P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

Intelligent Control for High Autonomy Process Control Systems[1]

by
P. J. Antsaklis and J. C. Kantor
Departments of Electrical and Chemical Engineering
University of Notre Dame, Notre Dame, IN 46556 USA
(219) 631-5792, (219) 631-5797
antsaklis@nd.edu kantor@nd.edu

Introduction

Highly demanding control requirements in chemical processes coupled with the complexity and uncertainty of the models require the use of sophisticated control methods.

To meet highly demanding control specifications in complex systems a number of methods have been developed that are collectively known as intelligent control methodologies. They enhance and extend traditional control methods. An alternative term used is intelligent autonomous control. It emphasizes the fact that an intelligent controller typically aims to attain higher degrees of autonomy in accomplishing and even setting control goals, rather than stressing the (intelligent) methodology that achieves those goals. Intelligent controllers are envisioned emulating human mental faculties such as adaptation and learning, planning under large uncertainty, coping with large amounts of data etc in order to effectively control complex processes; and this is the justification for the use of the term intelligent, since these mental faculties are considered to be important attributes of human intelligence; see for example [1-4] and the references therein.

In the following, an introduction to the area of Intelligent Control is given first in Sections 1 and 2. A research area important to intelligent control is the area of hybrid control systems and approaches to modeling and control of such systems are discussed in Sections 3 and 4. Concluding remarks are included in Section 5. In addition, a brief outline of some of our recent work in other research areas that are important to intelligent control, such as learning control and the control of DES, is included.

1 On Intelligent Autonomous Control Systems

It is perhaps appropriate to first explain further what it is meant by the term Intelligent Autonomous Control: In the design of controllers for complex dynamical systems there are needs today that cannot be successfully addressed with the existing conventional control theory. Heuristic methods may be needed to tune the parameters of an adaptive control law. New control laws to perform novel control functions to meet new objectives should be designed while the system is in operation. Learning from past experience and planning control actions may be necessary. Failure detection and identification is needed. Such functions have been performed in the past by human operators. To increase the speed of response, to relieve the operators from mundane tasks, to protect them from hazards, high degree of autonomy is desired. To achieve this, high level decision making techniques for reasoning under uncertainty and taking actions must be utilized. These techniques, if used by humans, may be attributed to intelligent behavior. Hence, one way to achieve high degree of autonomy is to utilize high level decision making techniques, intelligent methods, in the autonomous controller. In our view, *higher autonomy is the objective, and intelligent controllers are one way to achieve it.* The need for quantitative methods to model and analyze the dynamical behavior of such autonomous systems presents significant challenges well beyond current capabilities. It is clear that the development of autonomous controllers requires significant interdisciplinary research effort as it integrates concepts and methods from areas such as Control, Identification, Estimation, and Communication Theory, Computer Science, Artificial Intelligence, and Operations Research.

Control systems have a long history. Mathematical modeling has played a central role in its development

[1] Presented at ISPE '95, Snowmass Village, Colorado, July 9-14, 1995.

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

in the last century and today conventional control theory is based on firm theoretical foundations. Designing control systems with higher degrees of autonomy has been a strong driving force in the evolution of control systems for a long time. What is new today is that with the advances of computing machines we are closer to realizing highly autonomous control systems than ever before. One of course should never ignore history but learn from it. For this reason, a brief outline of conventional control system history and methods is given below.

Conventional Control - Evolution and Quest for Autonomy

The first feedback device on record was the water clock invented by the Greek Ktesibios in Alexandria Egypt around the 3rd century B.C. This was certainly a successful device as water clocks of similar design were still being made in Baghdad when the Mongols captured the city in 1258 A.D.! The first mathematical model to describe plant behavior for control purposes is attributed to J.C. Maxwell, of the Maxwell equations' fame, who in 1868 used differential equations to explain instability problems encountered with James Watt's flyball governor; the governor was introduced in 1769 to regulate the speed of steam engine vehicles. Control theory made significant strides in the past 120 years, with the use of frequency domain methods and Laplace transforms in the 1930s and 1940s and the development of optimal control methods and state space analysis in the 1950s and 1960s. Optimal control in the 1950s and 1960s, followed by progress in stochastic, robust and adaptive control methods in the 1960s to today, have made it possible to control more accurately significantly more complex dynamical systems than the original flyball governor.

When J.C Maxwell used mathematical modeling and methods to explain instability problems encountered with James Watt's flyball governor, he demonstrated the importance and usefulness of mathematical models and methods in understanding complex phenomena and signaled the beginning of mathematical system and control theory. It also signalled the end of the era of intuitive invention. The flyball governor worked fine for a long time meeting the needs. As time progressed and more demands were put on the device there came a point when better and deeper understanding of the device was necessary as it started exhibiting some undesirable and unexplained behavior, in particular oscillations. This is quite typical of the situation in man made systems even today. Similarly to the flyball governor, one can rely on systems developed based mainly on intuitive inventions so much. To be able to control highly complex and uncertain systems we need deeper understanding of the processes involved and systematic design methods, we need quantitative models and design techniques. Such need is quite apparent in Intelligent Autonomous control systems and in particular in Hybrid Control systems; this is discussed further below.

Conventional control design : Conventional control systems are designed today using mathematical models of physical systems. A mathematical model, which captures the dynamical behavior of interest, is chosen and then control design techniques are applied, aided by Computer Aided Design (CAD) packages, to design the mathematical model of an appropriate controller. The controller is then realized via hardware or software and it is used to control the physical system. The procedure may take several iterations. The mathematical model of the system must be "simple enough" so that it can be analyzed with available mathematical techniques, and "accurate enough" to describe the important aspects of the relevant dynamical behavior. It approximates the behavior of a plant in the neighborhood of an operating point.

The control methods and the underlying mathematical theory were developed to meet the ever increasing control needs of our technology. The need to achieve the demanding control specifications for increasingly complex dynamical systems has been addressed by using more complex mathematical models and by developing more sophisticated design algorithms. The use of highly complex mathematical models however, can seriously inhibit our ability to develop control algorithms. Fortunately, simpler plant models, for example linear models, can be used in the control design; this is possible because of the feedback used in control which can tolerate significant model uncertainties. Controllers can for example be designed to meet the specifications around an operating point, where the linear model is valid and then via a scheduler a controller emerges which can accomplish the control objectives over the whole operating range. This is in fact the method typically used for aircraft flight control and it is a method to design fixed controllers for certain classes of nonlinear systems. When the uncertainties in the plant and environment are large, the fixed

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

feedback controllers may not be adequate, and adaptive controllers are used. Note that adaptive control in conventional control theory has a specific and rather narrow meaning. In particular it typically refers to adapting to variations in the constant coefficients in the equations describing the linear plant; these new coefficient values are identified and then used, directly or indirectly, to reassign the values of the constant coefficients in the equations describing the linear controller. Adaptive controllers provide for wider operating ranges than fixed controllers and so conventional adaptive control systems can be considered to have higher degrees of autonomy than control systems employing fixed feedback controllers. There are many cases however where conventional adaptive controllers are not adequate to meet the needs and novel methods are necessary.

Intelligent Control for High Autonomy Systems

There are cases where we need to significantly increase the operating range. We must be able to deal effectively with significant uncertainties in models of increasingly complex dynamical systems in addition to increasing the validity range of our control methods. We need to cope with significant unmodelled and unanticipated changes in the plant, in the environment and in the control objectives. This will involve the use of intelligent decision making processes to generate control actions so that certain performance level is maintained even though there are drastic changes in the operating conditions.

In view of the above it is quite clear that in the control of systems there are requirements today that cannot be successfully addressed with the existing conventional control theory. They mainly pertain to the area of uncertainty, present because of poor models due to lack of knowledge, or due to high level models used to avoid excessive computational complexity.

The need to use intelligent autonomous control stems from the need for an increased level of autonomous decision making abilities in achieving complex control tasks. Note that intelligent methods are not necessary for increase in the control system autonomy. It is possible to attain higher degrees of autonomy by using methods that are not considered intelligent. It appears however that to achieve the highest degrees of autonomy, intelligent methods are necessary.

2 Defining Intelligent Control

It is perhaps of interest at this point to further explain some of the differences between conventional and intelligent control, to briefly discuss the meaning of the term control in intelligent control and to comment on the use of the term intelligent. In this we follow a recent report [4].

Conventional and Intelligent Control

The term "conventional (or traditional) control" is used here to refer to the theories and methods that were developed in the past decades to control dynamical systems, the behavior of which is primarily described by differential and difference equations. Note that this mathematical framework may not be general enough in certain cases. In fact it is well known that there are control problems that cannot be adequately described in a differential/difference equations framework. Examples include discrete event systems in manufacturing and communication, the study of which has led to the use of automata and queuing theories in the control of systems.

In the minds of many people, particularly outside the control area, the term "intelligent control" has come to mean some form of control using fuzzy and/or neural network methodologies. This perception has been reinforced by a number of articles and interviews in the nonscientific literature. However intelligent control does not restrict itself only to those methodologies. The fact is that there are problems of control which cannot be formulated and studied in the conventional differential/difference equation mathematical framework. To address these problems in a systematic way, a number of methods have been developed that are collectively known as intelligent control methodologies.

There are significant differences between conventional and intelligent control and some of them are described below. It is worth remembering at this point that intelligent control uses conventional control methods to solve "lower level" control problems and that conventional control is included in the area of intelligent control. Intelligent control attempts to build upon and enhance the conventional control methodologies to solve new challenging control problems.

The word control in "intelligent control" has different, more general meaning than the word control in "conventional control". First, the processes of interest are more general and may be described, for example by either discrete event system models or differential/difference equation models or both. This has led to the development of theories for hybrid control systems, that study the control of continuous-state dynamic processes by discrete-state sequential machines. In addition to the more general processes considered in intelligent control, the control objectives can also be more general. For example, "replace part A in satellite" can be the general task for the controller of a space robot arm; this is then decomposed into a number of subtasks, several of which may include for instance "follow a particular trajectory", which may be a problem that can be solved by conventional control methodologies. To attain such control goals for complex systems over a period of time, the controller has to cope with significant uncertainty that fixed feedback robust controllers or adaptive controllers cannot deal with. Since the goals are to be attained under large uncertainty, fault diagnosis and control reconfiguration, adaptation and learning are important considerations in intelligent controllers. It is also clear that task planning is an important area in intelligent control design. So the control problem in intelligent control is an enhanced version of the problem in conventional control. It is much more ambitious and general. It is not surprising then that these increased control demands require methods that are not typically used in conventional control. The area of intelligent control is in fact interdisciplinary, and it attempts to combine and extend theories and methods from areas such as control, computer science and operations research to attain demanding control goals in complex systems.

Note that the theories and methodologies from the areas of operations research and computer science cannot, in general be used directly to solve control problems, as they were developed to address different needs; they must first be enhanced and new methodologies need to be developed in combination with conventional control methodologies, before controllers for very complex dynamical systems can be designed in systematic ways. Also traditional control concepts such as stability may have to be redefined when, for example, the process to be controlled is described by discrete event system models; and this issue is being addressed in the literature. Concepts such as reachability and deadlock developed in operations research and computer science are useful in intelligent control, when studying planning systems. Rigorous mathematical frameworks, based for example on predicate calculus are being used to study such questions. However, in order to address control issues, these mathematical frameworks may not be convenient and they must be enhanced or new ones must be developed to appropriately address these problems. This is not surprising as the techniques from computer science and operations research are primarily analysis tools developed for nondynamic systems, while in control, synthesis techniques to design real-time feedback control laws for dynamic systems are mainly of interest. In view of this discussion, it should be clear that intelligent control research, which is mainly driven by applications has a very important and challenging theoretical component. Significant theoretical strides must be made to address the open questions and control theorists are invited to address these problems. The problems are nontrivial, but the pay-off is very high indeed.

As it was mentioned above, the word control in intelligent control has a more general meaning than in conventional control; in fact it is closer to the way the term control is used in every day language. Because intelligent control addresses more general control problems that also include the problems addressed by conventional control, it is rather difficult to come up with meaningful bench mark examples. Intelligent control can address control problems that cannot be formulated in the language of conventional control. To illustrate, in a rolling steel mill, for example, while conventional controllers may include the speed (rpm) regulators of the steel rollers, in the intelligent control framework one may include in addition, fault diagnosis and alarm systems; and perhaps the problem of deciding on the set points of the regulators, that are based on the sequence of orders processed, selected based on economic decisions, maintenance schedules, availability of machines etc. All these factors have to be considered as they play a role in controlling the whole production process which is really the overall goal.

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

Research areas relevant to intelligent control, in addition to conventional control include areas such as planning, learning, search algorithms, hybrid systems, fault diagnosis and reconfiguration, automata, Petri nets, neural nets and fuzzy logic. In addition, in order to control complex systems, one has to deal effectively with the computational complexity issue; this has been in the periphery of the interests of the researchers in conventional control, but now it is clear that computational complexity is a central issue, whenever one attempts to control complex systems.

It is appropriate at this point to briefly comment on the meaning of the word intelligent in "intelligent control". Note that the precise definition of "intelligence" has been eluding mankind for thousands of years. More recently, this issue has been addressed by disciplines such as psychology, philosophy, biology and of course by artificial intelligence (AI); note that AI is defined to be the study of mental faculties through the use of computational models. No consensus has emerged as yet of what constitutes intelligence. The controversy surrounding the widely used IQ tests also points to the fact that we are well away from having understood these issues.

Some comments on the term "intelligent control" are now in order. Intelligent controllers are envisioned emulating human mental faculties such as adaptation and learning, planning under large uncertainty, coping with large amounts of data etc in order to effectively control complex processes; and this is the justification for the use of the term intelligent in intelligent control, since these mental faculties are considered to be important attributes of human intelligence. Certainly the term intelligent control has been abused and misused in recent years by some, and this is of course unfortunate. Note however that this is not the first time, nor the last that terminology is used to serve one's purpose. Intelligent control is certainly a catchy term and it is used (and misused) with the same or greater abundance by some, as for example the term optimal has been used (or misused) by others; of course some of the most serious offenses involve the word "democracy"! For better or worse, the term intelligent control is used by many. An alternative term is "autonomous (intelligent) control". It emphasizes the fact that an intelligent controller typically aims to attain higher degrees of autonomy in accomplishing and even setting control goals, rather than stressing the (intelligent) methodology that achieves those goals. On the other hand, "intelligent control" is only a name that appears to be useful today. In the same way the "modern control" of the 60's has now become "conventional (or traditional) control", as it has become part of the mainstream, what is called intelligent control today may be called just "control" in the not so distant future. What is more important than the terminology used are the concepts and the methodology, and whether or not the control area and intelligent control will be able to meet the ever increasing control needs of our technological society.

3 An Approach to Hybrid Control Systems Modeling and Design

Hybrid control systems contain two distinct types of systems, systems with continuous dynamics and systems with discrete dynamics, that interact with each other. Their study is essential in designing sequential supervisory controllers for continuous systems, and it is central in designing intelligent control systems with high degree of autonomy [4, 19]. Our group has made a number of contributions in this area [18-27].

Hybrid control systems typically arise when continuous processes interact with, or are supervised by, sequential machines. Examples of hybrid control systems are common in practice and are found in such applications as flexible manufacturing, chemical process control, electric power distribution, and computer communication networks. A simple example of a hybrid control system is the heating and cooling system of a typical home. The furnace and air conditioner, along with the heat flow characteristics of the home, form a continuous-time system which is to be controlled. The thermostat is a simple discrete event system which basically handles the symbols *too hot, too cold* and *normal*. The temperature of the room is translated into these representations in the thermostat and the thermostat's response is translated back to electrical currents which control the furnace, air conditioner, blower, etc.

