) \le \mathfrak{U}_F^{(2)}, \qquad t = 0 \ldots \mathfrak{K} - 1$$

$$\mathfrak{L}_F^{(2)} \le y_t^{(2)} - (3u_t - 5) \le \mathfrak{U}_F^{(2)}, \qquad t = 0 \ldots \mathfrak{K} - 1$$

cannot be violated for any admissible set of values for $x_t^{(2)}$, $x_{t+1}^{(2)}$, $y_t^{(2)}$, and u_t. (Note that it is not necessary to use the same $\mathfrak{L}_F^{(s)}$ and $\mathfrak{U}_F^{(s)}$ constants for all equations in a given state. In fact, there are computational advantages in using the tightest values (i.e., largest $\mathfrak{L}_F^{(s)}$ and smallest $\mathfrak{U}_F^{(s)}$) consistent with each equation.)

State Transitions. The transition $s \to s'$ actually takes place if and only if the system is in state s at time t *and* the logical condition corresponding to the transition from s to s' is true. This can be written mathematically as:

$$\tilde{L}_{ss't} \le L_{ss't}, \qquad \forall s \in \mathcal{S}, (s,s') \in \mathfrak{I}_s, \quad t = 0 \ldots \mathfrak{K} - 1 \tag{5}$$

$$\tilde{L}_{ss't} \le X_{st}, \qquad \forall s \in \mathcal{S}, (s,s') \in \mathfrak{I}_s, t = 0 \ldots \mathfrak{K} - 1 \tag{6}$$

$$\tilde{L}_{ss't} \ge L_{ss't} + X_{st} - 1, \qquad \forall s \in \mathcal{S}, (s,s') \in \mathfrak{I}_s, \quad t = 0 \ldots \mathfrak{K} - 1. \tag{7}$$

It can be seen that Eqs. 5 and 6 ensure that $\tilde{L}_{ss't}$ will be zero (i.e., the transition $s \to s'$ will not take place at time t) if either $L_{ss't}$ or X_{st} are zero (i.e., if the corresponding logical condition is not satisfied or the system is not in that state). On the other hand, if both $L_{ss't}$ and X_{st} are one, then Eq. 7 ensures that $\tilde{L}_{ss't} = 1$ (i.e., the transition must take place).

If the transition $s \to s'$ does take place at time t, the system finds itself in state s' at time $t+1$:

$$X_{s',t+1} \ge \tilde{L}_{ss't}, \qquad \forall s \in \mathcal{S}, (s,s') \in \mathfrak{I}_s, t = 0 \ldots \mathfrak{K} - 1. \tag{8}$$

We note that Eq. 8 together with Eq. 3 automatically ensure that all other $X_{s'',t+1}$ $(s'' \neq s')$ will take a value of zero in this case.

For the mathematical model to describe the system behavior in a unique and unambiguous manner, we must also define what happens if *none* of the transitions associated with

the current state s of the system is triggered. In such a case, the system state will often remain unaltered. However, more generally, the intrinsic dynamic behavior of the system may cause it to move to a different state. In any case, this state is uniquely defined and is denoted by s_s^*. The corresponding *intrinsic* system transition can then be enforced through constraints of the following form:

$$X_{s_s^*,t+1} \geq X_{st} - \sum_{(s,s')\in \mathfrak{I}_s} \tilde{L}_{ss't}, \qquad \forall s \in \mathcal{S}, t = 0 \ldots \mathcal{H}-1. \tag{9}$$

We note that, if the system is in state s at time t (i.e., $X_{st}=1$) and none of the transitions $(s,s')\in\mathfrak{I}_s$ takes place (i.e., $\tilde{L}_{ss't}=0, \forall s':(s,s')\in\mathfrak{I}_s$), then Eq. 9 forces $X_{s_s^*,t+1}$ to take a value of one.

Binary Variable Encoding of Logical Conditions. As explained in the subsection on states and transitions, the occurrence of the transition from a state s to a state s' at a time t is determined by a logical condition expressed in terms of the values of the variables $\boldsymbol{x}_t^{(s)}$, $\boldsymbol{y}_t^{(s)}$ and/or the external inputs $\boldsymbol{u}_t$. Furthermore, a formal description of logical conditions of arbitrary complexity was introduced.

We now need to formulate mathematically the relationship between the binary variable $L_{ss't}$ associated with the logical condition and these continuous variable values. To this end, we introduce auxiliary binary variables $\hat{L}_{kss't}$ defined as:

$$\hat{L}_{kss't} = \begin{cases} 1, & \text{if node } k \text{ of the tree describing the logical condition associated with the transition } s \to s' \text{ is TRUE at time } t, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall k, s, (s,s') \in \mathfrak{I}_s, t.$$

We can then characterize these auxiliary variables depending on the type of node that they correspond to.

Leaf Nodes. In this case, $\hat{L}_{kss't}$ must take a value of one if the value of the algebraic expression $l_k^{(ss')}(\boldsymbol{x}_t^{(s)}, \boldsymbol{y}_t^{(s)}, \boldsymbol{u}_t)$ is nonpositive, and zero otherwise. This can be achieved by the constraints:

$$\mathfrak{L}_L^{(ss')}\hat{L}_{kss't} + \epsilon \leq l_k^{(ss')}(\boldsymbol{x}_t^{(s)}, \boldsymbol{y}_t^{(s)}, \boldsymbol{u}_t) \leq \mathfrak{U}_L^{(ss')}(1-\hat{L}_{kss't}), \qquad \forall k \in LF^{(ss')}, s, (s,s') \in \mathfrak{I}_s, t, \tag{10}$$

where $\mathfrak{L}_L^{(ss')}$ and $\mathfrak{U}_L^{(ss')}$ are suitable lower and upper bounds on $l_k^{(ss')}(\cdot)$, and ϵ a small positive tolerance. Note that if $l_k^{(ss')}(\cdot)\leq 0$, then $\hat{L}_{kss't}$ must take a value of one in order not to violate the first inequality in Eq. 10. Furthermore, if $l_k^{(ss')}(\cdot)>0$, then $\hat{L}_{kss't}$ is forced to zero so as not to violate the second inequality in Eq. 10.

Disjunctive Nodes. In this case, $\hat{L}_{kss't}$ must take a value of one if either of the two descendant nodes of node k has a value of TRUE; otherwise, $\hat{L}_{kss't}$ must be zero. This can be achieved by the constraints:

$$\hat{L}_{kss't} \leq \hat{L}_{\text{left}(k),ss't} + \hat{L}_{\text{right}(k),ss't}, \qquad \forall k \in DJ^{(ss')}, s, (s,s') \in \mathfrak{I}_s, t \tag{11a}$$

$$\hat{L}_{kss't} \geq \hat{L}_{\text{left}(k),ss't}, \qquad \forall k \in DJ^{(ss')}, s, (s,s') \in \mathfrak{I}_s, t \tag{11b}$$

$$\hat{L}_{kss't} \geq \hat{L}_{\text{right}(k),ss't}, \qquad \forall k \in DJ^{(ss')}, s, (s,s') \in \mathfrak{I}_s, t. \tag{11c}$$

We note that if *both* $\hat{L}_{\text{left}(k),ss't}$ and $\hat{L}_{\text{right}(k),ss't}$ are zero, then Eq. 11a ensures that $\hat{L}_{kss't}$ is also zero while Eqs. 11b and 11c are nonconstraining. On the other hand, if either or both of $\hat{L}_{\text{left}(k),ss't}$ and $\hat{L}_{\text{right}(k),ss't}$ take a value of one, then Eqs. 11b and/or 11c force $\hat{L}_{kss't}$ to adopt a value of one, while Eq. 11a becomes nonconstraining.

Conjunctive Nodes. In this case, $\hat{L}_{kss't}$ must take a value of one if both of the two descendent nodes of node k have a value of TRUE; otherwise, $\hat{L}_{kss't}$ must be zero. This can be achieved by the constraints:

$$\hat{L}_{kss't} \geq \hat{L}_{\text{left}(k),ss't} + \hat{L}_{\text{right}(k),ss't} - 1, \qquad \forall k \in CJ^{(ss')}, s, (s,s') \in \mathfrak{I}_s, t \tag{12a}$$

$$\hat{L}_{kss't} \leq \hat{L}_{\text{left}(k),ss't}, \qquad \forall k \in CJ^{(ss')}, s, (s,s') \in \mathfrak{I}_s, t \tag{12b}$$

$$\hat{L}_{kss't} \leq \hat{L}_{\text{right}(k),ss't}, \qquad \forall k \in CJ^{(ss')}, s, (s,s') \in \mathfrak{I}_s, t. \tag{12c}$$

We note that if *both* $\hat{L}_{\text{left}(k),ss't}$ and $\hat{L}_{\text{right}(k),ss't}$ are one, then Eq. 12a ensures that $\hat{L}_{kss't}$ is also one, while Eqs. 12b and 12c are nonconstraining. On the other hand, if either or both of $\hat{L}_{\text{left}(k),ss't}$ and $\hat{L}_{\text{right}(k),ss't}$ take a value of zero, then Eqs. 12b and/or 12c force $\hat{L}_{kss't}$ to adopt a value of zero, while Eq. 12a becomes nonconstraining.

Negation Nodes. In this case, $\hat{L}_{kss't}$ must take a value of one if its descendant node has a value of FALSE; otherwise, $\hat{L}_{kss't}$ must be zero. This can be achieved via the simple constraint:

$$\hat{L}_{kss't} + \hat{L}_{\text{desc}(k),ss't} = 1, \qquad \forall k \in NG^{(ss')}, s, (s,s') \in \mathfrak{I}_s, t. \tag{13}$$

The preceding constraints essentially determine the value of the binary variable $\hat{L}_{kss't}$ associated with node k of the tree describing the logical condition that triggers the transition $s \to s'$. The binary variable $L_{ss't}$ that determines the truth or otherwise of the entire logical expression is equal to the variable $\hat{L}_{k^*ss't}$ corresponding to the root node k^* of the tree. It is worth mentioning that very often the logical expression

triggering a transition is so simple that the corresponding tree comprises a single leaf node. In such cases, there is, of course, no need to introduce the auxiliary variables $\hat{L}_{kss't}$ and constraints 10 can be written directly in terms of the variable $L_{ss't}$ instead.

As an illustration of the preceding ideas, we list below the constraints associated with the logical condition that triggers the transition $1 \rightarrow 2$ in the example of Figure 4. The binary tree for this logical condition has already been shown in Figure 3.

Node 1

$$L_{1,2,t} \le \hat{L}_{2,1,2,t} + \hat{L}_{6,1,2,t}$$

$$L_{1,2,t} \ge \hat{L}_{2,1,2,t}$$

$$L_{1,2,t} \ge \hat{L}_{6,1,2,t}.$$

Node 2

$$\hat{L}_{2,1,2,t} + \hat{L}_{3,1,2,t} = 1.$$

Node 3

$$\hat{L}_{3,1,2,t} \ge \hat{L}_{4,1,2,t} + \hat{L}_{5,1,2,t} - 1$$

$$\hat{L}_{3,1,2,t} \le \hat{L}_{4,1,2,t}$$

$$\hat{L}_{3,1,2,t} \le \hat{L}_{5,1,2,t}.$$

Node 4

$$\mathfrak{L}\,\hat{L}_{4,1,2,t} + \epsilon \le x_t^{(1)} - 5 \le \mathfrak{U}(1 - \hat{L}_{4,1,2,t}).$$

Node 5

$$\mathfrak{L}\,\hat{L}_{5,1,2,t} + \epsilon \le 3 - u_t \le \mathfrak{U}(1 - \hat{L}_{5,1,2,t}).$$

Node 6

$$\mathfrak{L}\,\hat{L}_{6,1,2,t} + \epsilon \le 10 - x_t^{(1)} \le \mathfrak{U}(1 - \hat{L}_{6,1,2,t}).$$

where $\mathfrak{L}$ and $\mathfrak{U}$ are appropriate constants.

System Condition Following a Transition. If a transition from state s to state s' takes place at time t, the conditions in s' at $t+1$ have to be calculated, based on the values of the describing variables in s at t. This is done by enforcing the equations $I^{(ss')}(\cdot) = 0$ at time t:

$$\mathfrak{L}_I^{(ss')}(1 - \tilde{L}_{ss't}) \le I^{(ss')}(x_{t+1}^{(s')}, y_{t+1}^{(s')}, x_t^{(s)}, y_t^{(s)}, u_t) \le \mathfrak{U}_I^{(ss')}(1 - \tilde{L}_{ss't}), \qquad \forall s \in \mathcal{S}, (s, s') \in \mathfrak{I}_s, \quad t = 0 \ldots \mathfrak{K} - 1, \quad (14)$$

where $\mathfrak{L}_I^{(ss')}$ and $\mathfrak{U}_I^{(ss')}$ are appropriate lower and upper bounds on $I^{(ss')}(\cdot)$. The preceding constraint simply ensures that $I^{(ss')}(\cdot) = 0$ if $\tilde{L}_{ss't} = 1$, that is, if the transition $s \rightarrow s'$ takes place at time t; otherwise, it is nonconstraining.

For instance, in our example system, when transition $1 \rightarrow 2$ takes place, the initialization relationship $x_{t+1}^{(2)} - x_t^{(1)} = 0$ should be used to calculate the initial state of the system immediately following the transition. This is ensured by Eq. 14, which takes the following form:

$$\mathfrak{L}_I^{(12)}(1 - \tilde{L}_{12t}) \le x_{t+1}^{(2)} - x_t^{(1)} \le \mathfrak{U}_I^{(12)}(1 - \tilde{L}_{12t}), \qquad t = 0 \ldots \mathfrak{K} - 1.$$

Note that when $\tilde{L}_{12t} = 1$ (i.e., when such a transition takes place), the initialization relationship is enforced, whereas when $\tilde{L}_{12t} = 0$ (i.e., when such a transition does not happen), the preceding relationship is nonconstraining. Furthermore, if $\tilde{L}_{12t} = 1$, then Eq. 8 implies that $X_{2,t+1} = 1$, which, by virtue of Eq. 4, results in

$$x_{t+2}^{(2)} - (x_{t+1}^{(2)} + 5y_{t+1}^{(2)} x_{t+1}^{(2)} + 1) = 0$$

$$y_{t+1}^{(2)} - (3u_{t+1} - 5) = 0.$$

We can therefore determine the values of the variables $x_{t+1}^{(2)}$, $y_{t+1}^{(2)}$, and $x_{t+2}^{(2)}$.

System outputs

The output variables, z_t, of a hybrid model of the type described in the subsection on states and transitions are generally defined through constraints of the form:

$$\boldsymbol{g}(\boldsymbol{z}_t, \boldsymbol{u}_t, \{X_{st}, \boldsymbol{x}_t^{(s)}, \boldsymbol{y}_t^{(s)}, s \in \mathcal{S}\}) = 0 \qquad (15)$$

$$\boldsymbol{h}(\boldsymbol{z}_t, \boldsymbol{u}_t, \{X_{st}, \boldsymbol{x}_t^{(s)}, \boldsymbol{y}_t^{(s)}, s \in \mathcal{S}\}) \le 0 \qquad (16)$$

that, given any values $X_{st} \in \{0,1\}$, $\boldsymbol{x}_t^{(s)} \in \mathfrak{X}^{(s)}$, $\boldsymbol{y}_t^{(s)} \in \mathfrak{Y}^{(s)}$, $\boldsymbol{u}_t \in \mathfrak{U}$, determine unique values for $\boldsymbol{z}_t$.

Output variables are particularly useful for building models of complex systems by connecting models of lower-level subsystems through their corresponding input and output variables. In this way, the potentially complex task of modeling a large system is hierarchically decomposed into a number of simpler modeling tasks considered in isolation.

From a mathematical standpoint, these connections may be viewed as additional algebraic equations that can be incorporated in the overall mathematical model. These take the form

$$u_t^{\langle I \rangle} = z_t^{\langle J \rangle}, \qquad (17)$$

where $u_t^{\langle I \rangle}$ are a subset of the input variables of subsystem I and $z_t^{\langle J \rangle}$ a subset of the output variables of subsystem J.

Consider, for instance, the system shown in Figure 5a comprising two storage tanks in series. If we view the output flow rate, F^{out}, as an output variable of the simple storage tank model of Figure 1, we can construct the model for the system in Figure 5a simply by coupling the models for the two tanks view via the following equation (Figure 5b):

$$F_t^{\text{out},1} + F_t^{\text{in},2}.$$

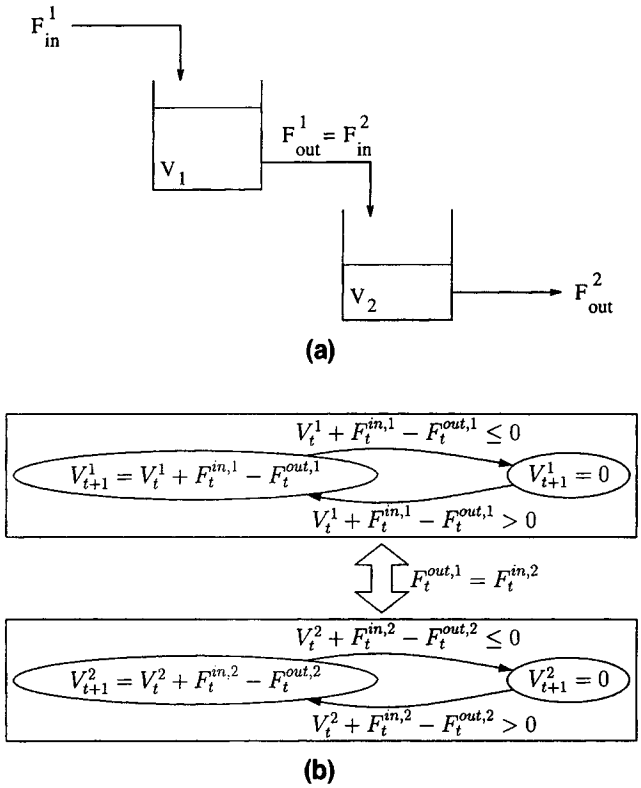

Figure 5. Storage tanks in series: (a) system; (b) modeling representation.

Comments

Relation to Purely Continuous and Purely Discrete System Models. The complete mathematical model comprises Eqs. 3–17. It represents a general description of discrete/continuous processing systems and encapsulates the special cases of purely continuous and purely discrete systems. In the former case, there is only one possible system state ($s = 1$ with $X_{1t} = 1, \forall t$) with continuous describing variables. In the latter case, the differential and algebraic describing variables vanish because the X_{st} variables are sufficient to describe completely the current state of the system and the overall system is equivalent to a finite automaton.

The proposed modeling framework bears much similarity to the hybrid automata modeling framework mentioned earlier. The model structure is essentially identical, the only difference being that, in our method, the description of the system in each state and the logical conditions associated with the transition may, in general, be complex systems of nonlinear equations or complex logical expressions, as opposed to the linear describing equations and simple logical relationships of a hybrid automaton. However, the main difference is that in our approach the abstract representation of a system in terms of states and transitions is translated into a set of equalities and inequalities expressed in terms of sets of binary and continuous variables in the discrete time domain. This model completely describes the behavior of the system, and can therefore be used as a basis for analysis and design (e.g., simulation, verification, controller design, etc.) using well-established mathematical tools for hybrid systems such as mixed-integer programming techniques. The application of these ideas to the safety verification problem is presented in the next section.

We also note that the describing Eqs. 1 are the discrete time analog of mixed systems of differential and algebraic equations of the form

$$\tilde{f}^{(s)}(\dot{x}^{(s)}, x^{(s)}, y^{(s)}, u^{(s)}) = 0$$

that describe the behavior of many processing systems in the continuous time domain. In that context, $x^{(s)}(t)$, $y^{(s)}(t)$, and $u^{(s)}(t)$ represent the differential, algebraic, and system input variables, respectively, while $\dot{x}^{(s)}$ are the time derivatives of $x^{(s)}$. Although $x^{(s)}(t)$ are often also called *state* variables (especially in the control literature), we avoid this terminology to prevent confusion with our concept of system state.

Exploitation of Common Variable and Equation Subsets. In many systems of practical interest, sets of variables $x^{(s)}$, $y^{(s)}$ and the sets of equations $f^{(s)}(\cdot) = 0$ describing two or more states s may overlap to a considerable extent. This can be exploited to reduce the size of the resulting model. Consider, for instance, a set of states $\mathcal{K}(\mathcal{K} \subseteq \mathcal{S})$, and define the sets of variables x and y that occur in *all* states $s \in \mathcal{K}$, that is,

$$x^{[\kappa]} \equiv \bigcap_{s \in \mathcal{K}} \{x^{(s)}\}$$

$$y^{[\kappa]} \equiv \bigcap_{s \in \mathcal{K}} \{y^{(s)}\}.$$

Now, if $f^{[\kappa]}(\cdot) = 0$ is a set of equations that are identical in all states $s \in \mathcal{K}$ and involve only variables $x^{[\kappa]}$, $y^{[\kappa]}$, and u, then we can combine the constraints 4 for all $s \in \mathcal{K}$ into a single constraint:

$$\mathfrak{L}_k^{[\kappa]}\left(1 - \sum_{s \in \mathcal{K}} X_{st}\right) \le f^{[\kappa]}(x_{t+1}^{[\kappa]}, x_t^{[\kappa]}, y_t^{[\kappa]}, u_t) \le \mathfrak{U}_k^{[\kappa]}\left(1 - \sum_{s \in \mathcal{K}} X_{st}\right), \qquad t = 0 \ldots \mathcal{H} - 1. \quad (18)$$

Furthermore, for the special case where $\mathcal{K} = \mathcal{S}$, Eq. 18 together with Eq. 3 imply the much simpler form:

$$f^{[\mathcal{S}]}(x_{t+1}^{[\mathcal{S}]}, x_t^{[\mathcal{S}]}, y_t^{[\mathcal{S}]}, u_t) = 0, \qquad t = 0 \ldots \mathcal{H} - 1, \quad (19)$$

which is merely a mathematical statement of the fact that the subset of equations $f^{[\mathcal{S}]}$ holds irrespective of the current state of the system.