The hybrid control systems of interest here consist of a continuous-state system to be controlled, also called the plant, and a discrete-state controller connected to the plant via an interface. It is generally

assumed that the dynamic behavior of the plant is governed by a set of known nonlinear ordinary differential equations, although our development [18] is based on the state trajectories of the plant rather than the particular mechanism in force that generated those trajectories; that is, our results apply, under certain assumptions, to systems where a differential or difference equation description is not known of may not even exist. The plant contains all continuous-state subsystems of the hybrid control system, such as any conventional continuous-state controller that may have been developed, a clock if time and synchronous operations are to be modeled, etc. The controller is an event driven, asynchronous discrete event system (DES), described here by a finite state automaton. The hybrid control system also contains an interface that provides the means for communication between the continuous-state plant and the DES controller. The interface receives information from the plant in the form of a measurement of a continuous variable, such as the continuous state, and issues a sequence of symbols to the DES controller. It also receives a sequence of control symbols from the controller and issues (piecewise) continuous input commands to the plant.

The interface plays a key role in determining the dynamic behavior of the hybrid control system. Understanding how the interface affects the properties of the hybrid system is one of the fundamental issues in the theory of hybrid control systems. In [18-27], the interface has been chosen to be simply a partitioning of the state space. If memory is necessary to derive an effective control, it is included in the DES controller and not in the interface. Also the piecewise continuous command signal issued by the interface is simply a staircase signal, not unlike the output of a zero-order hold in a digital control system.Including an appropriate continuous system at (the input of) the plant, signals such as ramps, sinusoids, etc., can be generated if desired. So the simple interface used is without loss of generality and it allows analysis of the hybrid control system, with development of properties such as controllability [25], stability [21] and determinism [18,19], in addition to control design methodologies, [18,27]. The simplicity of the interface with the resulting benefits in identifying central issues and concepts in hybrid control systems is perhaps the main characteristic of the approach in [18-27]. It is also what has been distinguishing this approach from other approaches (with more complex interfaces, or with restrictions on the class of systems studied) since early versions of the model first appeared in 1991.

In general the design of the interface depends not only on the plant to be controlled, but also on the control policies available, as well as on the control goals. Depending on the control goals, one may need, for example, detailed state information or not; the corresponds to small or large regions in the partition of the measured signal space (or greater of lower granularity). This is, of course, not surprising as it is rather well known that to stabilize a system, for example, requires less detailed information about the system's dynamic behavior than to do tracking. The fewer the distinct regions in the partitioned signal space, the simpler (fewer states) the resulting DES plant model and the simpler the DES controller design. Since the systems to be controlled via hybrid controllers are typically complex, it is important to make every effort to use only the necessary information to derive the control goals; as this leads to simpler interfaces that issue only the necessary number of distinct symbols, and to simpler DES plant models and controllers. The question of systematically determining the minimum amount of information needed from the plant in order to achieve particular control goals via a number of specialized control policies is an important and still open question; recent results in [27] partially resolves this question.

In [18] first a model in presented that is developed for hybrid control system analysis and design. This model is general and it is characterized by a simple interface. This allows the development of a DES plant model, including the continuous-state plant together with the interface, the properties of which are discussed. In particular, DES plant properties such as determinism, quasideterminism and observability are introduced and discussed, as well as the notion of controllability, which is an extension of the corresponding controllability notion in logical DES's. Given control goals, such as forbidden states, a design methodology is presented that leads to a DES controller. This design method is an extension of the corresponding method in logical DES. The approach in [18] is briefly outlined below in Sections 3.1-3.5. A thermostat and a distillation column example are given in Sections 3.3 and 3.5. Also a brief discription of the control methodology of [27] is given in Section 3.6 below. An underwater vehicle example is given in Section 3.7.

It is important to note that the core issue in hybrid control systems is the way in which the interface

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

relates the continuous plant to the DES plant. This issue is rather fundamental and quite difficult to resolve. Its solution will depend on the answer to the question of what is the minimum amount of information about the plant that will allow, with the controls available, the accomplishment of the control objectives.

Note that the approach taken in [18,27], where a DES plant model is derived to describe the dynamic behavior of the continuous plant together with the interface, is similar to one of the main approaches in digital control (sampled data) systems. There the continuous-time plant is combined with the A/D and D/A converters, typically a sampler and a zero-order hold, to obtain a discrete-time plant model. Then the discrete-time controller is designed using discrete-time control design techniques. Of course attention should be paid to the intersample behavior of the continuous system and the selection of the sampling period. There is an alternative approach to digital control design that may lead to an alternative approach to hybrid controller design. In this approach, the discrete-time controller is an approximation of an existing continuous-time controller; that is the control design is carried out in the continuous domain and then appropriate approximations (similar to the ones used in digital filter design) are employed to derive the discrete-time controller. This is a convenient design approach, however it does not take full advantage of the discrete nature of the controller, since the discrete controller can be, at best, as good as the continuous controller and behavior such as deadbeat is not attained; furthermore this approach typically requires higher sampling rates than the previous ones.

Finally in [18] and below, finite automata models were used to represent the DES plant model. This was done because the purpose of this work was to use the tools from the logical theory of DES, with appropriate modifications, to design controllers for hybrid systems. Note that the controller used is also a finite automaton. It is of course possible to use other models to represent the DES plant, such as Petri nets.

3.1 Hybrid Control System Modeling

A hybrid control system, can be divided into three parts, the plant, interface, and controller as shown in Figure 1. In this model, the plant represents the continuous-time components of the system, while the controller represents the discrete-event portions. The interface is the necessary mechanism by which the former two communicate. The models used for each of these three parts, as well as the way they interact are now described.

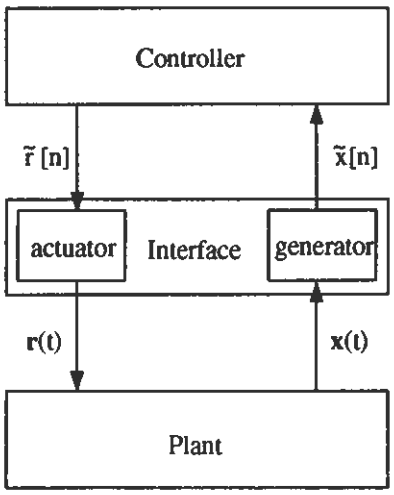

Figure 1: Hybrid Control System

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

3.1.1 Plant

The plant is the part of the model which represents the entire continuous portion of the hybrid control system. The distinguishing feature of the plant is that it has a continuous state space, where the states take on values that are real numbers, and evolve with time according to a set of differential equations. Motivated by tradition, this part of the model is referred to as the plant but since it contains all the continuous dynamics, it can also contain a conventional, i.e. continuous-time, controller.

The plant is a nonlinear, time-invariant system represented by a set of ordinary differential equations.

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{r}(t)) \tag{1}$$

where $\mathbf{x}(t) \in \mathbf{X}$ and $\mathbf{r}(t) \in \mathbf{R}$ are the state and input vectors respectively, and $\mathbf{X} \subset \Re^n, \mathbf{R} \subset \Re^m$, with $t \in (a, b)$ some time interval. For any $\mathbf{r}(t) \in \mathbf{R}$, the function $f : \mathbf{X} \times \mathbf{R} \rightarrow \mathbf{X}$ is continuous in $\mathbf{x}$ and meets the conditions for existence and uniqueness of solutions for initial states, $\mathbf{x}_0 \in \mathbf{X}$. Note that the plant input and state are continuous-time vector valued signals. Boldface letters are used here to denote vectors and vector valued signals.

3.1.2 Controller

The controller is a discrete event system which is modeled as a deterministic automaton. This automaton is specified by a quintuple, $(\tilde{S}, \tilde{X}, \tilde{R}, \delta, \phi)$, where $\tilde{S}$ is the set of states, $\tilde{X}$ is the set of *plant symbols*, $\tilde{R}$ is the set of *controller symbols*, $\delta : \tilde{S} \times \tilde{X} \rightarrow \tilde{S}$ is the state transition function, and $\phi : \tilde{S} \rightarrow \tilde{R}$ is the output function. The symbols in set $\tilde{R}$ are called controller symbols because they are generated by the controller. Likewise, the symbols in set $\tilde{X}$ are called plant symbols and are generated based on events in the plant. The action of the controller is described by the equations

$$\tilde{s}[n] = \delta(\tilde{s}[n-1], \tilde{x}[n]) \tag{2}$$

$$\tilde{r}[n] = \phi(\tilde{s}[n]) \tag{3}$$

where $\tilde{s}[n] \in \tilde{S}, \tilde{x}[n] \in \tilde{X}$, and $\tilde{r}[n] \in \tilde{R}$. The index n is analogous to a time index in that it specifies the order of the symbols in the sequence. The input and output signals associated with the controller are sequences of symbols.

Tildes are used to indicate a symbol valued set or sequence. For example, $\tilde{X}$ is the set of plant symbols and $\tilde{x}[n]$ is the nth symbol of a sequence of plant symbols. Subscripts are also used, e.g. $\tilde{x}_i$ which denotes the ith member of the symbol alphabet $\tilde{X}$.

3.1.3 Interface

The controller and plant cannot communicate directly in a hybrid control system because each utilizes a different type of signal. Thus an interface is required which can convert continuous-time signals to sequences of symbols and vice versa. The way that this conversion is accomplished determines, to a great extent, the nature of the overall hybrid control system. The interface consists of two simple subsystems, the generator and actuator.

The generator issues symbols to the controller and plays the role of a quantizer of the signals analogous to an A/D converter (sampler) in a digital control system. The actuator injects the appropriate control signal into the plant and it is analogous to a D/A converter (typically a zero-order hold) in a digital control system. The generator and the actuator perform, however, more general functions than their counterparts in a typical digital control system.

The *generator* is the subsystem of the interface which converts the continuous-time output (state) of the plant to an asynchronous, symbolic input for the controller. To perform this task, two processes must be

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

in place. First, a triggering mechanism is required which will determine when a plant symbol should be generated. Second, a process to determine which particular plant symbol should be generated is required.

In the generator, the triggering mechanism is based on the idea of *plant events*. A plant event is simply an occurrence in the plant, an idea borrowed from the field of discrete event systems. In the case of hybrid control, a plant event is defined by specifying a hypersurface which separates the plant's state space. The plant event occurs whenever the plant state trajectory crosses this hypersurface. The basis for this definition of a plant event is that an event is considered to be the realization of a specified condition. This condition can be given as an open region of the state space, separated from the remainder of the state space by a hypersurface. If the state crosses the hypersurface into the given open region, the event has occurred. Mathematically, the set of plant events recognized by the generator is determined by a set of smooth functionals, $\{h_i : \Re^n \rightarrow \Re, i \in I\}$, defined on the state space of the plant. Each functional must satisfy the condition,

$$\nabla_x h_i(\xi) \neq 0, \forall \xi \in \mathcal{N}(h_i) \tag{4}$$

which ensures that the null space of the functional, $\mathcal{N}(h_i) = \{\xi \in \Re^n : h_i(\xi) = 0\}$, forms an $n-1$ dimensional smooth hypersurface separating the state space.

Let the sequence of plant events be denoted by e, where $e[n] = i$ means that the nth plant event was triggered by crossing the hypersurface defined by h_i. Let the sequence of plant event instants be given by τ_e, where $\tau_e[n]$ is the time of the nth plant event and $\tau_e[0] = 0$. By definition, these sequences satisfy the following conditions.

$$e[n] = i \Rightarrow \begin{cases} h_i(\mathbf{x}(\tau_e[n])) = 0 \\ \exists\ \delta_1 > 0 \text{ s.t. } \forall \epsilon, 0 < \epsilon < \delta_1, \pm h_i(\mathbf{x}(\tau_e[n] + \epsilon)) < 0 \\ \exists\ \delta_2 > 0 \text{ s.t. } \forall \epsilon, 0 < \epsilon < \delta_2, \pm h_i(\mathbf{x}(\tau_e[n] - \delta_2)) > 0, \pm h_i(\mathbf{x}(\tau_e[n] - \epsilon)) \geq 0 \end{cases} \tag{5}$$

and

$$\forall n, \tau_e[n] < \tau_e[n+1] \vee (\tau_e[n] = \tau_e[n+1] \wedge e[n] < e[n+1]) \tag{6}$$

The first group, equation (5), contains three conditions: (i) at the time of the plant event the plant state lies on the triggering hypersurface, (ii) immediately after the event the plant state lies on the negative (positive) side of the triggering hypersurface, and (iii) prior to reaching the triggering hypersurface the plant state lied on the negative (positive) side. The fourth condition, equation (6) concerns the ordering of the sequences. It requires that plant events be ordered chronologically and simultaneous plant events be ordered according to their number, that is the value of i.

An alternative, and perhaps simpler, way of expressing the conditions of (5) is by the condition, $h_i(\mathbf{x}(t)) = 0, \frac{d}{dt} h_i(\mathbf{x}(t)) \neq 0$. In this case the assumption is made that the derivative is nonzero, that is $\frac{d}{dt} h_i(\mathbf{x}(t)) \neq 0$ at the crossing. Note however that these conditions do not take into account the case where the crossing occurs exactly at an inflection point. When $\frac{d}{dt} h_i(\mathbf{x}(t)) = 0$, one must use (5).

A plant event will only cause a plant symbol to be generated if the hypersurface is crossed in the negative direction. The reason for this is that in many applications sensors only detect when a threshold is crossed in one direction, e.g. a thermostat. When the hypersurface is crossed in the opposite direction the event is silent, and for convenience, assume that a null symbol, ϵ, is generated. At each time in the sequence $\tau_e[n]$, a plant symbol is generated according to the function $\alpha_i : \mathcal{N}(h_i) \rightarrow \tilde{X}$. The sequence of plant symbols can now be defined as

$$\tilde{x}[n] = \begin{cases} \alpha_i(\mathbf{x}(\tau_e[n])) & \text{nonsilent event} \\ \epsilon & \text{silent event} \end{cases} \tag{7}$$

where i identifies the hypersurface which was crossed. Alternatively one could select the interface to generate information bearing symbols when crossed in either direction. This is not as flexible and leads to some difficulties in the analysis of the hybrid control system. Our definition in (7) is more general and can easily model this case. See the thermostat example in 3.3 below.

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

The actuator converts the sequence of controller symbols to a plant input signal, using the function $\gamma : \tilde{R} \rightarrow \Re^m$, as follows.

$$\mathbf{r}(t) = \sum_{n=0}^{\infty} \gamma(\tilde{r}[n]) I(t, \tau_c[n], \tau_c[n+1]) \tag{8}$$

where $I(t, \tau_1, \tau_2)$ is a characteristic function taking on the value of unity over the time interval $[\tau_1, \tau_2)$ and zero elsewhere. $\tau_c[n]$ is the time of the nth control symbol which is based on the sequence of plant symbol instants, defined in equation 5, according to

$$\tau_c[n] = \tau_e[n] + \tau_d \tag{9}$$

where τ_d is the total delay associated with the interface and controller. Following the occurrence of a plant event, it takes a time of τ_d for a new control policy to be used by the plant. It will be assumed that $\tau_e[n] < \tau_c[n] < \tau_e[n+1]$.

The plant input, $\mathbf{r}(t)$, can only take on certain constant values, where each value is associated with a particular controller symbol. Thus the plant input is a piecewise constant signal which may change only when a controller symbol occurs.

Remark The model presented uses the plant state, $\mathbf{x}(t)$, as the feedback signal. When the state is not available for measurement, a plant output signal, $\mathbf{y}(t)$, can also be used. This case is not treated in this paper.

Remark In the interface a delay, τ_d was introduced. The presence of the delay is necessary for two reasons. First, from a practical point of view, the generator will not be able to detect an event until after the state has actually crossed the hypersurface. Second, if a chattering control strategy is used, the delay partially determines the chattering period. It is of course possible for two plant events to occur within the period of a single delay. In such a case each event will be acted upon, in turn, τ_d after it occurs. In this way the delay can pose a problem for the controller, but it is unavoidable as real systems cannot react instantaneously.

3.2 DES Plant Model

In a hybrid control system, the plant taken together with the actuator and generator, behaves like a discrete event system; it accepts symbolic inputs via the actuator and produces symbolic outputs via the generator. This situation is somewhat analogous to the way a continuous-time plant, equipped with a zero order hold and a sampler, "looks" like a discrete-time plant. In a hybrid control system, the DES which models the plant, actuator, and generator is called the *DES plant model.* From the DES controller's point of view, it is the DES plant model which is controlled.

It must be pointed out that the DES plant model is an approximation of the actual plant-actuator-generator combination. Since the DES plant model has a discrete state space, it cannot model the exact behavior of a system which has a continuous state space [18].

The DES plant model is a nondeterministic automaton, represented mathematically by a quintuple, $(\tilde{P}, \tilde{X}, \tilde{R}, \psi, \lambda)$. $\tilde{P}$ is the set of states, $\tilde{X}$ is the set of plant symbols, and $\tilde{R}$ is the set of control symbols. $\psi : \tilde{P} \times \tilde{R} \rightarrow 2^{\tilde{P}}$ is the state transition function, for a given DES plant state and a given control symbol, it specifies which DES plant states are enabled. The output function, $\lambda : \tilde{P} \times \tilde{P} \rightarrow 2^{\tilde{X}}$, maps the previous and current state to a set of plant symbols. The set of DES plant model states, $\tilde{P}$, is based upon the set of hypersurfaces realized in the generator.

Formally, the set of DES plant states is defined as a set of equivalence classes on the state space of the plant. Details of the extraction of the DES plant model can be found in [18].

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

The DES plant model is an approximation of the actual hybrid system. Specifically, the state of the DES plant model is an approximation of the state of the continuous plant. As a result, the future behavior cannot be determined uniquely, in general, from knowledge of the DES plant state. The approach taken here is to incorporate all the possible future behaviors into the DES plant model. From a control point of view this means that if undesirable behaviors can be eliminated from the DES plant (through appropriate control policies) then these behaviors can likewise be eliminated from the actual system. On the other hand, just because a control policy permits a given behavior in the DES plant, is no guarantee that that behavior will occur in the actual system; this phenomenon is due to the nondeterminism in the DES plant model.

3.3 Example - Furnace and Thermostat

Consider a system made up of a thermostat, room, and heater. If the thermostat is set at 70°F, and assuming it is colder outside, the system behaves as follows. If the room temperature falls below 70 degrees the heater starts and remains on until the room temperature exceeds 75 degrees at which point the heater shuts off. For simplicity, we will assume that when the heater is on it produces heat at a constant rate.

The plant in this hybrid control system is made up of the heater and room, and it can be modeled with the following differential equation.

$$\dot{\mathbf{x}}(t) = .0025(T_O - \mathbf{x}(t)) + .02\mathbf{r}(t) \tag{10}$$

Here $\mathbf{x}(t)$ is the room temperature, T_O is the outside temperature, and $\mathbf{r}(t)$ is the voltage into the heater. Temperatures are in degrees Fahrenheit and time is in minutes.

The generator and controller are found in the thermostat. The generator partitions the state space with two hypersurfaces.

$$h_1(\mathbf{x}) = \mathbf{x} - 70 \tag{11}$$

$$h_2(\mathbf{x}) = -\mathbf{x} + 75 \tag{12}$$

The first hypersurface detects when the temperature falls below 70°F and the second detects when the temperature rises above 75°F. The events are represented symbolically to the controller.

$$\alpha_1(\xi) = cold \tag{13}$$

$$\alpha_2(\xi) = hot \tag{14}$$

It is common to see bimetallic strips performing this function in an actual thermostat, where the band is physically connected to the controller. The controller has two states (typically it is just a switch in the thermostat) as illustrated in Figure 2. The output function of the thermostat controller provides two

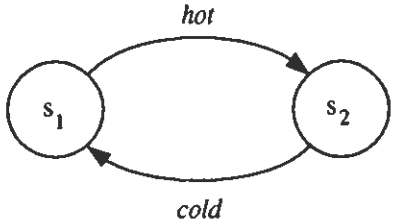

Figure 2: Controller in Thermostat

controller symbols, *on* and *off*.

$$\phi(\tilde{s}_1) = on \qquad \phi(\tilde{s}_2) = off \tag{15}$$

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

Finally the actuator converts the symbolic output of the controller to a continuous input for the plant.

$$\gamma(on) = 110 \qquad \gamma(off) = 0 \tag{16}$$

In this case the plant input is the voltage supply to the heater, 0 or 110 volts. Physically, the symbolic output from the controller could be a low voltage signal, say 0 or 12 volts, or perhaps a pneumatic signal.

The thermostat/heater example has a simple DES plant model which is useful to illustrate how these models work. Figure 3 shows the DES plant model for the heater/thermostat. The convention for labeling the arcs is to list the controller symbols which enable the transition followed by a "/" and then the plant symbols which can be generated by the transition. Notice that two of the transitions are labeled with null symbols, ϵ. This reflects the fact that nothing actually happens in the system at these transitions. When the controller receives a null symbol it remains in the same state and reissues the current controller symbol. This is equivalent to the controller doing nothing, but it serves to keep all the symbolic sequences, $\tilde{s}, \tilde{p}$, etc., in phase with each other.

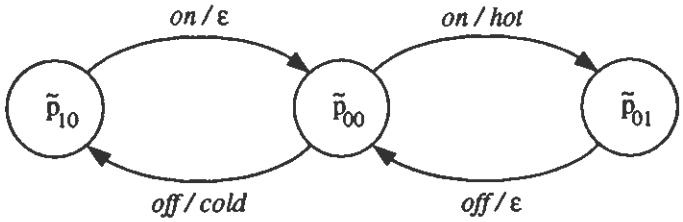

Figure 3: DES Plant for Thermostat/Heater

3.4 Supervisory Control via DES Plant Models

In this section, we use the language generated by the DES plant to examine the controllability of the hybrid control system. This work builds upon the work done by Ramadge and Wonham on the controllability of discrete event systems in a logical framework. See [18] for a detailed discussion. Here we adapt several of those results and apply them to the DES plant model obtained from a hybrid control system.

Before existing techniques, developed in the logical DES framework can be extended, certain differences must be dealt with. The Ramadge-Wonham model (RWM) consists of two interacting DES's called here the *RWM generator* and *RWM supervisor*. The RWM generator is analogous to the DES plant and the RWM supervisor is analogous to the DES controller. The RWM generator shares its name with the generator found in the hybrid control system interface but the two should not be confused. In the RWM, the plant symbols are usually referred to as "events", but we will continue to call them plant symbols to avoid confusion. The plant symbols in the RWM are divided into two sets, those which are controllable and those which are uncontrollable: $\tilde{X} = \tilde{X}_c \cup \tilde{X}_u$. A plant symbol being controllable means that the supervisor can prevent it from being issued by the RWM generator. When the supervisor prevents a controllable plant symbol from being issued, the plant symbol is said to be *disabled*. The plant symbols in $\tilde{X}_c$ can be individually disabled, at any time and in any combination, by a command from the RWM supervisor, while the plant symbols in $\tilde{X}_u$ can never be disabled. This is in contrast to our DES plant where each command (controller symbol) from the DES controller disables a particular subset of $\tilde{X}$ determined by the complement of the set given by the transition function, ψ. Furthermore, this set of disabled plant symbols depends not only on the controller symbol but also the present state of the DES plant. In addition, there is no guarantee that any arbitrary subset of $\tilde{X}$ can be disabled while the other plant symbols remain enabled.

The general inability to disable plant symbols individually is what differentiates the DES plant model from the automata of earlier frameworks.

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

Controllability and Supervisor Design

A DES is controlled by having various symbols disabled by the controller based upon the sequence of symbols which the DES has already generated. When a DES is controlled, it will generate a set of symbol sequences which lie in a subset of its language. If we denote this language of the DES under control as L_c then $L_c \subset L$.

It is possible to determine whether a given RWM generator can be controlled to a desired language. That is, whether it is possible to design a controller such that the RWM generator will be restricted to some target language K. Such a controller can be designed if K is prefix closed and

$$\bar{K}\bar{X}_u \cap L \subset \bar{K} \tag{17}$$

where $\bar{K}$ represents the set of all prefixes of K. A prefix of K is a sequence of symbols, to which another sequence can be concatenated to obtain a sequence found in K. A language is said to be prefix closed if all the prefixes of that language are also in the language.

When equation 17 is true for a given RWM generator, the desired language K is said to be controllable, and provided K is prefix closed, a controller can be designed which will restrict the generator to the language K. This condition requires that if an uncontrollable symbol occurs after the generator has produced a prefix of K, the resulting string must still be a prefix of K because the uncontrollable symbol cannot be prevented.

Since the DES plant model belongs to a slightly different class of automata than the RWM, another definition for controllable language which applies to the DES plant is used. An algorithm that generates the supremal controllable sublanguage and leads to a supervisor is derived. Full details can be found in [18].

3.5 Example - Distillation Column

This example uses the model of a two product distillation column with a single feed. A complete description of the non-linear model can be found in [Morari and Zafiriou, *Robust Process Control*, Prentice Hall, 1989]. Here a condensed description is given to show the source of the DES plant model and provide insight into the physical meaning of the states and events.

Figure 4 shows the distillation column. F represents the feed flow into the column, B is the flow of bottom product out of the column, x_B is the mole fraction of the light compound in the bottom product, D is the flow of distillate out of the column, and y_D is the mole fraction of light compound in the distillate. The boilup flow is denoted by V and the reflux flow by L. All units are in kmol's and minutes. The column can be controlled by setting the feed, boilup, and reflux. In general, the goal is to have a high level of light compound in the distillate ($y_D \rightarrow 1$) and a low level of light compound in the bottom product ($x_B \rightarrow 0$).

There are 40 trays stacked vertically in the column. The state consists of the mole fractions of light compound in the liquid of each tray. The states evolve according to the following equations.

$$\begin{aligned}
2\dot{x}_1 &= (L+F_L)x_2 - Vy_1 - Bx_1 \\
2\dot{x}_i &= (L+F_L)x_{i+1} + Vy_i - (L+F_L)x_i - Vy_i \\
2\dot{x}_{21} &= Lx_{22} + Vy_{20} - (L+F_L)x_{21} - Vy_{21} + F_L * x_F \\
2\dot{x}_{22} &= Lx_{23} + Vy_{21} - Lx_{22} - (V+F_V)y_{22} + F_V * y_F \\
2\dot{x}_j &= Lx_{j+1} + (V+F_V)y_j - Lx_j - (V+F_V)y_j \\
2\dot{x}_{41} &= (V+F_V)y_{40} - Lx_{41} - Dx_{41}
\end{aligned}$$

where $2 \leq i \leq 20$ and $23 \leq j \leq 40$. Trays 21 and 22 are special because they are below and above the feed location. Tray 41 is actually the condenser. The quantities y_i are the mole fractions of light compound in the vapor, given by

$$y_i = \frac{\alpha x_i}{1+(\alpha-1)x_i}$$

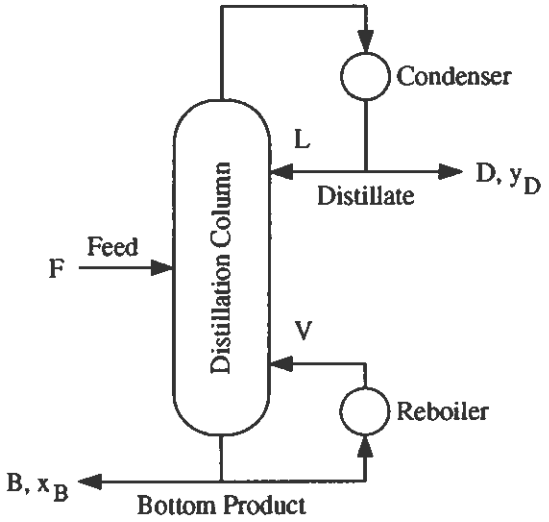

Figure 4: Distillation Column

where $\alpha = 1.5$ is relative volatility. Other quantities of interest are

$$\begin{aligned} F_L &= .6F \\ F_V &= F - F_L \\ x_F &= (.5F - F_V y_F)/F_L \end{aligned}$$

and the outputs are

$$\begin{aligned} D &= V + F_V - L \\ B &= L + F_L - V \\ y_D &= y_{41} \\ x_B &= x_1 \end{aligned}$$

To obtain a hybrid control system, appropriate control policies and plant symbols must be chosen. Their selection is based on our knowledge of the control goals and the design constraints, and it will determine the interface. Let the control policies be

$$\mathbf{r}(t) = \begin{bmatrix} L \\ V \\ F \end{bmatrix} \in \left\{ \begin{bmatrix} 9.5 \\ 10 \\ 1 \end{bmatrix}, \begin{bmatrix} 10 \\ 10 \\ 0.1 \end{bmatrix}, \begin{bmatrix} 9.5 \\ 10 \\ 2 \end{bmatrix}, \begin{bmatrix} 10 \\ 10 \\ 2 \end{bmatrix} \right\}$$

These input values correspond to $\tilde{r}_1, \tilde{r}_2, \tilde{r}_3$, and $\tilde{r}_4$. Next, plant symbols are defined based on events as follows.

$\tilde{x}_1$	$B + D$ falls below 2
$\tilde{x}_2$	$B + D$ exceeds 2
$\tilde{x}_3$	x_B falls below .13
$\tilde{x}_4$	x_B exceeds .13
$\tilde{x}_5$	x_B falls below .12
$\tilde{x}_6$	x_B exceeds .12
$\tilde{x}_7$	x_B falls below .08
$\tilde{x}_8$	x_B exceeds .08
$\tilde{x}_9$	y_D falls below .84
$\tilde{x}_{10}$	y_D exceeds .84
$\tilde{x}_{11}$	y_D falls below .85
$\tilde{x}_{12}$	y_D exceeds .85
$\tilde{x}_{13}$	y_D falls below .95
$\tilde{x}_{14}$	y_D exceeds .95

We would like to keep x_B below .13, y_D above .95, and the feed at 2. These conditions correspond to increased production of high purity products. Simulations reveal that given the available controls and events this is not possible, that is, even if the initial state is in this region, no available control policy will cause it to remain there. It is possible to drive the system close to this point, however. Specifically, our control goal shall be two-fold: first to drive the system near the ideal point and second to avoid having a high feed rate (2 kmol/min) when the system is not near the ideal point.

The distillation column is an example of a rather complex hybrid system. The generator was designed to recognize 14 different plant events. This leads to 32 distinct regions in the state space and therefore there are 32 DES plant states. Figure 5 shows the DES plant model. The two states labeled 'G' correspond to the desired operating regions of the system. This DES plant model was extracted by automating the testing process and implementing it on a computer.

A controller was obtained by automating the the procedure for finding the supremal controllable sub-language. The controller is shown in Figure refdccon. This controller drives the plant from the initial state to a loop containing the two good states. Notice that in this figure, the states of the controller have been labeled with the controller symbol which is generated by that state.