Modeling of Systems with Widely Different Time Scales. In many processing systems, there are large differences in the time scales of operation of the constituent subsystems. For instance, the reaction time of a digital controller is in the range of 5–50 ms, whereas a small valve changes from open to closed in 0.1 s, and larger valves may need up to 1 min for these changes. On the other hand, the average human reaction time is considered to be around 3 s. Although, in principle, it is possible for the models of all components to be written on the same (shortest) time scale, this will have an adverse impact on both the size of the model for the entire system and the complexity of the modeling task.

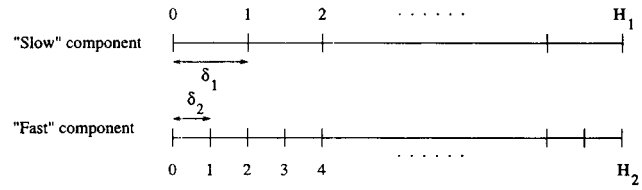

Figure 6. Components operating in different time domains.

The modeling framework described earlier can easily be extended to handle *time domain separation* where components operating on different time scales are also modeled on different time domains. Consider the case where two components operate in two different discrete time domains, $t_1 = 0 \ldots \mathcal{HC}_1$ and $t_2 = 0 \ldots \mathcal{HC}_2$, with the output of component 1 being an input of component 2. It is assumed that $t_1 = 0$ and $t_2 = 0$ correspond to the same instant in real-time terms, and so do $t_1 = \mathcal{HC}_1$ and $t_2 = \mathcal{HC}_2$ (see Figure 6). The model of each of the two components can be written in its own time domain. Then, the simple connecting constraints, Eq. 17, that ensure coordination between the two subsystems are replaced by

$$z_t^{\langle 1 \rangle} = u_{\lfloor t(\delta_1/\delta_2) \rfloor}^{\langle 2 \rangle}, \qquad t = 0 \ldots \mathcal{HC}_1, \tag{20}$$

where δ_1 and δ_2 correspond to actual time steps for the two components, respectively, and $\lfloor x \rfloor$ denotes the largest integer not exceeding $x \in \Re$. This results in the input of component 2 being held constant over $\lfloor \delta_1/\delta_2 \rfloor$ consecutive intervals.

Safety Verification in the Discrete Time Domain

Basic concepts

According to the modeling framework described in the previous section, the values of the binary variables X_{st} determine the state, $s \in \mathbb{S}$, in which a system exists at any time, t. The values of the variables, $x_t^{(s)}$ and $y_t^{(s)}$, then provide the additional information needed to describe the system within that state. On the other hand, the values of the input variables u_t can be considered as degrees of freedom that vary independently affecting the behavior of the system.

We now turn our attention to the problem of verifying the safety of such systems. A process is usually deemed to be "unsafe" when, under the effect of external disturbances, it reaches certain undesirable states and the describing variables in those states take values within certain undesirable regions. For example, the storage tank system of Figure 1 may be considered unsafe when it reaches either of the following undesirable situations:

1. The tank is in the nonempty state and the volume of liquid in the tank exceeds a certain limit (overflow).
2. The tank is in the empty state (drainage).

In general, the space of possible dangerous conditions, $\mathcal{F}$, comprises a set of potentially unsafe states, $\mathbb{S}_F \subseteq \mathbb{S}$, and certain unsafe operating regimes within each of these states. The latter are assumed to be hyperrectangles defined by given lower and upper bounds (not necessarily finite) on the describing variables of the state concerned, or combinations thereof:

$$\mathcal{F} \equiv \bigcup_{s \in \mathbb{S}_F} \left\{ \bigcup_{k=1}^{N_F^{(s)}} \{x^{(s)}, y^{(s)} \mid \chi_{ks}^L \le x^{(s)} \le \chi_{ks}^U, \psi_{ks}^L \le y^{(s)} \le \psi_{ks}^U\} \right\},$$

where $N_F^{(s)}$ is the number of unsafe variable ranges within state $s \in \mathbb{S}_F$. In this way, unsafe operating conditions within each state can be defined as combinations of ranges of values for the various process parameters (e.g., temperature greater than 500 K *and* reactor holdup greater than 10 tons).

Similarly, the space of possible initial conditions for the system, $\mathcal{I}$, can be defined as follows:

$$\mathcal{I} \equiv \bigcup_{s \in \mathbb{S}_I} \left\{ \bigcup_{l=1}^{N_I^{(s)}} \left\{ x^{(s)}, y^{(s)} \mid \bar{\chi}_{ls}^L \le x^{(s)} \le \bar{\chi}_{ls}^U, \bar{\psi}_{ls}^L \le y^{(s)} \le \bar{\psi}_{ls}^U \right\} \right\},$$

where $\mathbb{S}_I \subseteq \mathbb{S}$ is the set of possible initial states and $N_I^{(s)}$ is the number of initial variable ranges within state $s \in \mathbb{S}_I$.

In principle, it could reasonably be argued that very few processes are permanently and unconditionally safe. For the purposes of this article, a process is characterized as inherently safe with respect to a given space of disturbances, $\mathcal{D}$, and time horizon, $\mathcal{HC}$. Here, $\mathcal{D}$ is a subset of the allowable space of input variables, that is,

$$u_t \in \mathcal{D} \subseteq \mathcal{U}.$$

The *safety* verification problem is then stated as follows: Is there a sequence of disturbances from within the space of disturbances $\mathcal{D}$ that may lead the system from any initial condition in $\mathcal{I}$ to any potentially unsafe condition in $\mathcal{F}$ within the time horizon $\mathcal{HC}$? This problem is illustrated schematically in Figure 7 for a simple system involving only one state and two continuous describing variables x_1 and x_2.

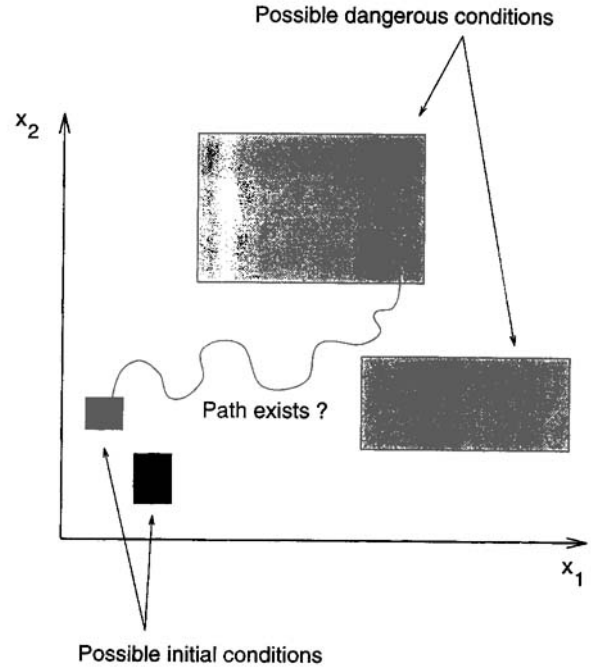

Figure 7. Representation of the safety verification problem.

Mathematical formulation

Consider a dynamic system operating in a discrete time domain $t = 0 \ldots \mathcal{H}$. The dynamic model of the system comprises Eqs. 3–17 and is therefore a set of algebraic equalities and inequalities.

Initial System Condition. In assessing the safety of the system, the only scenarios of interest are those in which the initial state is within the space of possible initial conditions, $\mathcal{I}$. To account for this, we introduce binary variables $\zeta_{sl}, \forall\, s \in \mathcal{S}_I, l = 1 \ldots N_I^{(s)}$, which take a value of 1 if the system is initially in state s and its describing variables lie within initial condition hyperrectangle l, and a value of 0 otherwise.

The system state will initially lie in one and only one of the initial condition hyperrectangles. This is expressed by the following constraint:

$$\sum_{s \in \mathcal{S}_I} \sum_{l=1}^{N_I^{(s)}} \zeta_{sl} = 1, \tag{21}$$

which implies that exactly one of the ζ_{sl} variables has a value of 1. We note that the initial condition hyperrectangles are not necessarily disjoint. However, if the initial condition lies within two or more overlapping hyperrectangles, then it is not important which one of the corresponding ζ_{sl} variables takes a value of 1 while the rest are set to 0.

Obviously, the system can initially lie in one of the $N_I^{(s)}$ initial condition hyperrectangles l associated with state s (i.e., $\zeta_{sl} = 1$) only if it is in state s in the first place (i.e., $X_{s0} = 1$). This is written mathematically in the following form:

$$X_{s0} \geq \sum_{l=1}^{N_I^{(s)}} \zeta_{sl}, \qquad \forall s \in \mathcal{S}_I. \tag{22}$$

Furthermore, if $\zeta_{sl} = 1$, the describing variables of the system must lie within initial hyperrectangle l. This is enforced by the following constraints:

$$\sum_{l=1}^{N_I^{(s)}} \zeta_{sl} \overline{x}_{ls}^{L} \leq \boldsymbol{x}_0^{(s)} \leq \sum_{l=1}^{N_I^{(s)}} \zeta_{sl} \overline{x}_{ls}^{U}, \qquad \forall s \in \mathcal{S}_I \tag{23}$$

$$\sum_{l=1}^{N_I^{(s)}} \zeta_{sl} \overline{\psi}_{ls}^{L} \leq \boldsymbol{y}_0^{(s)} \leq \sum_{l=1}^{N_I^{(s)}} \zeta_{sl} \overline{\psi}_{ls}^{U}, \qquad \forall s \in \mathcal{S}_I. \tag{24}$$

Note that when one (and, by virtue of Eq. 21, only one) of the ζ_{sl} variables takes a value of one, Eqs. 23 and 24 constrain the values of the system describing variables at $t = 0$ within the bounds of the corresponding initial condition hyperrectangles.

Unsafe System Conditions. We are interested in checking whether the system enters any dangerous region at any time during the time horizon considered. We therefore introduce binary variables $\sigma_t, t = 0 \ldots \mathcal{H}$, which take a value of 1 if the system is safe at time t, and a value of 0 otherwise. We also introduce binary variables $\eta_{skt}, \forall\, s \in \mathcal{S}_F, k = 1 \ldots N_F^{(s)}, t = 0 \ldots \mathcal{H}$, which take a value of 1 if the system is in dangerous state s and its describing variables lie within dangerous condition hyperrectangle k at time t, and a value of 0 otherwise.

At any given time, t, the system is either safe or in (at least) one of the dangerous condition hyperrectangles. Furthermore, if $\eta_{skt} = 1$, the system has to be in state s at time t and its describing variables should lie within the bounds of dangerous condition hyperrectangle k. The resulting constraints are very similar to Eqs. 21–24:

$$\sigma_t + \sum_{s \in \mathcal{S}_F} \sum_{k=1}^{N_F^{(s)}} \eta_{skt} = 1, \qquad \forall t \tag{25}$$

$$X_{st} \geq \sum_{k=1}^{N_F^{(s)}} \eta_{skt}, \qquad \forall s \in \mathcal{S}_F, t \tag{26}$$

$$\sigma_t \, \mathfrak{L}_X^{(s)} + \sum_{k=1}^{N_F^{(s)}} \eta_{skt} \chi_{ks}^{L} \leq \boldsymbol{x}_t^{(s)} \leq \sigma_t \mathfrak{U}_X^{(s)} + \sum_{k=1}^{N_F^{(s)}} \eta_{skt} \chi_{ks}^{U}, \qquad \forall s \in \mathcal{S}_F, t \tag{27}$$

$$\sigma_t \, \mathfrak{L}_Y^{(s)} + \sum_{k=1}^{N_F^{(s)}} \eta_{skt} \psi_{ks}^{L} \leq \boldsymbol{y}_t^{(s)} \leq \sigma_t \mathfrak{U}_Y^{(s)} + \sum_{k=1}^{N_F^{(s)}} \eta_{skt} \psi_{ks}^{U}, \qquad \forall s \in \mathcal{S}_F, t, \tag{28}$$

where $\mathfrak{L}_X^{(s)}$, $\mathfrak{U}_X^{(s)}$ and $\mathfrak{L}_Y^{(s)}, \mathfrak{U}_Y^{(s)}$ are appropriate lower and upper bounds on the describing variables $\boldsymbol{x}^{(s)}$ and $\boldsymbol{y}^{(s)}$, respectively.

Minimum Safety Objective. In the worst-case situation (which is the one we are interested in determining), the system is unsafe for the maximum possible number of time steps during the horizon. We therefore consider the following objective function:

$$\Phi \equiv \min \sum_{t=0}^{\mathcal{H}} \sigma_t \tag{29}$$

subject to constraints 3–17 and 21–28 and

$$\boldsymbol{u}_t \in \mathfrak{D}, \quad \forall t. \tag{30}$$

Clearly, Φ can only take integer values ranging between 0 and $\mathcal{H}+1$. If $\Phi = \mathcal{H}+1$, it is not possible for the system to enter a dangerous state at any time during the horizon and therefore it is intrinsically safe within the space of disturbances considered. On the other hand, if $\Phi < \mathcal{H}+1$, the solution will indicate one possible combination of

- An initial condition, X_{s0}, $\boldsymbol{x}_0^{(s)}$, $\forall s$, *and*
- A sequence of disturbances $\boldsymbol{u}_t$, $t = 0 \ldots \mathcal{H}-1$

that leads to one of the identified dangerous conditions. In fact, the solution will always determine the combination that causes the system to remain unsafe for as many time steps as possible.

Extension: Handling Unsafe Transitions. In some cases, unsafe situations may be associated with a *transition* between certain states taking place rather than the system being in a given state. This can be accommodated within the formula-

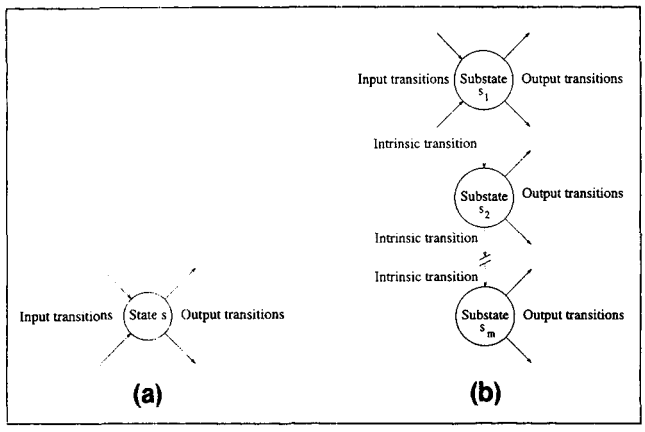

Figure 8. Model modification to deal with persistent unsafe states: (a) original representation; (b) modified representation.

tion presented earlier if we define a set of unsafe transitions, $\mathfrak{I}_F$, and augment objective function 29 with the appropriate transition variables:

$$\Phi \equiv \min \sum_{t=0}^{\mathfrak{K}} \sigma_t - \sum_{t=0}^{\mathfrak{K}} \sum_{(s,s')\in \mathfrak{I}_F} \tilde{L}_{ss't} \qquad (29')$$

It will be recalled (the subsection on the system model) that the binary variable $\tilde{L}_{ss't}$ takes a value of one if transition $s \to s'$ takes place at time t, and is zero otherwise. Thus the only way in which Φ can have an optimal value of $\mathfrak{K}+1$ is if none of the unsafe transitions $(s,s')\in \mathfrak{I}_F$ occur during the horizon.

Extension: Handling Persistent Unsafe States. In certain cases, a system is deemed to be unsafe if it stays in a given state, s, for a minimum number, m, of consecutive time intervals. This can be accommodated by modifying the modeling representation for the system. In particular, we decompose the state s into m substates s_ν, $\nu = 1 \ldots m$, as shown in Figure 8. The new substate s_1 inherits both the input and the output transitions of the original state s. However, substates s_ν, $\nu = 2 \ldots m$ inherit only the output transitions and hence the system can enter state s only through substate s_1 but can leave it from any one of its substates. Finally, substates s_ν and $s_{\nu+1}$ are connected through intrinsic transitions that take place if none of the other output transitions is triggered (cf. constraints 9). The overall effect of this decomposition is that the system reaches substate s_m in the modified model only if it enters state s in the original model at a certain time, *and* stays there for m consecutive time intervals. Therefore the system safety can be verified by applying the verification procedure to the modified model and treating substate s_m as a potentially unsafe state.

Illustrative Examples

This section presents three simple examples of the application of the safety verification procedure to a purely continuous, a purely discrete, and a hybrid system, respectively. We restrict our attention to linear systems for which the formulation of the preceding section results in mixed integer linear programming (MILP) optimization problems.

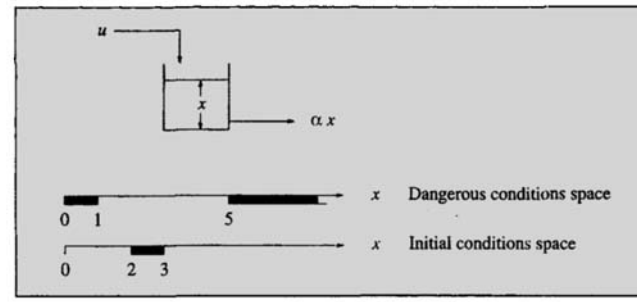

Figure 9. Tank example.

Purely continuous systems

Within the modeling framework proposed in earlier sections, purely continuous systems are easily represented by a single state with continuous describing variables. The continuous differential and algebraic equations that are normally used to describe the dynamic behavior of these systems in the continuous time domain must be suitably discretized over the time horizon of interest.

Consider, for example, the simple tank system depicted in Figure 9. We identify the volume of liquid in the tank as the variable describing the system state ($V \equiv x$), and the feed flow rate as a possible disturbance ($F^{in} \equiv u$). The output flow rate is assumed to be directly proportional to x. The equation that describes the behavior of the system is the dynamic mass balance:

$$\dot{x} = u - \alpha x,$$

which can be discretized to give the following approximation:

$$x_t - x_{t-1} = (u_{t-1} - \alpha x_{t-1})\Delta t.$$

Here, we select a unit time step ($\Delta t = 1$) and a time horizon of 10 time steps.

The two potential dangers for this system are drainage and overflow of the tank. These correspond to two regions in the space of variable x as shown in Figure 9. The initial state of the system is restricted to lie within a third region. Finally, the disturbance variable, u, can take any value between a lower and an upper bound:

$$0 \le u_t \le 0.5.$$

Applying the safety verification formulation to this small example results in an MILP optimization problem. The solution of this problem depends on the value of the proportionality constant, α. When α is small (e.g., $\alpha = 0.05$), meaning that it is difficult to empty the tank, the disturbance that leads to a dangerous condition in the least time (seven time steps) is maximum input flow rate. On the other hand, when α is large (e.g., $\alpha = 0.15$), meaning that it is easy to empty the tank, the disturbance that leads most quickly (five time steps) to a dangerous situation is minimum input flow rate. The results for both cases are shown in Figures 10 and 11.

This simple model can be extended to cover a wider range of disturbances (e.g., feed-pump failures). In any case, the solution is straightforward because the system model is linear

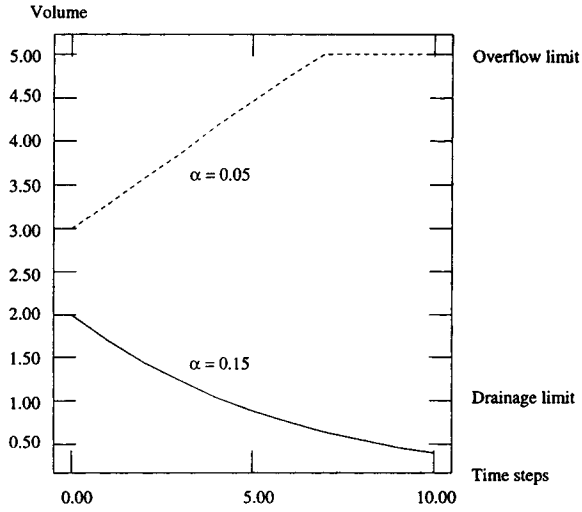

Figure 10. Volume of liquid over time for tank example.

and the resulting optimization problems are MILPs with the number of binary variables depending on the number of time intervals considered.

Purely discrete systems

Although the continuous part of a process is usually described in terms of continuous variables, there is sometimes the need for dynamic models of systems whose states only have logical or symbolic, rather than numerical values. These values change with the occurrence of events that may also be described in nonnumerical terms.