3.6 Invariant Based Approach

Here a methodology is presented to design the controller and the interface together based on the natural invariants of a plant described by

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{r}(t)) \tag{18}$$

where certain smoothness assumptions apply.

In particular, this section discusses the design of the generator, which is part of the interface, and the dsesign of the controller. We assume that the plant is given, the set of available control policies is given, and the control goals are specified as follows. Each control goal for the system is given as a starting set and a target set, each of which is an open subset of the plant state space. To realize the goal, the controller must be able to drive the plant state from anywhere in the starting set to somewhere in the target set using the available control policies. Generally, a system will have multiple control goals.

To successfully control the plant, the controller must know which control policy to apply and when to apply it. The controller receives all its information about the plant from the generator, and therefore the generator must be designed to provide that information which the controller requires.

We propose the following solution to this design problem. For a given target region, identify the states which can be driven to that region by the application of a single control policy. If the starting region is

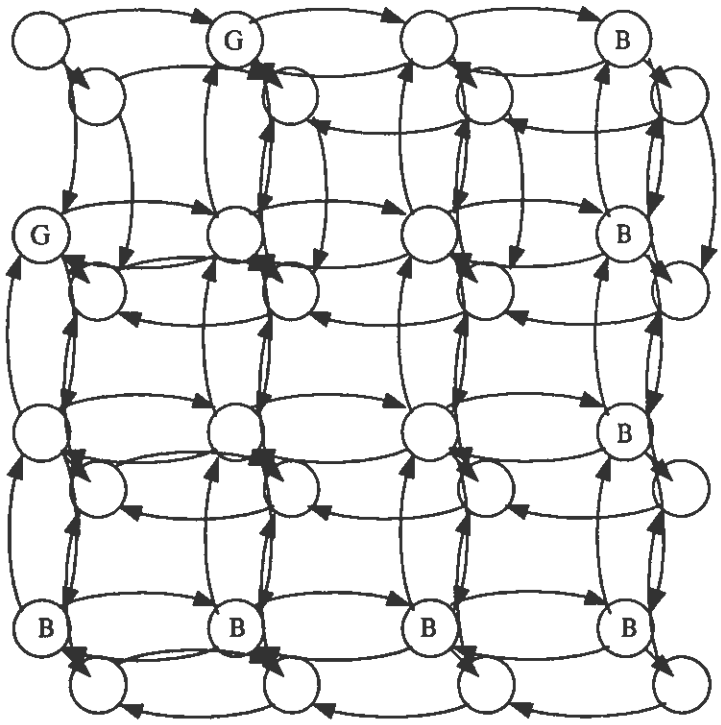

Figure 5: DES Plant for Distillation Column

contained within this set of states, the control goal is achievable via a single control policy. If not, then this new set of states can be used as a target region and the process can be repeated. This will result in a set of states which can be driven to the original target region with no more than two control policies applied in sequence. This process can be repeated until the set of states, for which a sequence of control policies exists to drive them to the target region, includes the entire starting region (provided the set of control policies is adequate as mentioned below).

When the regions have been identified, the generator is designed to tell the controller, via plant symbols, which region the plant state is currently in. The controller will then call for the control policy which drives the states in that region to the target region.

3.6.1 Generator Design

To describe the regions mentioned above, we use the concept of the flow. Let the flow for the plant (1) be given by $F_k : \mathbf{X} \times \mathbb{R} \rightarrow \mathbf{X}$, where

$$\mathbf{x}(t) = F_k(\mathbf{x}(0), t). \tag{19}$$

The flow represents the state of the plant after an elapsed time of t, with an initial state of $\mathbf{x}(0)$, and with a constant input of $\gamma(\tilde{r}_k)$. Since the plant is time invariant, there is no loss of generality when the initial state is defined at $t = 0$. The flow is defined over both positive and negative values of time. The flow can be extended over time using the forward flow function, $F_k^+ : \mathbf{X} \rightarrow \mathbb{P}(\mathbf{X}^n)$, and the backward flow function, $F_k^- : \mathbf{X} \rightarrow \mathbb{P}(\mathbf{X}^n)$, which are defined as follows.

$$F_k^+(\xi) = \bigcup_{t \geq 0} \{F_k(\xi, t)\} \tag{20}$$

$$F_k^-(\xi) = \bigcup_{t \leq 0} \{F_k(\xi, t)\} \tag{21}$$

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

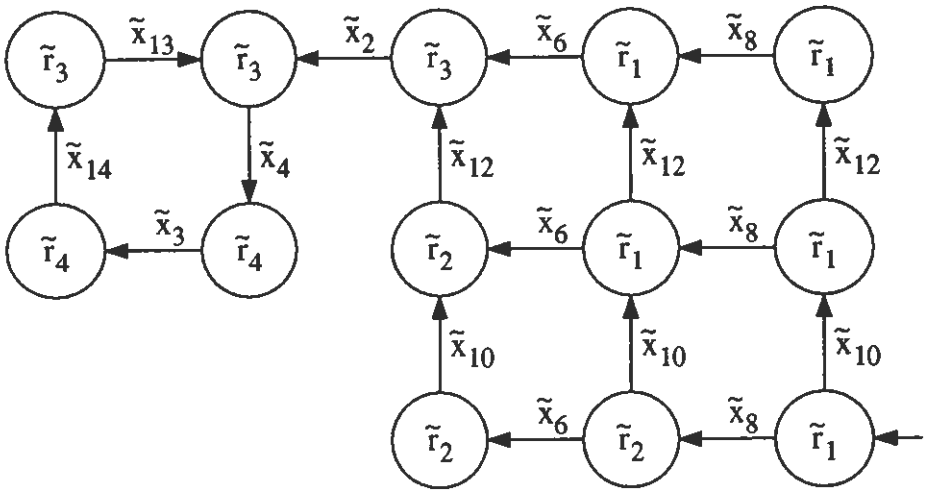

Figure 6: Sample Controller for Distillation Column

The backward and forward flow functions can be defined on an arbitrary set of states in the following natural way.

$$F_k^+(\mathbf{A}) = \bigcup_{\xi \in \mathbf{A}} \{F_k^+(\xi)\} \tag{22}$$

$$F_k^-(\mathbf{A}) = \bigcup_{\xi \in \mathbf{A}} \{F_k^-(\xi)\} \tag{23}$$

where $\mathbf{A} \subset \mathbf{X}$. For a target region, T, $F_k^-(T)$ is the set of initial states from which the plant can be driven to T with the input $\gamma(\tilde{r}_k)$. In addition, $F_k^+(T)$ is the set of states which can be reached with input $\gamma(\tilde{r}_k)$ and an initial state in T.

Now a generator design procedure can be described using the backward flow function. This is a preliminary procedure, upon which the final design method, developed subsequently, is based. For a given starting region, $S \subset \mathbf{X}$, and target region, $T \subset \mathbf{X}$, use the following algorithm.

1. If $S \subset T$, stop.
2. Identify the regions, $F_k^-(T), \forall \tilde{r}_k \in \tilde{R}$.
3. Let $T = \bigcup_{\tilde{r}_k \in \tilde{R}} F_k^-(T)$
4. Go to 1.

There are two problems associated with this algorithm as stated. First, it will not stop if there is no sequence of available control policies which will achieve the control goal, and second, actually identifying the regions given by the flow functions is quite involved. The first issue is related to the adequacy of the available control policies and will not be dealt with here. The second problem will be addressed. The difficulty in identifying a region given by a flow function is integrating over all the points in the target region. In the generator design procedure developed here, we will concentrate on finding a subset of the region $F_k^-(T)$, rather than the region itself. By definition, all the trajectories passing through $F_k^-(T)$ lead to the target region, T, and therefore all the trajectories found in a subset of $F_k^-(T)$ will also lead to the target.

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

Here, we will focus on identifying subsets of $F_k^-(T)$ which we call *common flow regions*. Common flow regions are bounded by invariant manifolds and an exit boundary. The invariant manifolds are used because the state trajectory can neither enter nor leave the common flow region through an invariant manifold. The exit boundary is chosen as the only boundary through which state trajectories leave the common flow region.

To design the generator, it is necessary to select the set of hypersurfaces, $\{h_i : \mathbf{X} \to \mathbb{R} \mid i \in I\}$ and the associated functions, $\{\alpha_i : \mathcal{N}(h_i) \to \tilde{R} \mid i \in I\}$, described in Section 3.1.3. These hypersurfaces make up the invariant manifolds and exit boundaries mentioned above, as well as forming the boundary for the target region(s).

A target region, T, is specified as

$$T = \{\xi \in \mathbf{X} : \forall i \in I_T, h_i(\xi) < 0\}, \tag{24}$$

where I_T is the index set indicating which hypersurfaces bound the target region. A common flow region, B, is specified as

$$B = \{\xi \in \mathbf{X} : h_i(\xi) < 0, h_e(\xi) > 0, \forall i \in I_B\}, \tag{25}$$

where I_B is an index set indicating which hypersurfaces form the invariant manifolds bounding B and h_e defines the exit boundary for B.

The goal, of course, is that B should include only states whose trajectories lead to the target region. Figure 7 shows an example of this where $I_T = \{1\}$ and $I_B = \{2, 3\}$. The target region, T, is surrounded by h_1, the common flow region lies between h_2 and h_3 above the exit boundary, h_e.

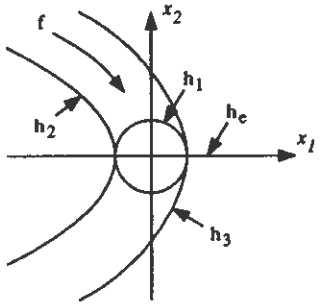

Figure 7: Target Region and Invariants

We now present two propositions which can be used to determine the suitability of a set of hypersurfaces to achieve our goal of identifying a common flow region. In different situations, one of the propositions may be easier to apply than the other. The following propositions give sufficient conditions for the hypersurfaces bounding B and T to ensure that all state trajectories in B will reach the target region.

Proposition 1 *Given the following:*

1. *A flow generated by a smooth vector field, f_k*
2. *A target region, $T \subset \mathbf{X}$*
3. *A set of smooth hypersurfaces, $h_i, i \in I_B \subset 2^I$*
4. *A smooth hypersurface (exit boundary), h_e*

such that $B = \{\xi \in \mathbf{X} : h_i(\xi) < 0, h_e(\xi) > 0, \forall i \in I_B\} \neq \emptyset$. *For all* $\xi \in B$ *there is a finite time,* t, *such that* $F_k(\xi, t) \in T$, *if the following conditions are satisfied:*

1. $\nabla_\xi h_i(\xi) \cdot f(\xi) = 0, \forall i \in I_B$
2. $\exists \epsilon > 0, \nabla_\xi h_e(\xi) \cdot f(\xi) < -\epsilon, \forall \xi \in B$
3. $B \cap \mathcal{N}(h_e) \subset T$

Proof: The proof of this proposition is straightforward. The first condition of the proposition, which can be rewritten as

$$\frac{dh_i(\mathbf{x}(t))}{dt} = 0, \tag{26}$$

precludes the state trajectory crossing any hypersurface indexed by the set I_B, thus ensuring no trajectory in B will leave B except through the remaining boundary. The second condition, which can be rewritten as

$$\frac{dh_e(\mathbf{x}(t))}{dt} < -\epsilon, \tag{27}$$

ensures that within a finite time,

$$t < \frac{h_e(\xi)}{\epsilon}, \tag{28}$$

the trajectory at $\xi \in B$ will cross the exit boundary. The final condition guarantees that any trajectory leaving B through the exit boundary will be in the target region when it does so. Together these conditions are sufficient to guarantee that any state in B will enter the target region in finite time. □

The second proposition uses a slightly different way of specifying a common flow region. In addition to the invariant manifolds and the exit boundary, there is also a cap boundary. The cap boundary is used to obtain a common flow region which is bounded. So for this case

$$B = \{\xi \in \mathbf{X} : h_i(\xi) < 0, h_e(\xi) > 0, h_c(\xi) < 0, \forall i \in I_B\}. \tag{29}$$

Proposition 2 *Given the following:*

1. *A flow generated by a smooth vector field,* f_k
2. *A target region,* $T \subset \mathbf{X}$
3. *A set of smooth hypersurfaces,* $h_i, i \in I_B \subset 2^I$
4. *A smooth hypersurface (exit boundary),* h_e
5. *A smooth hypersurface (cap boundary),* h_c

such that $B = \{\xi \in \mathbf{X} : h_i(\xi) < 0, h_e(\xi) > 0, h_c(\xi) < 0, \forall i \in I_B\} \neq \emptyset$ *and* $\overline{B}$ *(closure of* B*) is compact. For all* $\xi \in B$ *there is a finite time,* t, *such that* $F_k(\xi, t) \in T$, *if the following conditions are satisfied:*

1. $\nabla_\xi h_i(\xi) \cdot f(\xi) = 0, \forall i \in I_B$
2. $\nabla_\xi h_c(\xi) \cdot f(\xi) < 0, \forall \xi \in B \cap \mathcal{N}(h_c)$
3. $B \cap \mathcal{N}(h_e) \subset T$

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

4. There are no limit sets in $\overline{B}$

Proof: As in Proposition 1, the first condition precludes the state trajectory crossing any hypersurface indexed by the set I_B, thus ensuring no trajectory in B will leave B except through one of the remaining boundaries. The second condition, which can be rewritten as

$$\frac{dh_c(\mathbf{x}(t))}{dt} < 0, \tag{30}$$

ensures that no trajectory can leave B through the cap boundary. Thus, the exit boundary provides the only available egress from B. The third condition guarantees that any trajectory leaving B through the exit boundary will be in the target region when it does so. The final condition permits the application of a previously known result, stating that any state within a compact set without limit sets will leave that compact set in finite time. □

Consider the hypersurfaces defined by $\{h_i : i \in I_B\}$. These hypersurfaces must first be invariant under the vector field of the given control policy, f. This can be achieved by choosing them to be integral manifolds of an $n-1$ dimensional distribution which is invariant under f. An $n-1$ dimensional distribution, $\Delta(\mathbf{x})$, is invariant under f if it satisfies

$$[f(\mathbf{x}), \Delta(\mathbf{x})] \subset \Delta(\mathbf{x}), \tag{31}$$

where the $[f(\mathbf{x}), \Delta(\mathbf{x})]$ indicates the Lie bracket. Of the invariant distributions, those that have integral manifolds as we require, are exactly those which are involutive (according to Frobenius). This means

$$\delta_1(\mathbf{x}), \delta_2(\mathbf{x}) \in \Delta(\mathbf{x}) \Rightarrow [\delta_1(\mathbf{x}), \delta_2(\mathbf{x})] \in \Delta(\mathbf{x}). \tag{32}$$

Therefore by identifying the involutive distributions which are invariant under the vector field, f, we have indentified a set of candidate hypersurfaces. For details of these relationships between vector fields and invariant distributions, see [Isidori, *Nonlinear Control Systems*, Springer-Verlag, 1989].

Since an $n-1$ dimensional involutive distribution can be defined as the span of $n-1$ vector fields, over each of which it will then be invariant, and the control policy only gives one vector field, f, there will be more than one family of hypersurfaces which are all invariant under f. The set of all invariant hypersurfaces can be found in terms of $n-1$ functionally independent mappings which form the basis for the desired set of functionals, $\{h_i : i \in I_B\}$. This basis is obtained by solving the characteristic equation

$$\frac{dx_1}{f_1(\mathbf{x})} = \frac{dx_2}{f_2(\mathbf{x})} = \cdots = \frac{dx_n}{f_n(\mathbf{x})} \tag{33}$$

where $f_i(\mathbf{x})$ is the ith element of $f(\mathbf{x})$.

3.6.2 Controller Design

Once the interface has been designed, the design of the controller involves two steps. The first step is to construct one subautomaton for each control goal. This is the step which is already determined by the interface design. The second step is the connection of these subautomata to create a single DES controller. This step will depend upon the order in which the simpler control goals are to be achieved. For example, if a chemical process is to produce a sequence of different products, then each subautomaton in the controller would be designed to produce one of the products, and these subautomata would be connected to produce the products in the desired sequence.

The hypersurfaces in the generator divide the state space of the plant into a number of *cells*. Two states are in the same cell exactly when they are both on the same side (positive or negative) with respect to each hypersurface. States which lie on a hypersurface are not in any cell.

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

The first step in creating the controller is the contruction of the subautomata, one for each individual control goal. Each subautomaton is constructed in the following way.

i. Create a controller state to represent each cell.

ii. Place transitions between states which represent adjacent cells.

iii. Label each transition with the plant symbol which is generated by the hypersurface separating the associated cells.

We now have a subautomaton which can follow the progress of the plant state as it moves from cell to cell. Next the controller output function must be designed for each subautomaton.

The controller symbol output by a given controller state depends on which common flow region contains the associated cell. Each common flow region was constructed using a specific control policy, and the control symbol which initiates that control policy should be output by controller states representing cells contained in that common flow region. However, in general, common flow regions will overlap, meaning a given cell can lie in more than one common flow region. In such cases treat the cell as lying within the common flow region which is closest to the target region. Distance, in this case, is the number additional control policies which must be used to reach the target region. If common flow regions are both the same distance, then the choice is arbitrary, though the common flow region which is favored in one case must then be favored in all such cases. States which represent cells not contained in any common flow region or target region will never be visited and can thus be deleted.

Once the individual subautomata have been constructed they must be connected to form a single controller. This can be accomplished by following these steps for each subautomaton.

i. Remove the state(s) which represent cells in the target region as well all transitions emanating from such states.

ii. Connect the dangling transitions to states in the subautomaton which achieves the next desired control goal. The connections will be to the states which represent the same cells as the states which were removed.

In this way, as soon as one control goal is achieved, the system will begin working on the next one. The actual order in which each control goals are pursued is up to the designer.

3.7 Example - Unmanned Underwater Vehicle

Unmanned underwater vehicles (UUV) have a wide variety of practical usages in missions where it is dangerous or impossible to send a manned underwater vehicle. Exploration, search and rescue, salvage, mine disposal, and demolition are some examples of the uses of UUV's.

This example used a simplified model of a six-degree-of-freedom UUV depicted in Figure 8. The three types of linear displacement are *surge, sway,* and *heave*, which represent translation in the x, y, and z, directions respectively. The three types of angular displacement are *roll, pitch,* and *yaw*. The model employed here has six states which are the time derivatives of the three linear displacements and the magnitudes of the three angular displacements. By expanding to a nine state model, the magnitudes of the linear displacements, could also be included. They are omitted here to simplify the control problem and because they do not affect the dynamics of the UUV.

The UUV model has three inputs which control the rudder, stern plane, and screw. The following table summarizes the variables of the model.

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

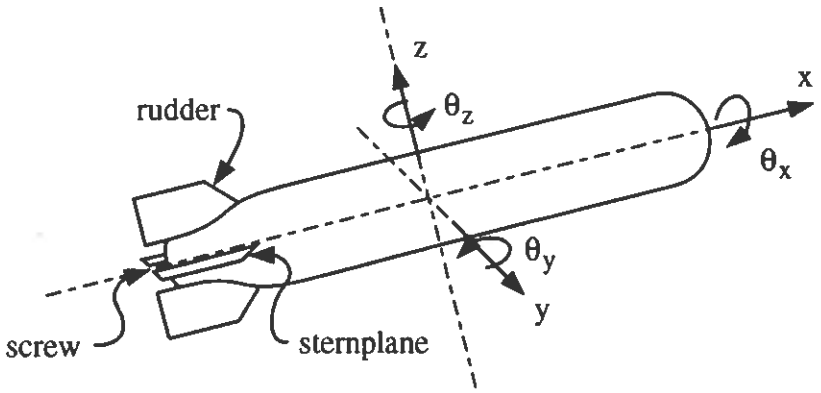

Figure 8: Unmanned Underwater Vehicle

x	surge rate (forward speed)
y	sway rate (lateral speed)
z	heave rate (vertical speed)
θ_x	roll angle in radians
θ_y	pitch angle in radians
θ_z	yaw angle in radians
u_x	screw
u_y	stern plane angle
u_z	rudder angle

The simplified model is as follows.

$$\begin{aligned} \dot{x} &= -x + u_x \\ \dot{y} &= -y + 0.01 x u_y \\ \dot{z} &= -z + 0.01 x u_z \\ \dot{\theta}_y &= 0.15 x u_y \\ \dot{\theta}_z &= 0.15 x u_z \end{aligned} \tag{34}$$

Notice that the roll angle is not included in the model. It is assumed that the center of mass of the UUV is sufficiently far beneath the center of bouyancy so that the roll angle is always zero.

To control the UUV, the actuator can implement ten different control policies. The policies allow various combinations of two screw speeds (on and off), three stern plane positions (up, level, down), and three rudder positions (left, right, straight). More will be said about these policies when the example is revisited.

$$\mathrm{r}(t) = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \tag{35}$$

where $u_x = 1, u_y \in \{-1, 0, 1\}, u_z \in \{-1, 0, 1\}$, or $u_x = u_y = u_z = 0$. This unmanned underwater vehicle example is sufficiently complex, with five states, three inputs, and non-linear dynamics, that the design methodology of this chapter cannot be carried out without some form of systematic, automated procedure.

The above hybrid control design procedure was used to design a controller for the UUV and then the design is evaluated through simulation. The state space of the UUV plant is quantized and bounded as follows.

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

$$\begin{array}{llllll} x_{1,\min} = 0 & x_{1,\max} = 1 & \Delta x_1 = 0.1 \\ x_{2,\min} = -0.5 & x_{2,\max} = 0.5 & \Delta x_2 = 0.2 \\ x_{3,\min} = -0.5 & x_{3,\max} = 0.5 & \Delta x_3 = 0.2 \\ x_{4,\min} = -\pi/2 & x_{4,\max} = \pi/2 & \Delta x_4 = \pi/20 \\ x_{5,\min} = -\pi/2 & x_{5,\max} = \pi/2 & \Delta x_5 = \pi/20 \end{array}$$

The controller has ten control policies to choose from. These policies are obtained by combining two propeller speeds $u_x \in \{0, 1\}$, three stern plane angles $u_y \in \{-10, 0, 10\}$, and three rudder angles $u_z \in \{-10, 0, 10\}$. Of the control policies with $r_1 = 0$, only the one with $r_2 = r_3 = 0$ is kept, reducing the total number of control policies from eighteen to ten.

The target region consists of the following interval.

$$\begin{bmatrix} 0.1 \\ -0.1 \\ -0.1 \\ -\pi/40 \\ -\pi/40 \end{bmatrix} < \mathbf{x} < \begin{bmatrix} 0.2 \\ 0.1 \\ 0.1 \\ \pi/40 \\ \pi/40 \end{bmatrix} \tag{36}$$

The results of two simulations are presented here. The two trials have the following initial conditions.

$$\mathbf{x} = \begin{bmatrix} \text{surge} \\ \text{sway} \\ \text{heave} \\ \text{pitch} \\ \text{yaw} \end{bmatrix}, \mathbf{x}_1 = \begin{bmatrix} 0.8 \\ 0 \\ 0 \\ -1.17 \\ 1.00 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 0.2 \\ 0.4 \\ 0 \\ 0.67 \\ -1.50 \end{bmatrix} \tag{37}$$

Results of the first trial are shown in Figure 9 and results of the second in Figure 10. Each figure consists of four graphs. The graph in the upper left shows the surge, sway, and heave over time. The graph in the upper right shows the pitch and yaw over time. The trajectories on these two graphs can be distinguished by noting the initial conditions. The two lower graphs show the control signal. On the lower left is the propeller speed which is either 0 or 1. The segments which appear to overlap reveal the presence of chattering (due to quantization). The final graph shows the stern plane angle and the rudder angle. In Figure 9, the stern plane switches between 0 and 10 and the rudder angle switches between -10 and 0. In Figure 10 these are reversed. As can be seen, the basic control strategy which developed is simply to accelerate and turn until the pitch and yaw are within the bounds of the target region, and then coast until the forward speed is also in the target.

4 An Alternative Approach to Hybrid Control and Supervision

The term 'hybrid systems' is not well defined, but is generally understood as a class of dynamical systems in which the state space includes discrete and continuous components. In the prior discussion of this chapter, hybrid systems were the consequence of quantizing the state and input spaces of an otherwise continuous process model. The quantization may be a result of physical considerations, such as the on/off quantization of measurements and control in conventional home furnace controlled by a simple thermostat. In other cases, the quantization may be imposed by the analyst for reasons of abstracting the process behavior.

In this section, we further discuss notions of supervision for DES and hybrid systems, and survey existing approaches to these problems. To illustrate some of the issues in the supervision of hybrid systems, we consider a class of linear *discrete-time* models where some variables are discrete. A simple, one-event-ahead (OEA) supervisor is developed and applied to a fuel/air combustor example. Representative types of decomposition results are list.

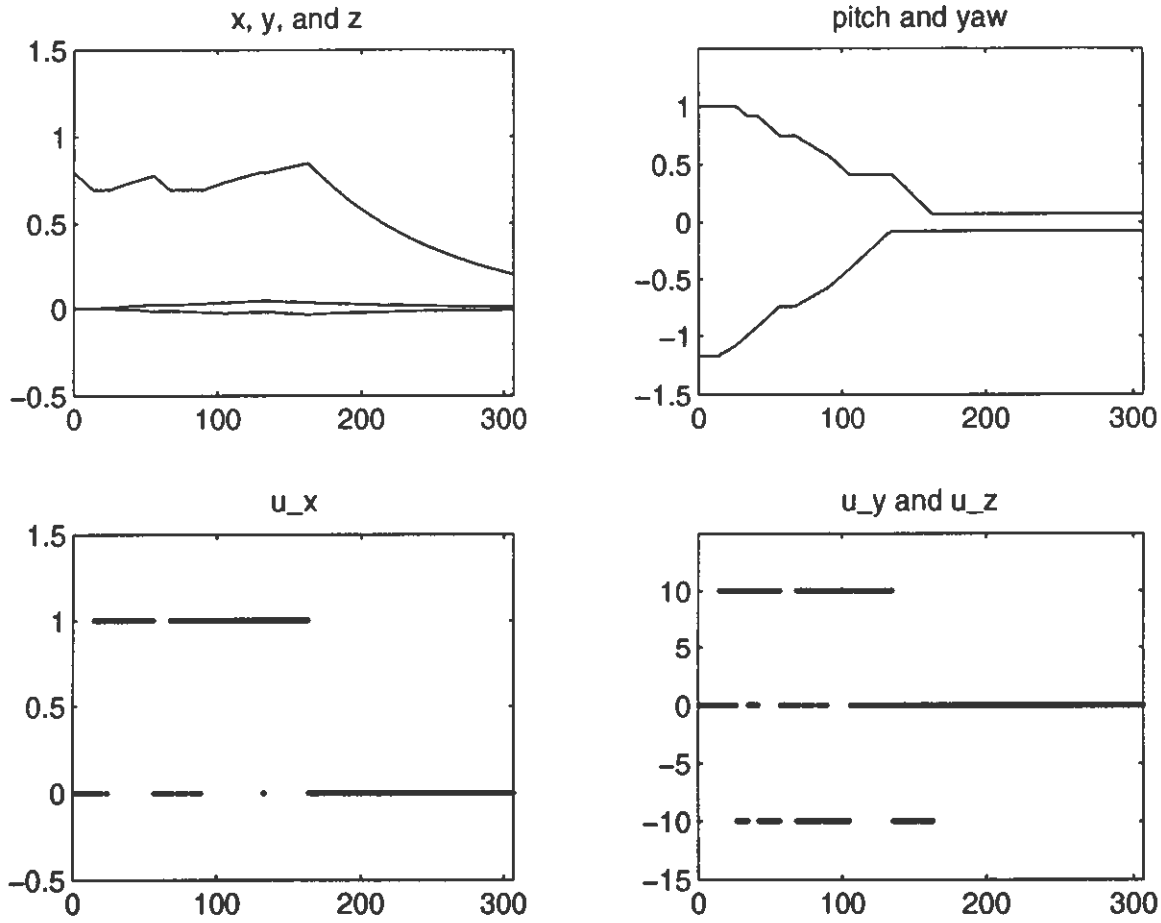

Figure 9: UUV Simulation #1

4.1 The notion of a hybrid supervisor

Supervision is a weak form of feedback control that has undergone extensive development in the context of DES systems (see, for example, [RW89, HK90, BKS93, SW93, CLL92, HALL94, LW94] and references therein). A supervisor directly or indirectly monitors the system state, then uses that information to enables or disables possible control actions. The final choice of the control action is determined by some other agent, the purpose of the supervisor is essentially to prohibit 'bad' choices from being made. Supervisors, therefore, are not usually designed to achieve particular closed-loop setpoint or disturbance rejection properties, but rather to permit maximum permissiveness in process operation without violating certain constraints. The jargon for this field includes suggestive terminology like 'maximally permissive supervisors'.

For DES systems modeled as finite state automata, Petri nets, or vector addition systems, the computation of a DES supervisor is of polynomial complexity in the number of system states. Note, however, that the number of states is typically exponential in the number of interacting elements that make up a processing system. Consequently, the direct synthesis of DES supervisors is computationally difficult for larger process systems, though promising special computational procedures have been developed and applied to realistic process models [HK90, SW93, BHG+93, dSCSC94].

Look-ahead supervision has been proposed for systems with infinite or large state spaces, or where specifications are not completely known *a priori* [CLL92, HALL94]. The look-ahead notion is similar to the ideas of 'receding horizon' or 'predictive' control that are part of the standard process control lexicon.