These systems are also easily dealt with using the modeling framework of the second section. In this case, the describing variables, x and y, vanish, and the system states are completely defined through the X_{st} variables. Furthermore, the input variables, u, take binary values, indicating the occurrence or not of the corresponding external events. In this way, the overall representation is completely equivalent to the finite state machine (FSM) formalism that has been used extensively in the past to describe the logical behavior of such systems.

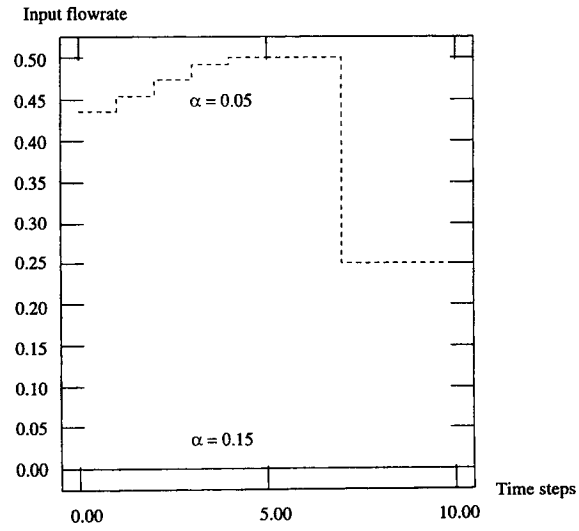

Figure 11. Disturbance over time for tank example.

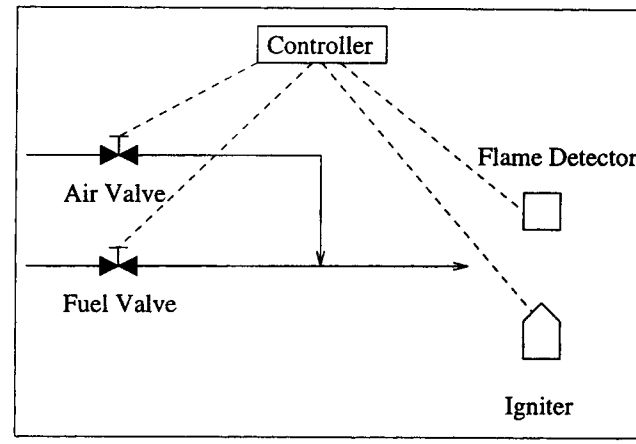

Figure 12. Burner example.

The remaining mathematical model now comprises constraints 3, 5–13 and 15–17.

Consider the simple combustion system illustrated in Figure 12. This example, which was first presented by Moon et al. (1992), involves a burner connected to air and fuel supplies and equipped with an igniter and a flame detector. The five possible states of the system are:

State	Description
1	Empty
2	Air/No Fuel
3	No Air/Fuel
4	Air/Fuel
5	Flame

and the external inputs are:

Input	Control Action/Disturbance
1	Switch Air Valve On (SAVON)
2	Switch Air Valve Off (SAVOFF)
3	Switch Fuel Valve On (SFVON)
4	Switch Fuel Valve Off (SFVOFF)
5	Ignite (IGN)
6	Failure To Ignite (FIGN)
7	Flame Disappears (FlOFF)

Each of the aforementioned events can occur unconditionally at any time and will have the corresponding effect on the behavior of the system.

The operation of this system can readily be modeled in the manner discussed in the second section. In particular, we introduce binary input variables $u_{it} \in \{0,1\}$, $i = 1 \ldots 7$ and variables $X_{st} \in \{0,1\}$, $s = 1 \ldots 5$ characterizing the state of the system at any time t. The resulting representation (Figure 13) is equivalent to an FSM and can be expressed mathematically through the appropriate constraints.

A simple feedback control law that can be applied to this system is the following:

If and only if the current state is *Air/Fuel*, then *Ignite*.

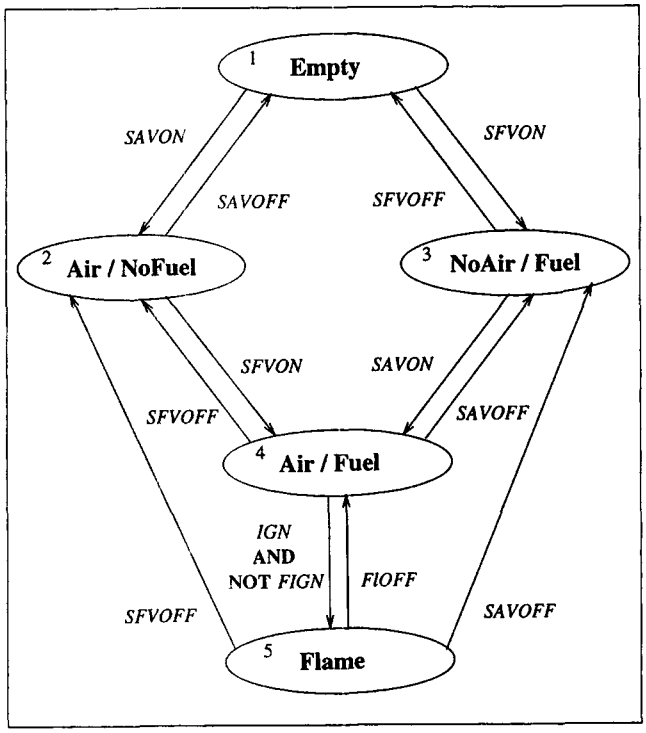

Figure 13. Burner model.

A controller implementing the preceding law is shown in Figure 14. The controller alternates between two states that represent the ignition ($Ign_{t+1} = 1$) and no-ignition ($Ign_{t+1} = 0$) situations, respectively, the transitions between them depending on the value of the input variable AF_t. The burner and controller subsystems are linked through their input and output variables as shown in Figure 15. Thus, the input u_{5t} of the burner subsystem is identified as the output of the controller subsystem; in this case, this output is simply the state of the controller. Similarly, the input AF_t to the controller subsystem is identified as the output of the burner, namely whether or not the latter is in state *Air/Fuel* as determined by the value of the variable X_{4t}.

The initial state of the system is specified as *Empty*. States *Air/Fuel* and *NoAir/Fuel* are deemed to be unsafe because they correspond to situations where fuel accumulates in the combustion chamber without being burned. The application of the safety verification formulation to this problem identifies the obvious unsafe sequence of events shown in Table 1. It can be seen that, according to the control law, the igniter is turned on when the system reaches state *Air/Fuel*, but the disturbance *Failure to Ignite* prevents the system from leaving this state. This clearly points out a deficiency of the control law that needs to be corrected. This error in the controller logic is the same as the one identified by Moon et al. (1992).

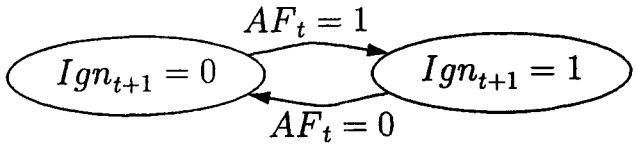

Figure 14. Feedback-controller model for burner.

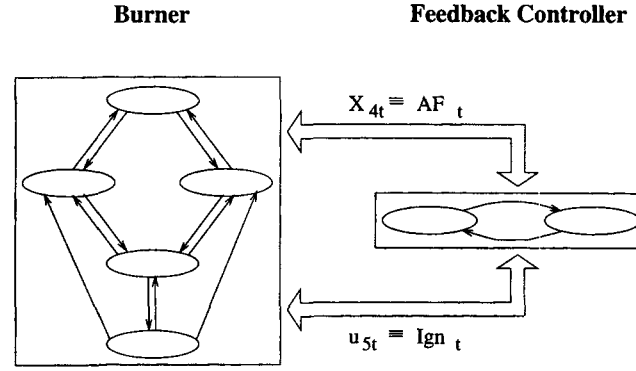

Figure 15. Overall system model for burner example.

Hybrid discrete/continuous systems

The system considered in this example is depicted in Figure 16 and is taken from Chase et al. (1992). A constant liquid flow rate, q, has to be redistributed using N tanks that serve as intermediate inventories. The demand from tank i, q_i, is constant and the total demand is exactly equal to the input flow rate:

$$\sum_{i=1}^{N} q_i = q.$$

The feed can be directed into only one tank at any given time. The time required to switch the inlet flow from one tank to another is assumed to be negligible.

A discrete controller is installed that measures the holdups in the four tanks at discrete time instants and determines the direction of the inlet stream accordingly. Measurements ("sampling") take place every Δt time units. The control action is then determined instantaneously and is held constant over the next time interval of length Δt. The operation of the controller is based on a set of rules that relate the control action taken at t to the system state that was observed at t and possibly at previous time instances $t - \Delta t$, $t - 2\Delta t$, and so on. It is very important that the control action be *uniquely* dictated by these observations. The control rules typically depend on a number of parameters, the values of which must be determined before the controller is implemented on the process.

In view of the preceding considerations, one possible discrete control law for this system is the following:

- If the amount of liquid in tank i, is strictly less than a prespecified amount, $M_i^{\min}$, then mark this tank as "requiring feed."
- Feed is directed to the highest priority tank that is marked as "requiring feed."

Table 1. Verification Results for Burner Example

Time Step	Burner State	Control Action	Disturbance
0	Empty		Switch air valve on
1	Air/no fuel		Switch fuel valve on
2	Air/fuel	Ignition	Failure to ignite
3	Air/fuel	Ignition	Failure to ignite
4	Air/fuel	Ignition	Failure to ignite

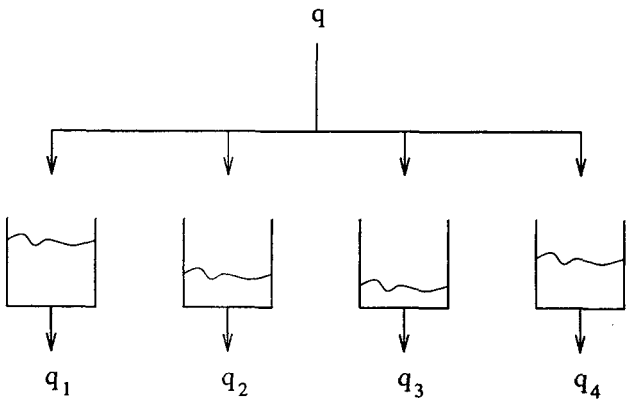

Figure 16. Flow distribution system example.

• If no tank is marked as "requiring feed," feed is directed into the highest priority tank.