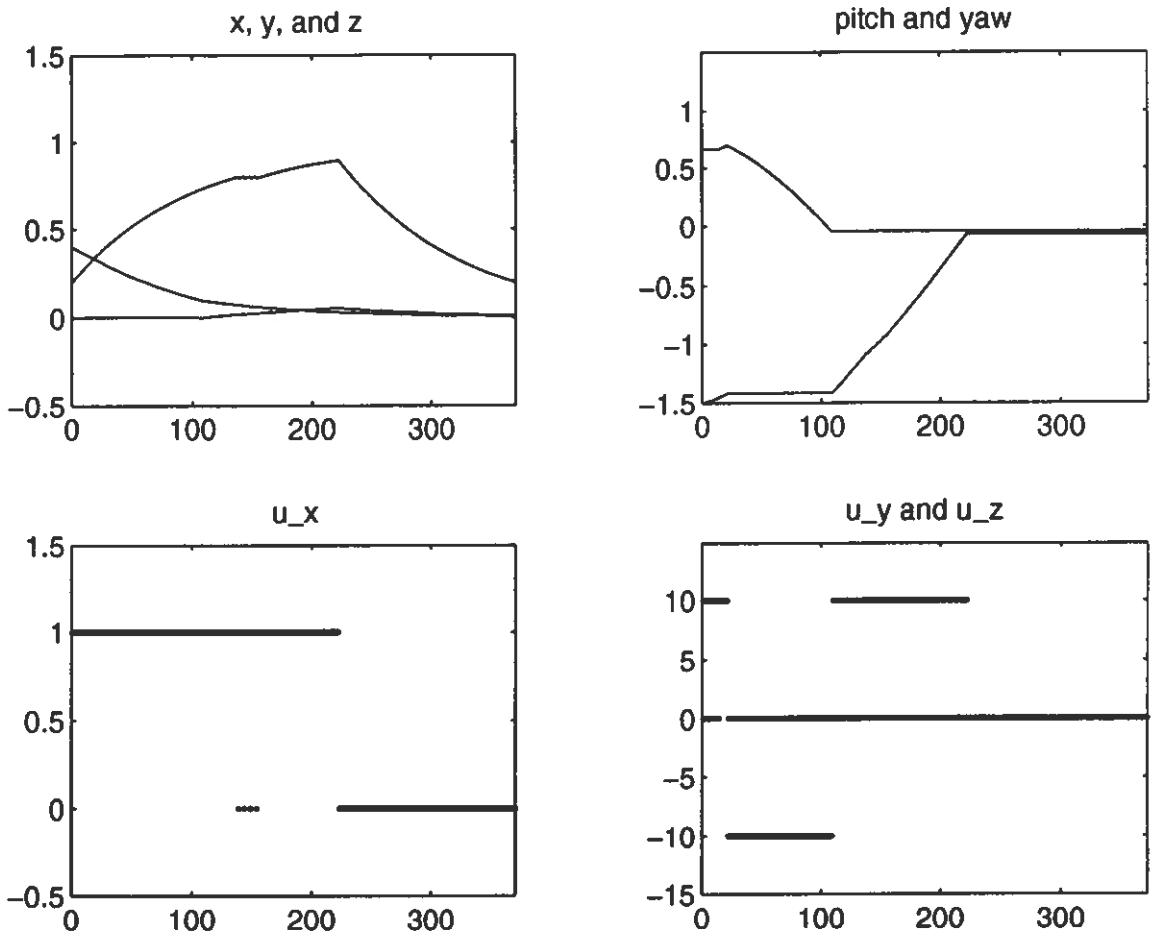

Figure 10: UUV Simulation #2

Look-ahead supervision is basically an N event ahead projection of possible system behavior. The search is conducted under 'conservative' and 'optimistic' attitudes regarding unexpanded states at the computational horizon. The worst case complexity is quadratic in the number of expanded states. A number of desirable closed-loop properties are known. Overall, look-ahead supervision shows promise for intelligent process control.

A third approach to supervisor synthesis for Petri net models has been independently proposed by Guia, *et al.* [GDS92, GDS93], and Yamalidou, *et al* [YMLA94]. The special value of the method is that it does not expand the state space of the Petri net, and involves only simple linear algebra computations. However, its application to DES systems has only been developed for Petri nets subject to convex constraints (termed generalized mutual exclusion constraints). If some of the transitions are not controllable, then the class of systems is further restricted to *safe* and *conservative* Petri nets. Unfortunately, the more general *forbidden marking* problem is outside the scope of this computationally attractive technique.

By contrast, the supervision, control, and verification of hybrid systems is in a much earlier stage of theoretical and computational development. Oversimplifying, the available approaches can be broken into three categories that are mainly distinguished by type of models used to represent the hybrid system in question:

- **Simulation/Optimization.**

The familiar commercial modeling packages, such as Simulink and MatrixX, include elements that can be used simulate the behavior of hybrid systems. Newer tools include gPROMS [Pan95], dstools [ABM93], and other packages. Supporting analysis is typically based on existing theories for variable structure systems (see, e.g., [Guc95]) and quantized systems (e.g., [Del90]).

- **Logic/Formal Systems.**

  There are a number of distinct approaches to a logic for hybrid systems, including TLA$^+$ [Lam93], Duration calculus [CRH93], and Hybrid Automata [ACHH93, TE94]. SIGNAL is an ambitious attempt to couple a workable simulation environment for hybrid systems with a strong logical foundation [BG90].

- **DES encoding.**

  A number of results have presented wherein hybrid systems are essentially 'encoded' as DES systems and DES methods applied. For examples of this approach, see [ASL93, PD94, RO94].

There would appear to be significant but unexploited overlap among the different approaches cited above. For example, it would seem the special class of Linear Hybrid Automata studied effectively by Tittus, *et al.* [TE94] could also be a very efficient basis for the simulation/optimization approach based on gPROMS [Pan95].

A broad range of wide range of potential applications for hybrid systems exists in the process industries. These include batch processing systems, safety interlock systems, pipe and valve network operations, and intelligent control. However, it is not yet clear which of these methods will lead to practical tools for control design and analysis.

4.2 A class of linear hybrid models with exclusion constraints

In this section, we consider in more depth a simple class of linear hybrid models in discrete time. This class generalizes standard condition/event Petri nets to include continuous and discrete state variables. As a subset, it includes the continuous Petri net formulation suggested by David and Alla [DA94]. This model can be exploited very effectively to develop simple hybrid supervisors, and yield strong analytical results.

For modeling purposes, the system state prior to the kth event is represented by an n element vector $x(k)$. The individual elements are defined as either real, integer, or 0-1 quantities. The state is an element of a state space $\mathcal{S}$ constructed as a direct sum

$$\mathcal{S} \equiv \mathcal{R}^{n_r} \oplus \mathcal{Z}^{n_i} \oplus \mathcal{B}^{n-n_r-n_i}$$

where $\mathcal{R}$ is the set of reals, $\mathcal{Z}$ is the set of integers, and $\mathcal{B}$ is the boolean set $\{0, 1\}$.

The state of a hybrid system model changes in response to events. An m element vector $q(k)$ represents the kth event. The elements of $q(k)$ represent components of an event, which are individually represented as either real or integer, or 0-1 variables. The vector $q(k)$ is an element of an event space $\mathcal{E}$ constructed as a direct sum

$$\mathcal{E} \equiv \mathcal{R}^{m_r} \oplus \mathcal{Z}^{m_i} \oplus \mathcal{B}^{m-m_r-m_i}.$$

The evolution of the state of a hybrid system can be modeled in many different ways, depending on the application and the purpose for building the model. In the present case, we restrict our attention to models in which the next state is a linear function of the current state and event, subject to a system of linear constraints. We will call this the Linear Hybrid Model and refer to it with an acronym LHM.

Definition 1:[Linear Hybrid Model (LHM)] A linear hybrid model is given in the form

$$x(k+1) = Ax(k) + Bq(k) \tag{38}$$

$$Px(k) + Qq(k) \leq R \tag{39}$$

where $x(k) \in \mathcal{S}$ is the state vector, and $q(k) \in \mathcal{E}$ is the event vector. A, B, P, Q, and R are conformable matrix or vector valued parameters.

Not all choices of matrix parameters A, B are compatible with a given state space $\mathcal{S}$ and event space $\mathcal{E}$. For example, the system

$$x(k+1) = \pi x(k)$$

is inconsistent with a specification $x(k) \in \mathcal{Z}$. This situation motivates the following definition.

Definition 2:[Well Formed LHM] A linear hybrid model is well formed if $Ax(k)+Bq(k) \in S$ for all $x(k) \in \mathcal{S}$ and $q(k) \in \mathcal{E}$.

In the sequel, we will assume that we are discussing well-formed LHM's.

4.3 Examples of linear hybrid models

4.3.1 An LHM with complex dynamics

Consider the hybrid model

$$x(k+1) = 2x(k) - q(k) \tag{40}$$
$$-2x(k) + q(k) \leq 0 \tag{41}$$
$$2x(k) - q(k) \leq 1 \tag{42}$$

where $x(k) \in \Re$ is a real number, and $q(k) \in \{0, 1\}$ is a 0-1 variable. For initial conditions in the range $0 < x(0) < 1$, the dynamics can be rewritten as a relation describing the next state.

$$x(k+1) = \begin{cases} 2x(k) & 0 < x(k) \leq \frac{1}{2} \\ 2x(k) - 1 & \frac{1}{2} \leq x(k) < 1 \end{cases} \tag{43}$$

This relation maps the intervals $[0, \frac{1}{2})$ and $(\frac{1}{2}, 1]$ onto $[0, 1)$. The next state is not uniquely defined for the point $\frac{1}{2}$. However, initial conditions can be defined on the interval $[0, 1]$ which yield trajectories that never encounter this point.

Initial conditions in the range $0 < x(0) < 1$ can be represented as

$$x(0) = \sum_{j=1}^{\infty} \alpha_j 2^{-j}$$

where the coefficients α_j are 0 or 1. At iteration k, the value of $x(k)$ will lie in the interval $0 < x(k) \leq \frac{1}{2}$ if $\alpha_{k+1} = 0$, or will lie in the interval $\frac{1}{2} \leq x(k) < 1$ if $\alpha_{k+1} = 1$. Typically $x(k)$ will bounce back and forth around the value of $\frac{1}{2}$ following the pattern given by the binary digits $\alpha_1, \alpha_2, \ldots$.

In fact, this dynamical system is directly related to a classical algorithm for computing a binary representation of a real number, sometimes referred to as the 'Bernoulli Shift' [Sch88]. For this case, there are three possible situations.

a) If the initial condition is such that the sequence $\alpha_j = 0$ for $j > j_0$, then $x(j_0 - 1) = \frac{1}{2}$, and the next state $x(j_0)$ could be either 0 or 1, but is not uniquely determined.

b) If the initial condition yields an infinite repeating sequence (e.g., $x(0) = \frac{1}{3}$), then $x(k)$ attains a limit cycle.

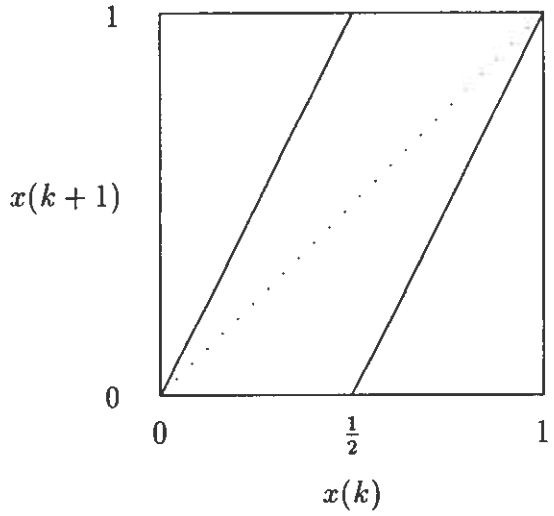

Figure 11: Next state map for the 'Bernoulli Shift'.

c) If the initial condition yields a infinite nonrepeating sequence (e.g., $x(0) = e = 2.71828\ldots$), then $x(k)$ attains an aperiodic trajectory.

This example demonstrates that even very simple LHM's can exhibit complex dynamic behavior as a consequence of the interplay between unstable linear dynamics, and linear and integer constraints.

4.3.2 An LHM exhibiting chaos

Consider an LHM with state update equation

$$x(k+1) = 2x(k) + \begin{bmatrix} 2 & -1 \end{bmatrix} \begin{bmatrix} q_1(k) \\ q_2(k) \end{bmatrix}$$

where $x(k)$ and $q_2(k)$ are real variables, and $q_1(k)$ is a 0-1 variable, subject to constraints

$$\begin{bmatrix} -2 \\ 2 \\ -4 \\ 4 \\ 0 \\ 0 \end{bmatrix} x(k) + \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 2 & -1 \\ -4 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} q_1(k) \\ q_2(k) \end{bmatrix} \leq \begin{bmatrix} 0 \\ 1 \\ 0 \\ 2 \\ 0 \\ 0 \end{bmatrix}$$

Consider initial conditions $x(0)$ in the interval $[0, 1]$. The 0-1 condition and the first pair of constraints on $q_1(k)$ causes to be an indicator function showing which half of the interval contains $x(k)$, i.e,

$$q_1(k) = \begin{cases} 0 & 0 \leq x(k) \leq \frac{1}{2} \\ 1 & \frac{1}{2} \leq x(k) \leq 1 \end{cases}$$

Coupling this result with the four remaining constraints yields an solution for $q_2(k)$,

$$q_1(k) = \begin{cases} 0 & 0 \leq x(k) \leq \frac{1}{2} \\ 4x & \frac{1}{2} \leq x(k) \leq 1 \end{cases}$$

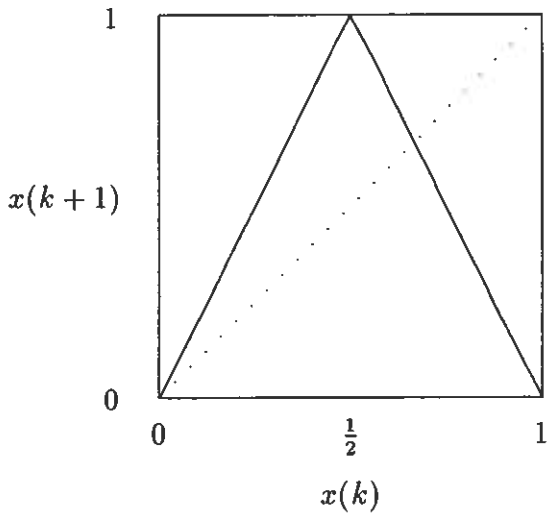

Figure 12: Next state map for the 'tent' function.

Eliminating $q_1(k)$ and $q_2(k)$ from the state update equation yields the net result

$$x(k+1) = \begin{cases} 2x & 0 \leq x(k) \leq \frac{1}{2} \\ 2(1-x) & \frac{1}{2} \leq x(k) \leq 1 \end{cases}$$

This is the famous 'tent' map first analyzed by Ulam and von Nuemann (a recent discussion is available in [Sch88]) and illustrated in Fig. 12. Any initial conditions on the unit interval (other than the fixed points 0, 2/3, and 1, and the countably infinite set of points that are precursors to them) yield a chaotic sequence of iterates that uniformly covers the interval.

4.3.3 Condition-Event Petri nets

The LHM specializes to a condition-event Petri net. Consider the case where the elements of $x(k)$ and $q(k)$ are 0-1 variables, and the model parameters are integer arrays of the form

$$A = I_n \tag{44}$$
$$B = D^+ - D^- \tag{45}$$
$$P = \begin{bmatrix} -I_n \\ 0_{1\times n} \end{bmatrix} \tag{46}$$
$$Q = \begin{bmatrix} D^- \\ 1_{1\times n} \end{bmatrix} \tag{47}$$
$$R = \begin{bmatrix} 0_{n\times 1} \\ 1 \end{bmatrix} \tag{48}$$

where $D^{\pm} \geq 0$, and the notations $0_{r\times s}$ and $1_{r\times s}$ denote $r \times s$ matrices of zeros and ones, respectively. These definitions yield a model

$$x(k+1) = x(k) + (D^+ - D^-)q(k) \tag{49}$$
$$x(k) \geq D^- q(k) \tag{50}$$
$$1 \geq \sum_{i=1}^{m} q_i(k) \tag{51}$$

A non-negative initial condition $x(0) \geq 0$ will result in $x(k) \geq 0$ for all $k > 0$. The condition $\sum_{i=1}^{m} q_i(k) \leq 1$ is introduced here to yield a model where one event transition occurs per increment of the counter k. This is a standard representation of a condition-event Petri net where the matrices D^- and D^+ describe the net topology [CR90].

4.3.4 Vector Discrete-Event Systems

Li and Wonham introduced Vector Discrete-Event Systems (VDES) as an analytically tractable device for modeling discrete-event systems [LW93, LW94]. The state space of a VDES is a set of n dimensional integer vectors, i.e., $\mathcal{S} \equiv \mathcal{Z}^n$. A set of possible events is denoted by Σ which is further partitioned into controllable and uncontrollable event sets. A state transition is modeled as $\delta : \Sigma \times \mathcal{S} \rightarrow \mathcal{S}$ denoted by

$$\delta(\alpha, x) = x + E_\alpha$$

where $E_\alpha \in \mathcal{S}$ is the displacement vector for event $\alpha \in \Sigma$. In the formulation of Li and Wonham, there is state transition function written for each event. A state transition is defined, denoted by $\delta(\alpha, x)!$, if $x \geq F_\alpha$, where F_α is the called the occurrence-condition vector for event α.

A VDES model can be directly translated to an instance of an LHM. First, we consider α as index into the m elements i of the event set Σ. Then $q_\alpha(k) = 1$ if event $\alpha \in \Sigma$ is the kth event to take occur, otherwise $q_\alpha(k) = 0$. In the obvious way, a VDES translated to an LHM given by

$$x(k+1) = x(k) + \begin{bmatrix} E_1 & \cdots & E_m \end{bmatrix} \begin{bmatrix} q_1(k) \\ \vdots \\ q_m(k) \end{bmatrix} \tag{52}$$

$$-x(k) + \begin{bmatrix} F_1 & \cdots & F_m \end{bmatrix} \begin{bmatrix} q_1(k) \\ \vdots \\ q_m(k) \end{bmatrix} \leq 0 \tag{53}$$

An additional constraint $\sum_{i=1}^{m} q_i(k) \leq 1$ assures that only one event takes place per unit 'time'. The result is very similar to the case of a condition-event Petri net, with the main difference being that the state vector is defined over the set of integers (including non-positive integers) rather than 0-1 variables.

4.3.5 Linear discrete time systems

A specialization of LHM to conventional linear time-invariant discrete time systems is constructed very simply. Let $P = Q = R = [\,]$, and let $x(k) \in \mathcal{R}$, then

$$x(k+1) = Ax(k) + Bq(k)$$

where $q(k)$ may be interpreted as exogenous inputs.

The inequalities can be used to model systems with feedback. For example, choosing $P = \begin{bmatrix} K \\ K \end{bmatrix}$, $Q = \begin{bmatrix} I_m \\ I_m \end{bmatrix}$, and $R = 0_{2m \times n}$ is equivalent to the conventional LTI system above with the added feedback constraint

$$q(k) = -Kx(k)$$

4.3.6 Modeling a buffer tank

A simple buffer tank system demonstrates the application LHM's to the modeling of simple processes [dSCSC94]. Consider the system shown in Fig. 13. The outlet flow of the tank changes with time under the

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

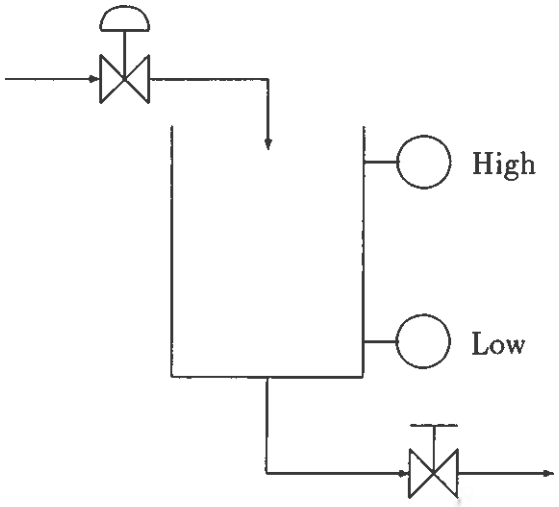

Figure 13: A simple buffer tank system.

control of an external agent. Sensors are placed near the top and bottom of the tank to detect near full and near empty conditions. The tank inlet valve may be turned on or off, and is to be controlled to regulate liquid level.

Let variable $x_1(k) \in \mathcal{R}_+$ denote the liquid height as measured from the bottom of the tank at time k. A model is then

$$x_1(k+1) = x_1(k) + \frac{\delta t}{A}(F x_2(k) - q_1(k))$$

where $q_1 \in \mathcal{R}_+$ denotes the outlet flowrate, and Fx_2 is the inlet flowrate computed as the product of a model parameter F and a state variable $x_2 \in \mathcal{B}$ that indicates whether the inlet valve is open. This model could be derived from a continuous time model by employing the zero-order hold where δt is the sample time, and A is the cross-sectional area of the tank.

The inlet valve is modeled by a simple Petri net as shown in Fig. 14. Discrete states $x_2 \in \mathcal{B}$ and $x_3 \in \mathcal{B}$ correspond to the valve open and valve closed conditions. Corresponding transitions q_2 and q_3 are controllable. When expressed in the form of an LHM, the inlet valve model is given by the relations

$$\begin{bmatrix} x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} q_2(k) \\ q_3(k) \end{bmatrix}$$
$$- \begin{bmatrix} x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q_2(k) \\ q_3(k) \end{bmatrix} \leq 0$$

The final piece of the model is a dynamical description of the high and low alarm sensors. As illustrated in Fig. 15, three discrete states x_4, x_5, x_6 are introduced to model conditions of high, normal, and low liquid level, respectively. There a four transitions which we label q_4, q_5, q_6, q_7. The state-update component of this part of the process model may be written as

$$\begin{bmatrix} x_4(k+1) \\ x_5(k+1) \\ x_6(k+1) \end{bmatrix} = \begin{bmatrix} x_4(k) \\ x_5(k) \\ x_6(k) \end{bmatrix} + \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} q_4(k) \\ q_5(k) \\ q_6(k) \\ q_7(k) \end{bmatrix}$$

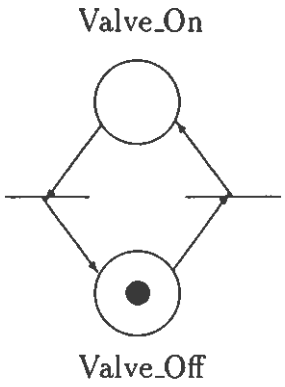

Figure 14: Model of a controllable inlet valve.

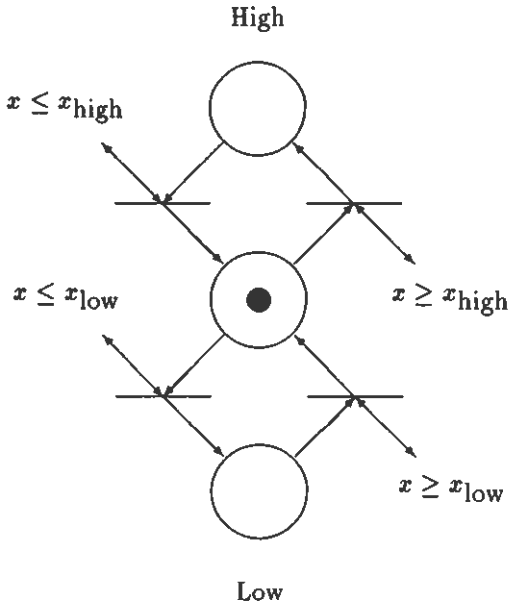

Figure 15: Model of the level sensing logic.

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

The transitions depend on the state of continuous and discrete variables. For example, consider the transition from an indication of high to normal liquid level. That transition at time k is equivalent to the condition $q_4(k) = 1$. Necessary preconditions for event $q_4(k) = 1$ are that $x_4(k) \geq 1$ (i.e., the liquid level indicator registers high), and $x_1(k) \leq x_{\text{high}}$ (i.e., the measured liquid level is below the value x_{high}). These necessary conditions may be expressed as $x_4(k) \geq q_4(k)$ and $x_{\text{high}} - x_1(k) \geq M(q_4(k) - 1)$ where M is a large, positive, model dependent parameter. Following a similar procedure for each transition yields a system of constraints:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_4(k) \\ x_5(k) \\ x_6(k) \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ M & 0 & 0 & 0 \\ 0 & M & 0 & 0 \\ 0 & 0 & M & 0 \\ 0 & 0 & 0 & M \end{bmatrix} \begin{bmatrix} q_4(k) \\ q_5(k) \\ q_6(k) \\ q_7(k) \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ M - x_1(k) + x_{\text{high}} \\ M + x_1(k) - x_{\text{high}} \\ M - x_1(k) + x_{\text{low}} \\ M + x_1(k) - x_{\text{low}} \end{bmatrix}$$

An LHM for the buffer tank is obtained by combining the three models described above for the tank, inlet valve, and level sensor components.

$$A = \begin{bmatrix} 1 & \begin{bmatrix} \frac{\delta t F}{A} & 0 & 0 & 0 & 0 \end{bmatrix} \\ 0_{5\times 1} & I_5 \end{bmatrix} \tag{54}$$

$$B = \begin{bmatrix} -\frac{\delta t}{A} & & & & & & & \\ & 1 & -1 & & & & & \\ & -1 & 1 & & & & & \\ & & & -1 & 1 & & & \\ & & & 1 & -1 & -1 & 1 & \\ & & & & & 1 & -1 & \end{bmatrix} \tag{55}$$

$$P = \begin{bmatrix} 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{56}$$

$$Q = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & M & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & M & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & M & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & M \end{bmatrix} \tag{57}$$

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

$$R = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ M + x_{\text{high}} \\ M - x_{\text{high}} \\ M + x_{\text{low}} \\ M - x_{\text{low}} \end{bmatrix} \tag{58}$$

4.4 A one-event ahead hybrid supervisor

Here we develop demonstrate the construction of a simple supervisor for an LHM subject to mutual exclusion constraints. The technique follows along the lines developed by Guia *et al.* [GDS92, GDS93] and Yamalidou *et al.* [YMLA94] for Petri nets, but incorporating certain elements of look-ahead supervision due to Lafortune and coworkers.

For application to LHM's, it is first necessary to resolve the difference between time and events. In the present discussion, the counter k refers to time measured as ticks on a master clock. The control horizon, N, is chosen to look ahead a fixed amount of time horizon. Over this time horizon, 0 or more events might occur as indicated by the discrete elements of the input vector $q(k)$ on the interval from k to $k+N-1$. The one-event ahead supervisor is developed assuming that only one controllable event occurs on this interval at time k. For the limiting case where the LHM represents a Petri net, we show this is equivalent to the situation presented by Guia and Yamalidou. In the other limit of continuous variable linear discrete-time systems, this is roughly analogous to one-step ahead model predictive control on a horizon N.

Consider an LHM for which the exclusion constraints depend on state alone, that is $Q = 0$,

$$x(k+1) = Ax(k) + Bq(k) \tag{59}$$

$$Px(k) \leq R \tag{60}$$

with initial conditions $x(0) = x_0$. Following Yamalidou, *et al.*, a conformable vector of slack variables $s(k)$ is introduced so that the constraints become

$$Px(k) + s(k) = R$$

The constraints $s(k) \geq 0$ are to hold at all future time steps. Considering just the next time step yields the recursion

$$\begin{bmatrix} x(k+1) \\ s(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ P(A-I) & I \end{bmatrix} \begin{bmatrix} x(k) \\ s(k) \end{bmatrix} + \begin{bmatrix} B \\ PB \end{bmatrix} q(k)$$

with initial conditions

$$\begin{bmatrix} x(0) \\ s(0) \end{bmatrix} = \begin{bmatrix} x_0 \\ Px_0 - R \end{bmatrix}$$

The extra states introduced by the recursion on the slack variables are called 'monitors', and a one-step ahead prediction requires that $q(k)$ satisfy the relation $s(k+1) \geq 0$, i.e.,

$$P(A-I)x(k) + s(k) + PBq(k) \geq 0$$

The results of Yamalidou *et al.* can be obtained for regular Petri nets (i.e., Petri nets without self-loops, c.f. [LY90]). In the case of Petri nets, $A \equiv I$, $P \equiv \begin{bmatrix} -I \\ P_2 \end{bmatrix}$, and $R \equiv \begin{bmatrix} 0 \\ R_2 \end{bmatrix}$. In this case, the slack variables

$s(k)$ represent controller memory, and the one step ahead constraint $s(k+1) \geq 0$ yields the relation

$$x(k+1) \geq 0 \tag{61}$$
$$s(k) + P_2 B q(k) \geq 0 \tag{62}$$

Under the no self-loop condition of regular Petri nets, the first inequality is the standard Petri net firing rule, and the second consists of problem specific exclusion constraints. Yamamidou, *et al.* [YMLA94], demonstrated the utility of this approach when applied to several sample problems, and Guia, *et al.* [GDS92, GDS93], proved several analytical results.

We introduce the notation $\hat{x}(k+j|k)$, $\hat{s}(k+j|k)$, and $\hat{q}(k+j|k)$ to denote projected values the state, slack, and input variables at time $k+j$ given information available up to time k. The input vector is partitioned into *controllable* and *uncontrollable* inputs, $\hat{q}^c(k+j|k)$ and $\hat{q}^u(k+j|k)$, respectively, as

$$\hat{q}(k+j|k) = \begin{bmatrix} \hat{q}^c(k+j|k) \\ \hat{q}^u(k+k|k) \end{bmatrix}$$

Likewise, B is partitioned conformably with $q(k)$ as $B = [B^c B^u]$. For purposes of this development, we make the following assumptions:

Definition 3:[One-event ahead supervision] For one-event ahead supervision, we consider one-event on the horizon k to $k+N$, and that the event occurs at k. That is,

$$\hat{q}^c(k+j|k) = 0$$

for $1 \leq j \leq N$.

Assuming that no uncontrolled events occur, then the operating constraints $\hat{s}(k+j|k) \geq 0$ yield

$$s(k) + P(A^j - I)x(k) \geq -PA^{j-1}B^c q^c(k|k) \quad j = 1, 2, \ldots, N \tag{63}$$

In contrast to the case of Petri nets, the enabling conditions on $q^c(k|k)$ depend on the current state $x(k)$ in addition to the current slack $s(k)$.

4.5 Supervision of a fuel/air combustor

The safe ignition of a fuel/air combustor requires careful monitoring of the flame state, and avoiding unsafe situations where an excess of unburnt fuel is the combustion chamber. This example has been frequently used in literature to illustrate synthesis and validation of logic based supervisors. Representative papers include [MPBC92, dSCSC94, Lam93]. Here we develop an LHM model, and demonstrate one-event ahead supervision.

The continuous state variables in the process correspond to the oxygen and fuel composition inside the combustion chamber. A qualitative model for the mixing dynamics can be written

$$x_{\text{oxy}}(k+1) = (1-a_1)x_{\text{oxy}}(k) + a_1 v_{\text{air}}(k) \tag{64}$$
$$x_{\text{fuel}}(k+1) = (1-a_2)x_{\text{fuel}}(k) + a_2 v_{\text{fuel}}(k) \tag{65}$$

where a_1 and a_2 are dilution coefficients for the mixing chamber, and $v_{\text{air}}(k)$ and $v_{\text{fuel}}(k)$ are discrete variables that denote valve state. The valve and ignitor states, in turn, are modeled by the condition/event Petri net

$$v_{\text{air}}(k+1) = q^+_{\text{air}}(k) - q^-_{\text{air}}(k) \tag{66}$$
$$v_{\text{fuel}}(k+1) = q^+_{\text{fuel}}(k) - q^-_{\text{fuel}}(k) \tag{67}$$
$$v_{\text{ign}}(k+1) = q^+_{\text{ign}}(k) - q^-_{\text{ign}}(k) \tag{68}$$

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

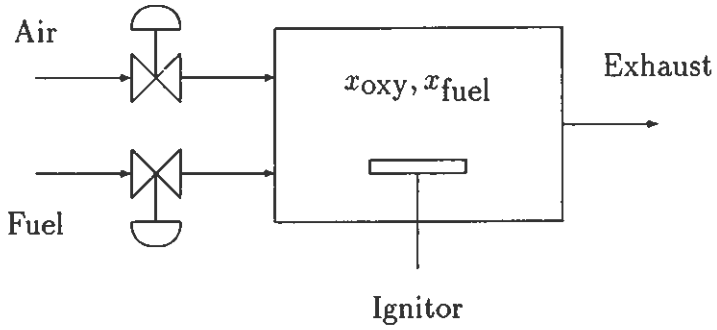

Figure 16: A Fuel/Air combustor.

where $q_j^{\pm}(k)$ are discrete controllable inputs.