It is easily verified that this control law can be used to determine a unique control action for any system state if the following are given:

1. An ordered priority list for the tanks.
2. A set of values for the parameters $M_i^{\min}$.

The objective of the system is to satisfy all downstream demands as much as possible, while at the same time preventing any potentially unsafe situations. If the tanks are assumed to be sufficiently large, overflow is not an issue and the only requirement for the system to operate successfully is that the tanks do not drain completely at any time during operation. Thus "safety" here is synonymous with "availability." The only disturbance considered here is the initial condition of the system, in other words, the amount of liquid in each tank at the beginning of operation.

It is desirable to verify the ability of a given control system to maintain availability within a given time horizon. It is worth noting that the system exhibits inherent hybrid characteristics, as the mass-balance equations take a different form depending on whether the corresponding tank is empty or not. Furthermore, the control law is also hybrid in nature, as it involves discrete control actions (i.e., feed is directed to a tank or not) that depend on values of continuous variables (i.e., the amount of liquid in each tank).

The model of tank i is depicted in Figure 17. It comprises the discretized dynamic mass balance equation for the liquid in the tank and accounts for the fact that no output can be drawn from an empty tank (cf. Figure 1). The model for the discrete controller is shown in Figure 18. It comprises four states that correspond to each of the four possible control actions. The transitions between these states depend on the relationship between the levels of liquid in the four tanks. The four tank models and the controller model are then connected through their corresponding input and output variables to form the overall system model (Figure 19).

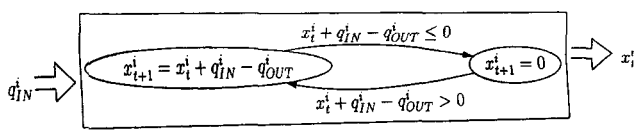

Figure 17. Model for tank i in flow distribution system example.

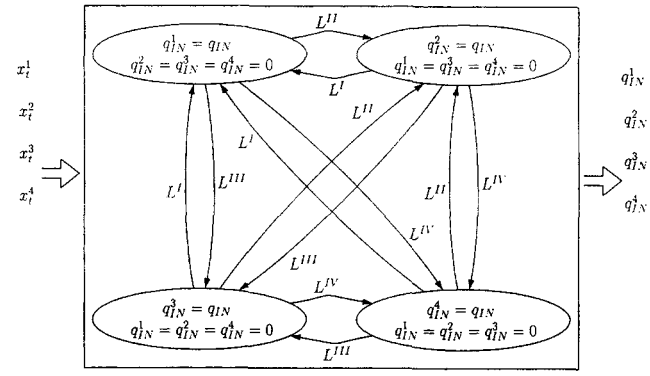

Figure 18. Controller model for flow distribution system example.

$L^{(I)}$:$(x_t^1 < M_1^{\min})$ OR $[x_t^2 \ge M_2^{\min})$ AND $(x_t^3 \ge M_3^{\min})$ AND $(x_t^4 \ge M_4^{\min})]$; L^{II}:$(x_t^1 \ge M_1^{\min})$ AND $(x_t^2 < M_2^{\min})$; L^{III}:$(x_t^1 \ge M_1^{\min})$ AND $(x_t^2 \ge M_2^{\min})$ AND $(x_t^3 < M_3^{\min})$; L^{III}:$(x_t^1 \ge M_1^{\min})$ AND $(x_t^2 \ge M_2^{\min})$ AND $(x_t^3 \ge M_3^{\min})$ AND $(x_t^4 < M_4^{\min})$.

As noted before, safety in this case is synonymous with availability and therefore the states corresponding to each of the tanks being empty are deemed to be "unsafe." Furthermore, the set of admissible initial conditions is restricted by adding the following constraint to the formulation:

$$M_{i0} \ge M^*, \quad \forall i,$$

thereby guaranteeing a minimum initial inventory M^* in each tank.

The safety verification formulation of the third section can now be applied to verify the performance of a given control system. For example, if we consider the system described in Table 2 and a value of $M^* = 2.5$ for the minimum initial holdup, the result of the verification problem for a time horizon of ten time steps is $\Phi = 11$. Therefore, there is no set of admissible initial conditions that results in a tank being empty at any time within the time horizon considered. If, however, the value of the minimum initial holdup is changed to $M^* = 2.0$, the solution of the safety verification problem is $\Phi = 9$, indicating that there is a set of initial conditions against which the performance of the control system is poor. The result is that, twice within the time horizon considered, at least one tank is unavailable.

Note that, in this case, the solution of the MILP problem identifies the following "worst-case" set of initial conditions:

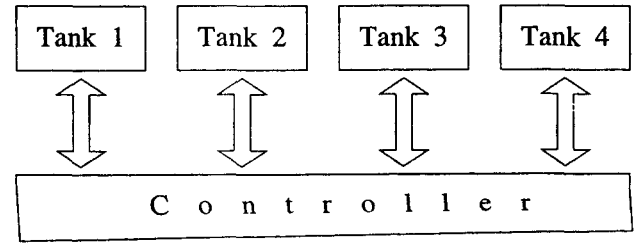

Figure 19. Overall system model for flow distribution system example.

Table 2. Data for Flow Distribution System Example

Tank	1	2	3	4
Input flow, $q\Delta t$		3.00		
Output flow, $q_i\Delta t$	1.25	1.00	0.50	0.25
Controller parameter, $M_i^{\min}$	0.76	2.01	2.01	0.76

$$x_0^1 = x_0^2 = x_0^4 = 2.00,\ x_0^3 = 3.00.$$

Intuitively it could be argued that setting *all* initial holdups to their minimum value (i.e., $x_0^1 = x_0^2 = x_0^3 = x_0^4 = 2.00$) would result in an even worse situation. However, it can be verified that this choice results in exactly the same value for the objective function as that identified by the MILP. The solution of the latter is therefore nonunique.

The evolution of the system state with time is shown in Figure 20. The vertical arrows indicate the tank receiving the input flow over each interval. We note that the first tank is empty at $t = 4$ and $t = 9$.

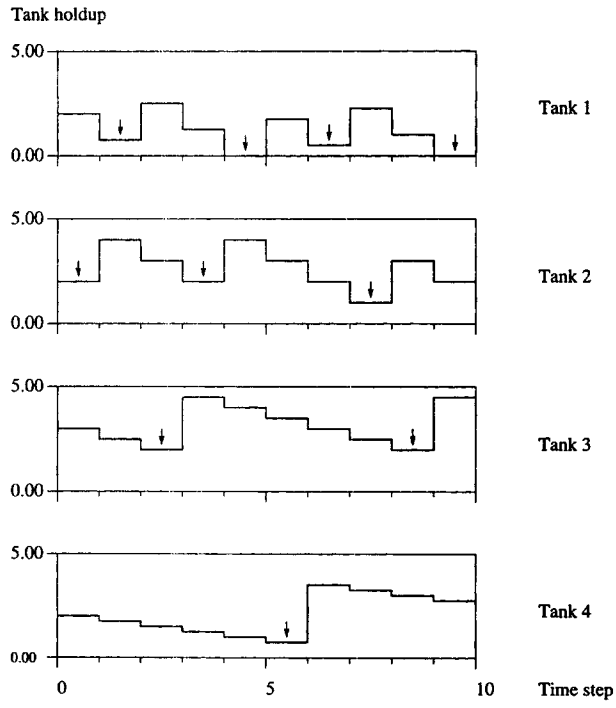

Figure 20. Verification results for flow distribution system example.

Safety Verification of a Batch Fertilizer Plant

In this section, the safety verification procedure is applied to a larger case study on the verification of a batch fertilizer production process, first reported by Probst and Powers (1994). We begin by describing the process under investigation. Then the process model is described and the results of the verification procedure for several cases are compared to those presented by Probst and Powers (1994).

Process description

The processing system under investigation converts waste from a magnesium plant into agricultural fertilizer prills. The waste includes MgO, CaO, and nascent Mg. To obtain a fertilizer containing MgO, CaO, $MgSO_4$, $CaSO_4$, $Mg(OH)_2$, and $Ca(OH)_2$, the waste is sprayed with sulfuric acid. Figure 21 gives a diagram of the process. More detailed descriptions can be found in Probst and Powers (1994), Probst (1996), and Hackenberg (1995).

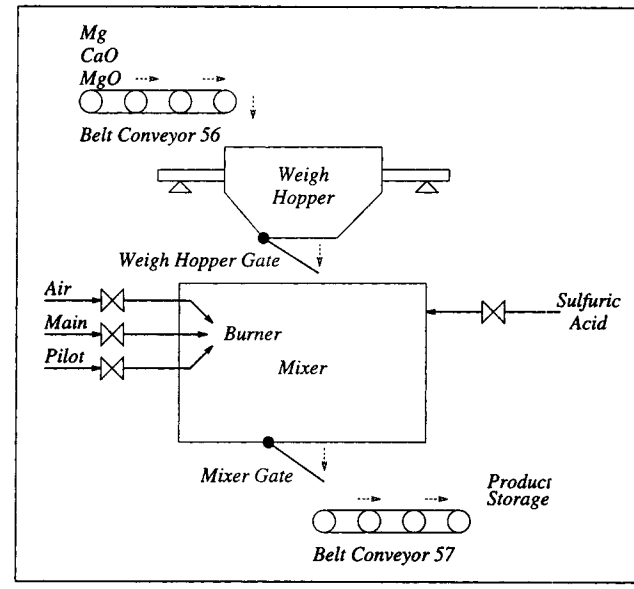

Figure 21. Batch-fertilizer production process.

Material enters the system on a belt conveyor and is deposited into a weigh hopper. When the appropriate amount of material has accumulated in the weigh hopper, its contents are discharged into the mixer. There it is sprayed with sulfuric acid, which causes the evolution of hydrogen gas. To prevent an explosive mixture of hydrogen gas and air from forming inside the mixer during the reaction, the evolving gas is burned by a natural gas burner. The mixer has a moving pan, impeller, and counterrotating inner shell driven by independent motors. After the materials have been adequately mixed, the fertilizer prills leave the mixer through a gate and are taken to storage on a second belt conveyor.

Certain elements of the process are controlled by a programmable logic controller (PLC) executing relay ladder logic (RLL). These include the solenoid valve for the acid spray, the moving parts of the mixer, the discharge gates for the mixer and the weigh hopper, and the motors of the belt conveyors. Burner valves are controlled by a separate system. The inputs to the controller are provided by sensors on the motors and gates, a level sensor in the weigh hopper, a flame detector in the burner, and push buttons on a console.

Human interaction is necessary for operation. The *human operator* follows an instruction list, starting and stopping equipment items from the console when certain responses from the process are observed (e.g., "stop belt conveyor 56 as soon as the weigh hopper has dumped material into the mixer").

Process modeling

The models of the different components of the batch fertilizer production process were constructed according to the modeling framework of the second section. Each component

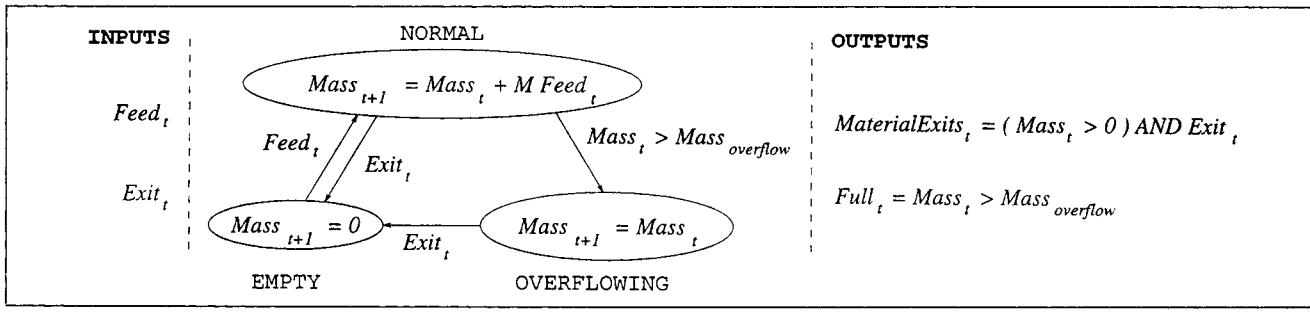

Figure 22. State-transition representation for the weigh hopper.

was modeled as a separate module, with its own state, input, and output variables. The modules corresponding to the mixer, the weigh hopper, and the timers are hybrid in nature, involving both discrete and continuous variables. The plant model was then constructed by connecting the different component models through their input and output variables.

For instance, the belt conveyor module has a binary state variable, *motor*, which is set to one if the input signal to *start* the motor is received and to zero otherwise. Other inputs to the model include a variable to indicate if there is a *source* of material at one end of the conveyor and a variable to determine if the belt is *slipping*. The outputs are a *motor sensor* (set to one when the motor is on), a *zero speed sensor* (set to one when the motor is on and the belt is not slipping), and a variable to indicate *flow of material* (set to one if the motor is on, the belt is not slipping, and there is a source of material).

The weigh hopper module has a continuous variable representing the *mass* of material in the hopper. The corresponding state transition representation is shown in Figure 22. Each time the hopper receives the input signal *feed*, the mass variable is increased by a specified amount. If the hopper receives the input signal *exit*, the mass variable is reset to zero. Finally, if the hopper overflows, the value of the mass variable remains constant. Output variables include a *full* signal to the PLC (set to one if the mass of deposited material exceeds a certain threshold value), and a signal to indicate that *material exits* the hopper (set to one if there is material deposited and the exit signal is received).

The operator is modeled as a finite automaton with a state corresponding to each action in his instruction list. Logical conditions imposed on the transitions determine if the operator executes the next action immediately or waits for a certain condition to be satisfied first.

In order to model the PLC controller, a transformation of the RLL logic into closed form expressions was necessary. A PLC controller scans the values of its input variables from the process at any time t and, based on these and the values of its own internal variables, executes an iteration of the RLL program, producing values for the output variables at time $t + \delta t$. Since RLL rungs are evaluated sequentially during a PLC scan, it usually takes a number of iterations before the values of the output variables of a PLC reach the steady state corresponding to the given values of the inputs. For instance, consider the following logical expressions that correspond to two consecutive rungs of the controller logic:

$$u' = (i_1 \vee u) \wedge \neg y$$
$$y' = i_2 \wedge i_3,$$

where i_1, i_2, and i_3 are input variables, y is an internal PLC variable, and u is an output variable. Primes are used to denote the updated variable values after the scan. Assume that the values of the input variables are $i_1 = i_2 = i_3 = 1$ and the previous state of the internal and output variables is $u = y = 0$. After the first RLL scan, the values of the internal and output variables will be $u = y = 1$. It takes another RLL scan before these values reach their steady state values of $u = 0$ and $y = 1$.

Since the controller is normally very fast compared to all other process elements, it is justified to assume that δt is small compared to the time steps corresponding to the other components i in the system ($\delta_{PLC} \ll \delta_i$). It is then possible to simplify the model of the process considerably if we transform the RLL logic into a set of closed form expressions for the output variables in terms of the input and internal variables. In this way, the controller does not need a separate time scale and the model size is decreased considerably. An algorithm to perform this transformation for RLL logic of arbitrary complexity is presented by Hackenberg (1995) and was applied to construct the model of the PLC controller of the process under investigation. For the small example used earlier, the application of the steady-state aggregation algorithm to the original RLL expression results in the following explicit steady-state expressions:

$$\begin{aligned} u' = {} & (i_1 \wedge \neg i_2) \vee \\ & (i_1 \wedge \neg i_3) \vee \\ & (u \wedge \neg y \wedge \neg i_2) \vee \\ & (u \wedge \neg y \wedge \neg i_3) \\ y' = {} & i_2 \wedge i_3, \end{aligned}$$

which are completely equivalent to the original expressions. It can easily be verified that they reach the same steady state for *any* combination of input variable values and in only *one* RLL scan.

Safety verification results

The obvious initial condition for the verification problem is that of a shutdown plant. However, if this is the only possible initial condition, the time horizon for every model to be checked has to be big enough for the process cycle to reach its end. Otherwise, errors occurring at the later stages of operation may not be identified, as the plant cannot reach these

stages within the time horizon considered. On the other hand, longer time horizons result in larger optimization problems, the solution of which is computationally expensive or even intractable.

In order to overcome this difficulty, the base-case model can be used to capture "snapshots" of the plant operating normally at a number of instants within the time horizon, namely, at $t = 10$, $t = 20$, $t = 30$, and $t = 40$. A set of possible initial conditions can then be generated, corresponding to different stages in the operation of the plant. This set defines the space $\mathcal{I}$ of possible initial conditions in our safety verification formulation (subsection on the mathematical formulation). By automatically choosing one of these initial conditions, the algorithm will have the ability to examine events happening at the later stages of the process using shorter time horizons.

For the purposes of safety verification, the set of unsafe (or otherwise undesirable) conditions comprised the following situations:

1. A gate receives signals to open and close at the same time.
2. The weigh hopper overflows.
3. The acid spray is on but there is no material in the mixer.
4. There is material in the mixer and at least one of the mixer components is not working properly.
5. The main gas valve is open but no flame exists.
6. There is material in the mixer but no flame exists.

Several different models of the fertilizer plant were checked against these specifications. In all these models, the system components were divided into two sets: a "slow" set comprising the weigh hopper, the operator, and both gates, and a "fast" set comprising the rest of the components (i.e., the belt conveyors, the burner, the mixer, and the PLC controller). The time step ratio, δ_2/δ_1, was set to a value of 2, that is, a time step for the slow components is equivalent to two time steps for the fast components. All the resulting MILP problems were solved using the IBM/OSL (IBM, 1992) code. The most interesting results are summarized in Table 3.

Model Without Failure Modes — Base. A base model with no failure modes was first examined to detect any serious errors in the RLL program and/or the instructions to the operator. This model had no degrees of freedom except for the variable *again* that determines whether a second batch is to be carried out or not. The solution indicates that the plant operates without violating any of the specifications.

Model With a Single Belt Conveyor Sensor Failure — Consensor. Here failure modes were introduced for the motors and zero speed sensors of the belt conveyors. The new models for the sensors involve two additional variables, indicating whether the sensor fails high or low, respectively. Only one of these conditions may be true at any one time. If a sensor fails high, its output signal is set to high regardless of its real value. The opposite happens if the sensor fails low. It was assumed that no corrective action is taken and thus, once a sensor fails high or low, it remains in that state until the end of the time horizon. Furthermore, a constraint was added to restrict the maximum number of failed sensors at any given time to one.

The solution identifies a combination of events that lead to the occurrence of unsafe condition 6 at $t = 25$. This happens because the zero speed switch on belt conveyor 57 fails low at $t = 22$. As a result, the burner is switched off (the signal to start the burner is latched to the zero speed switch). However, due to the dynamics of the burner, the flame is not extinguished until $t = 26$. Thus, at $t = 24$, the PLC is still unaware of any problems with the burner and opens the hopper gate, which results is material being dumped into the mixer and sprayed with acid while there is no flame.

Even though this situation is very improbable (the conveyor sensor has to fail low at exactly the time that the PLC checks the flame detector), it illustrates a major advantage of using detailed, timed models for the system under investigation. Probst and Powers (1994) used untimed, finite state models, in which such a situation cannot arise and therefore concluded that the system is safe.

Model With a Single Gate Sensor Failure — Gatsensor. In this case, the gate model is modified in a similar way to the conveyor model and failure modes are introduced for the gate limit switches. Only one limit switch on either the mixer or the weigh hopper gate is allowed to fail high or low.

The solution indicates that unsafe condition 3 may occur at $t = 27$. The reason is that, at $t = 22$, the weigh hopper gate open limit switch fails high. The PLC, under the impression that the gate is open and material has been dumped into the mixer, energizes the acid-spray delay timer and, after a while, acid is sprayed into the empty mixer. In the meantime, as the closed limit switch signals (correctly) that the gate is closed, no attempt is made to either open or close the gate and the material stays in the hopper indefinitely.

We can also exclude this unsafe condition and solve the problem again to determine if there are any other problems

Table 3. Safety Verification Results for Batch Fertilizer Plant

Time	Time Horizon	Identified Unsafe Situation	Corresponding Initial Condition*	Computational Statistics			
				NC**	NV†	BV††	CPUs‡
Base	60	—	0	28,615	12,800	581	217
Consensor	20	6	10	12,042	4,840	370	1,290
Gatsensor	30	3	0	15,924	6,835	425	86
	30	1	0	15,924	6,835	425	118
Idle	20	—	0	9,916	4,442	213	93
Neglect	25	4	20	12,240	5,468	261	231
	25	3	20	12,240	5,468	261	214

*This indicates the time along the "normal" operation trajectory that was chosen as the beginning of the time horizon by the MILP safety-verification procedure.
**Number of constraints.
†Number of variables.
††Number of binary variables.
‡CPU seconds on a Sun Ultra workstation.

that may arise. Indeed, the solution of this second problem shows that the plant reaches another unsafe situation at $t = 23$. This time, unsafe condition 1 is reached because the weigh hopper gate receives a *close* and an *open* signal at the same time. This happens because, at $t = 20$, the closed gate limit switch fails low. Consequently, the PLC assumes that the gate is open and sends the *close* signal. Meanwhile, the hopper is filling and, when it becomes full, the PLC sends an *open* signal. This example of undesirable behavior was also identified by Probst and Powers (1994).

Both these faults are a result of a design flaw in the RLL logic. The PLC uses the signals from the open and close limit switches independently and without cross-checking. A cross-check should be implemented and any conflicting signals from the limit switches should be reported to the operator as well as trigger an emergency procedure.

Model With Operator Idling — Idle. Here, an extra variable, *idle*, is added to the operator model to represent the possibility of the operator idling between states. Assume that the finite automaton that represents the operator's behavior is in state s and the logical condition for the transition to state s' is satisifed. If the idle variable is true, the transition takes place as before. If, however, it is true, the transition does not take place and the operator remains in the same state instead.

This situation may arise very frequently and the model can be used to check if any of the actions taken by the operator are time-critical, any delay in their execution leading to undesirable situations. The solution indicates that this is not the case and the system will not reach a dangerous situation as a result of these delays.

Model With Operator Neglecting Actions — Neglect. In the previous model, it was possible for the operator to remain idle but the actions corresponding to each state were still executed, sooner or later. In this model, an extra variable, *neglect*, is introduced in the operator model to represent the possibility of the operator not executing the required actions.

The solution indicates that unsafe condition 4 may arise at $t = 41$ if the operator neglects to press the stop button for conveyor 56 but chooses to shutdown the process after the first batch. In this case, the weigh hopper will fill again and, after the first batch is complete, material will be discharged into the mixer. However, by then the operator will have shut down the mixer components and therefore material will be in the mixer without its components working.

Another sequence of events gives rise to unsafe condition 3 at $t = 35$. The acid spray does not stop automatically when the mixer gate opens but has to be shut down manually. Thus, if the operator neglects to take action, acid will be sprayed into the empty mixer.

Both situations should be considered carefully. If it is judged that they pose a significant threat to the process, then the controller logic should be modified to perform automatically some of the tasks that now have to be performed manually.

As expected, with the exception of model consensor, the results are similar to those of Probst and Powers (1994), because the system models used are very similar. However, since the hybrid characteristics of the process are explicitly accounted for, the state explosion problem is avoided. (The state explosion problem in purely discrete systems is either caused by the sheer complexity of the system at hand or by the fact that attempts are made to model continuous subsystems by discretizing the continuous variables. Naturally, a hybrid methodology can only address the second case. Purely discrete subsystems may still require a model comprising a large number of states, but this will be due to their complexity and not an approximation.) This becomes more obvious when the continuous characteristics of a process become more significant. Furthermore, the addition of more degrees of freedom in the model has only a minor effect in the computational difficulty involved. In fact, due to the nature of the branch-and-bound algorithm used for solving the MILP problems, it is easier to detect a potentially unsafe situation for a system with many degrees of freedom than it is to verify that no such situations exist.

Concluding Remarks

A mathematical programming approach to the problem of safety verification in general processing systems operating in the discrete time domain has been presented. The proposed approach is based on a mathematical description of the process and is able to identify in a quantitative fashion the possible hazards together with their exact causes and consequences. Additionally, it can readily be applied to systems comprising multiple equipment items, taking account of any propagation of disturbance effects that might occur.

A general modeling framework was also described that can be used to build the mathematical system models required for the application of the safety verification procedure in a consistent way. The framework can be applied to purely discrete and purely continuous systems. However, it is worth emphasizing that such systems are only a special case of the discrete time systems considered in this article, which can have both discrete and continuous characteristics. In reality, most systems of interest to process engineering are likely to exhibit this hybrid nature. Model complexity is tackled by the ability to decompose hierarchically the structure of a system into a set of connected components, each of which is simpler to model.

One interesting question faced by the approach advocated here is the selection of an appropriate length of the time horizon to be considered. In general, if for a given time horizon, a system is found to be unsafe, then this provides an unambiguous indication that remedial actions may have to be taken. However, if the system is determined to be "safe," then it is unclear whether it will be safe for arbitrarily long periods of operation. In practice, it could be argued that few physical systems are permanently safe: given sufficient time, even minute disturbances may have a cumulatively dangerous outcome. Thus, a "permanent safety" requirement may be too stringent. A more realistic demand is that the system be safe for long enough to allow potential dangers to be detected and averted. In the context of the formulations presented, provided that the system is inspected at least as frequently as the time horizon $\mathcal{H}$ *and* is found to be within the space of allowable initial conditions $\mathcal{I}$, then it can be guaranteed to be safe until at least the time of the next inspection.

The effective application of our method requires an accurate dynamic model of the process. Fortunately, such models are increasingly becoming available even at the design stage

of chemical processes, already being used for a variety of other purposes such as control system design, and startup and shutdown studies. It is interesting to note that, when the values of certain model parameters (e.g., kinetic constants, heats of reaction, etc.) are not known exactly, one can view these parameters as "disturbances," that is, additional sources of uncertainty that can vary independently, affecting the behavior of the system. In this case, the safety verification formulation will determine, in addition to the worst-case set of inputs, the worst-case set of values for these parameters that drive the system to an unsafe situation. The results can be valuable, for example, in deciding whether or not additional effort should be directed toward determining more accurate values for these parameters.

In addition to the mathematical models, the approach still requires significant engineering input. In particular, the space of possible disturbances must be identified and judgments must be made as to the operating regimes that are deemed to be unsafe. Overall the new method complements rather than replaces more traditional approaches. This is particularly important for systems described by nonconvex models. In principle, the application of local optimization techniques in these situations may result in some unsafe situations remaining undetected.

A key feature of our approach is the formulation of the safety verification problem as a mixed integer optimization problem. The latter may involve hundreds or even thousands of binary variables and, therefore, its solution is by no means a trivial task. On the other hand, as illustrated by the case study of the preceding section, it is possible to solve problems of practical significance yielding useful, nontrivial results using currently available software and hardware. Perhaps a more serious problem is posed by the existence of nonlinearities and nonconvexities in process models. Although the formulation proposed in this article is entirely general, its solution for nonlinear problems could be problematic—indeed, all examples presented here were linear.

Current research effort is directed toward integrating our model-based approach with a qualitative knowledge-based approach (Srinivasan et al., 1997). The broad details of a particular hazardous scenario are extracted by inexpensive qualitative analyses. A detailed quantitative analysis is then performed if needed and only on those parts of the plant identified by the qualitative analysis to contribute to the hazard.

Finally, an interesting feature of the approach is that the quantitative information obtained from the safety verification step can be used to modify the design of the process and/or the associated controller so that it is able to deal with the considered disturbances. Some work in this direction is already under way (Dimitriadis et al., 1995).

Literature Cited

Alur, R., C. Courcoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The Algorithmic Analysis of Hybrid Systems," *Theor. Comput. Sci.*, **138**, 3 (1995).

Alur, R., C. Courcoubetis, T. Henzinger, and P. Ho, "Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems," *Hybrid Systems*, R. Grossman and A. Nerode, eds., No. 736, *Lecture Notes in Computer Science*, Springer-Verlag, New York, p. 209 (1993).

Barton, P., and C. Pantelides, "Modeling of Combined Discrete/Continuous Processes," *AIChE J.*, **40**, 966 (1994).

Cassandras, C., *Discrete Event Systems: Modeling and Performance Analysis*, Aksen, Boston (1993).

Catino, A., and L. Ungar, "Model-based Approach to Automated Hazard Identification of Chemical Plants," *AIChE J.*, **41**, 97 (1995).

Chase, C., J. Serrano, and P. Ramadge, "Periodicity and Chaos from Switched Flow Systems: Contrasting Examples of Discretely Controlled Continuous Systems," *Proc. Conf. on Control and Its Applications*, Minneapolis (1992).

Clarke, E., E. Emerson, and A. Sistla, "Automatic Verification of Finite State Concurrent Systems Using Temporal Logic Specifications," *ACM Trans. Program. Lang. Syst.*, **8**, 244 (1986).

Crowl, D., and J. Louvar, *Chemical Process Safety: Fundamentals with Applications*, Prentice Hall, Englewood Cliffs, NJ (1990).

Dimitriadis, V., N. Shah, and C. Pantelides, "Optimal Design of Hybrid Controllers for Hybrid Process Systems," DIMACS Workshop on Verification and Control of Hybrid Systems, New Brunswick, NJ (1995).

Grossman, R., and A. Nerode, eds., *Hybrid Systems*, No. 736, *Lecture Notes in Computer Science*, Springer-Verlag, New York (1993).

Hackenberg, J., "Modelling and Safety Verification of a Batch Fertiliser Plant," Tech. Rep., Centre for Process Systems Engineering, Imperial College, London (1995).

IBM, *Optimization Subroutine Library: Guide and Reference, Release 2*, 4th ed., IBM Corp., Kingston, New York (1992).

Kletz, T., *What Went Wrong? Case Histories of Process Plant Disasters*, Gulf Publ., Houston, TX (1985).

Kletz, T., *HAZOP and HAZAN. Identifying and Assessing Process Industry Hazards*, 3rd ed., Inst. Chem. Eng., Rugby, UK (1992).

Moon, I., "Automatic Verification of Discrete Chemical Process Control Systems," PhD Thesis, Carnegie Mellon University, Pittsburgh (1992).

Moon, I., and S. Macchietto, "Formal Verification of Batch Processing Control Procedures," *Proc. PSE '94*, Seoul, Korea, p. 469 (1994).

Moon, I., G. Powers, J. Burch, and E. Clarke, "Automatic Verification of Sequential Control Systems Using Temporal Logic," *AIChE J.*, **38**, 67 (1992).

Nicollin, X., A. Olivero, J. Sifakis, and S. Yovine, "An Approach to the Description and Analysis of Hybrid Systems," *Hybrid Systems*, R. Grossman and A. Nerode, eds., No. 736, *Lecture Notes in Computer Science*, Springer-Verlag, New York, p. 149 (1993).

Park, T., and P. Barton, "Towards Dynamic Simulation of a Process and Its Automatic Protective System," Int. Symp. and Workshop on Safe Chemical Process Automation, Houston, TX (1994).

Probst, S., "Chemical Process Safety and Operability Analysis," PhD Thesis, Carnegie Mellon University, Pittsburgh (1996).

Probst, S., and G. Powers, "Automatic Verification of Control Logic in the Presence of Process Faults," AIChE Meeting, San Francisco (1994).

Srinivasan, R., V. Dimitriadis, N. Shah, and V. Venkatasubramanian, "Integrating Knowledge Based and Mathematical Programming Approaches for Process Safety Verification," ESCAPE-7, Trondheim, Norway (1997).

Venkatasubramanian, V., and R. Vaidhyanathan, "A Knowledge-Based Framework for Automating Hazop Analysis," *AIChE J.*, **40**, 496 (1994).

Waters, A., and J. Ponton, "Qualitative Simulation and Fault Propagation in Process Plants," *Chem. Eng. Res. Des.*, **67**, 407 (1989).

Manuscript received June 25, 1996, and revision received Oct. 15, 1996.