There are several distinct classes of operating constraints. The first class are physical constaints which state that the valves and ignitors are either open or closed, and must be in one of those two states. These types of constraints are routine and not explicitly written here. The more significant constraints are posed by safety issues. For this illustration, we consider two constraints

$$x_{fuel} \leq 0.02x_{oxy}(k) + 0.98v_{ign} \quad (69)$$

$$x_{oxy} \geq 0.8v_{ign}(k) \quad (70)$$

The first of these constaints requires that the ignitor be on before the fuel composition reaches a flammability threshold. The second requires there to be a flow of air before the ignitor is turned on.

The one event ahead hybrid supervisor constructed above was applied to thsi example. As shown in Fig. 17, the supervisor was used to screen commands that were passed down from a higher-level operations planning module. The supervisor monitors the process state, and applies a requested command if enabled. If not enabled within a time-out window, the command is rejected and presumably an error recovery procedure is invoked.

Fig. 18 shows the result of applying the one event hybrid supervisor to vet a series of commands. For this simulation, the model parameters were set to $a_1 = 0.01$, $a_2 = 0.02$, and the supervisor look ahead horizon was set to $N = 5$. A series of commands were sent to the supervisor to turn on air, turn on ignitor, turn on fuel, and then to execute a safe shutdown. The supervisor accepted these commands, waiting until it identified a horizon for safe operation.

5 Concluding Remarks

Intelligent methods in control are becoming part of the mainstream control approaches. They are application driven for the most part and they represent our hope to meet the challenges of tomorrow.

A brief outline of some recent work in learning control and the control of DES using Petri nets is now given.

Learning Control: Learning is an important dimension or attribute of Intelligent Control [4]. Highly autonomous behavior is a very desirable characteristic of advanced control systems, so they perform well under changing conditions in the plant and the environment (even in the control goals), without external intervention. This requires the ability to adapt to changes affecting, in a significant manner, the operating region of

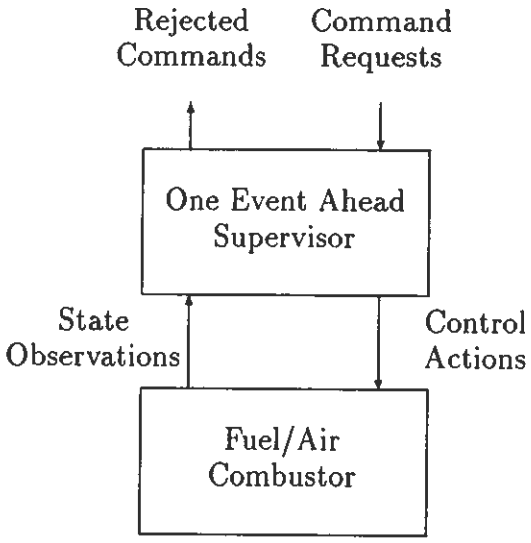

Figure 17: Hybrid Supervision of the Fuel/Air Combustor.

the system. Adaptive behavior of this type typically is not offered by conventional control systems. Additional decision making abilities should be added to meet the increased control requirements. The controller's capacity to learn from past experience is an integral part of such highly autonomous controllers. The goal of introducing learning methods in control is to broaden the region of operability of conventional control systems. Therefore the ability to learn is one of the fundamental attributes of autonomous intelligent behavior [1][4]. An introduction to learning in control can be found in [12]; see also the references therein. Contributions to learning control include our work in neural networks [5-9] and also [11-12, 20].

Discrete Event Systems and Petri Nets: Discrete event system theory is important in intelligent control, as it can be used for example to study planning the different control tasks. Discrete event systems have been studied in connection to hybrid systems; see also [13-16] for additional contributions. Recently a very promising approach to design feedback Petri net controllers for discrete event systems described by Petri nets has been developed [17]. Petri nets are very powerful and flexible graphical and mathematical modeling tools. As a graphical tools Petri nets can be used as a visual communication aid similar to flow charts, block diagrams and networks and for simulation of discrete event systems. As a mathematical tool it is possible to set up state equations that describe the behavior of the system. In the past their use in control has been somewhat limited, the main reason being the lack of appropriate methodologies to control systems described via Petri nets. Recently an approach to feedback control of systems described via Petri nets was developed, that uses the concept of place invariants of the net and it is simple and transparent. It appears that for the first time one will be able to systematically derive feedback controllers for real practical discrete event systems [17].

Acknowledgement - this work was partially supported by NSF grants MSS-9216559 and IRI91-09298.

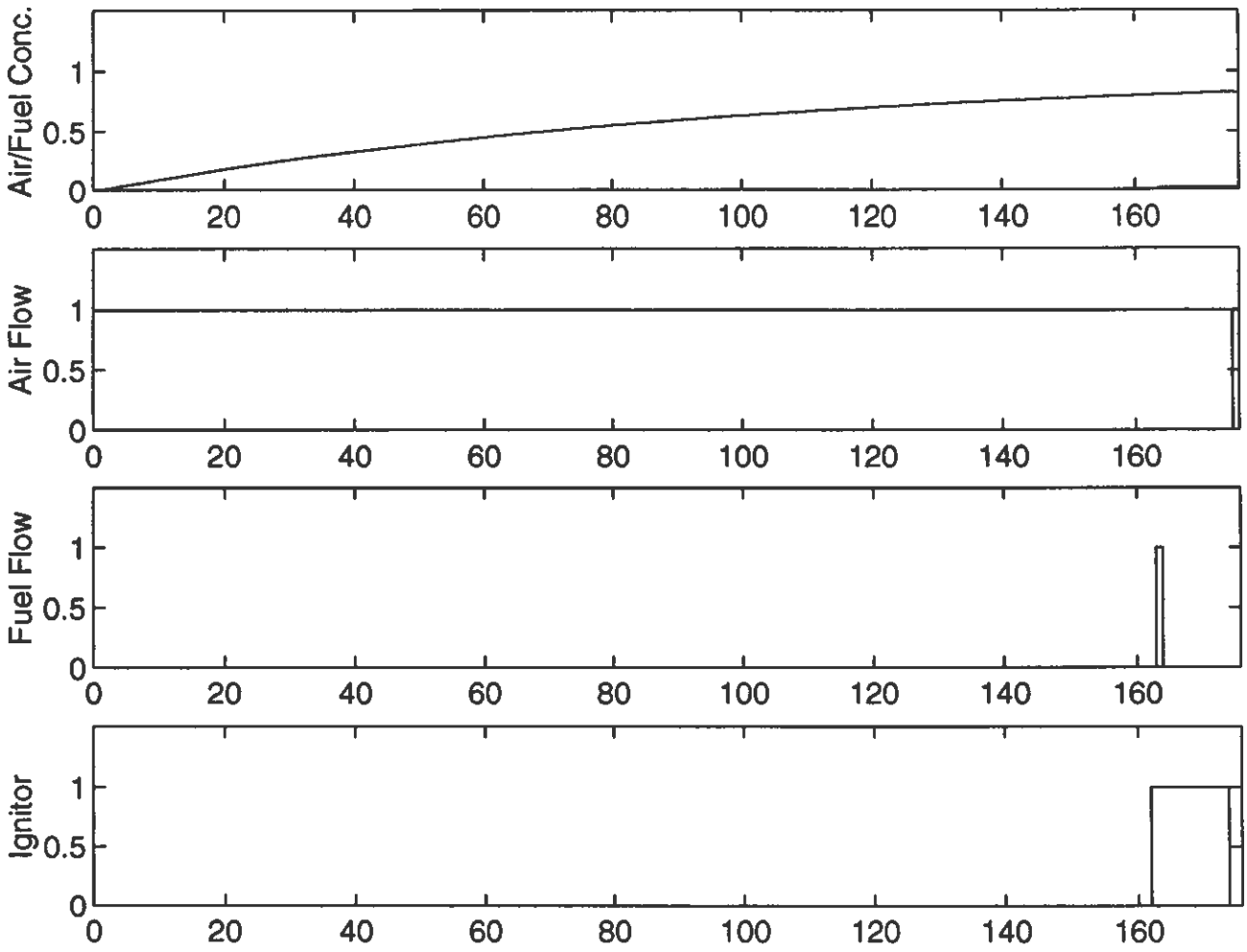

Figure 18: One event ahead hybrid supervisor applied to the Fuel/Air combustor example.

6 References

6.1 Part I

[1]. P.J.Antsaklis and K.M.Passino, Eds, An Introduction to Intelligent and Autonomous Control, 448 pages, Kluwer Academic Publishers, 1993.

[2]. P. J. Antsaklis and K. M. Passino, "Introduction to Intelligent Control Systems with High Degree of Autonomy", Introduction to Intelligent and Autonomous Control, P.J.Antsaklis and K.M.Passino, Eds., Chapter 1, pp. 1-26, Kluwer,1993.

[3]. P. J. Antsaklis, M. D. Lemmon and J. A. Stiver, "Learning to be Autonomous: Intelligent Supervisory Control", in Intelligent Control: Theory and Practice, Gupta M.M., Sinha N.K., eds., IEEE Press, Piscataway, NJ, 1995.

[4]. "Defining Intelligent Control", Report of the Task Force on Intelligent Control, P.J Antsaklis, Chair. In IEEE Control Systems Magazine, pp. 4-5 & 58-66, June 1994. Also in Proceedings of the 1994 International Symposium on Intelligent Control, pp. (i)-(xvii), Columbus, OH, August 16-18, 1994.

[5]. P. J. Antsaklis, "Neural Networks for the Intelligent Control of High Autonomy Systems", chapter 1 in Mathematical Approaches to Neural Networks, J.G. Taylor, Ed., pp 1-23, Elsevier, 1993.

[6]. I. K. Konstantopoulos and P. J. Antsaklis, "Integration of Controls and Diagnostics Using Neural Networks", Proc of the 1994 American Control Conference, pp. 1717-1721, Baltimore, MD, June 29-July 1, 1994.

[7]. M. A. Sartori and P.J. Antsaklis, "Implementations of Learning Control Systems Using Neural Networks", IEEE Control Systems, in Special Issue on 'Neural Networks in Control Systems', Vol.12, No.3, pp.49-57, April 1992.

[8]. P.J. Antsaklis, "Neural Networks in Control Systems", Guest Editor's Introduction, IEEE Control Systems Magazine, Vol.10, No.3, pp.3-5, April 1990; Special Issue on 'Neural Networks in Control Systems' of the IEEE Control Systems Magazine, Vol.10, No.3, pp.3-87, April 1990. Also Guest Editor's Introduction, IEEE Control Systems Magazine, Vol.12, No.3, pp.8-10, April 1992; Special Issue on 'Neural Networks in Control Systems' of the IEEE Control Systems Magazine, Vol.12, No.3, pp.8-57, April 1992.

[9]. J. O. Moody and P. J. Antsaklis, "The Dependence Identification Neural Network Construction Algorithm", 1994 International Conference on Neural Networks, Vol VII, pp 4799-4804, Orlando, FL, June 26-July 2, 1994. Also in IEEE Transactions on Neural Networks. To appear.

[10]. P. J. Antsaklis, "Guest Editor's Introduction: Intelligent Learning Control", IEEE Control Systems, Special Issue on Intelligent Learning Control, June 1995.

[11]. M.D. Lemmon, P. J. Antsaklis, Xiaojunun Yang and C. Lucisano, "Control System Synthesis through Inductive Learning of Boolean Concepts", IEEE Control Systems, Special Issue on Intelligent Learning Control, June 1995.

[12]. M. D. Peek and P. J. Antsaklis, "Parameter Learning for Performance Adaptation", IEEE Control Systems Magazine, pp.3-11, December 1990.

[13]. K. M. Passino and P. J. Antsaklis, "On the Optimal Control of Discrete Event Systems", Proc. of the 28th IEEE Conf. on Decision and Control, pp. 2713-2718, Tampa, FL, Dec. 13-15, 1989.

[14]. K. M. Passino and P. J. Antsaklis, "Event Rates and Aggregation in Hierarchical Discrete Event Systems", Journal of Discrete Event Dynamic Systems, Vol.1, No.3, pp. 271-288, January 1992.

[15]. K. M. Passino and P. J. Antsaklis, "A Metric Space Approach to the Specification of the Heuristic Function for the A* Algorithm", IEEE Trans. on Systems, Man, and Cybernetics, Vol 24, no 1, pp 159-166, Jan 1994.

[16]. K. M. Passino, A. N. Michel and P. J. Antsaklis, "Lyapunov Stability of a Class of Discrete Event Systems", IEEE Trans. Auto Contr, Vol.39, No.2, pp 269-279, Febr 1994; correction, p 1531, July 1994.

[17]. J. Moody, K. Yamalidou, M. D. Lemmon and P. J. Antsaklis, "Feedback Control of Petri Nets Based on Place Invariants", Proc of the 33rd IEEE Conference on Decision and Control, pp 3104-3109, Lake Buena Vista, FL, Dec 14-16, 1994. Also to appear in Automatica.

[18]. J.A. Stiver, P.J. Antsaklis and M.D. Lemmon , "A Logical DES Approach to the Design of Hybrid Control Systems", Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-94-011, Univ of Notre Dame, Nov. 1994, rev. March 1995. Also in Mathematical and Computer Modeling, Special Issue on Discrete Event Systems, 1995, to appear.

[19]. P. J. Antsaklis, J. A. Stiver and M. D. Lemmon, "Hybrid System Modeling and Autonomous Control Systems", Hybrid Systems, R L Grossman, A Nerode, A P Ravn, H Rischel Eds, pp 366-392, Lecture Notes in Computer Science, LNCS 736, Springer-Verlag, 1993.

[20]. M. D. Lemmon, J. A. Stiver and P. J. Antsaklis, "Event Identification and Intelligent Hybrid Control", Hybrid Systems, R L Grossman, A Nerode, A P Ravn, H Rischel Eds, pp 269-296, Lecture Notes in Computer Science, LNCS 736, Springer-Verlag, 1993.

[21]. M. D. Lemmon and P. J. Antsaklis, "Inductively Inferring Valid Logical Models of Continuous-State Dynamical Systems", Journal of Theoretical Computer Science, vol 137, January 1995.

[22]. P. J. Antsaklis, M. D. Lemmon and J. A. Stiver, "Modeling and Design of Hybrid Control Systems", 2nd IEEE Mediterranean Symposium on New Directions in Control and Automation, pp 440-447, Chania, Crete, Greece, June 19-22, 1994.

[23]. J. A. Stiver, P. J. Antsaklis and M. D. Lemmon, "Digital Control from a Hybrid Perspective", Proc of the 33rd IEEE Conference on Decision and Control, pp 4241-4246, Lake Buena Vista, FL, Dec 14-16, 1994.

[24]. Xiaojun Yang, P. J. Antsaklis and M.D. Lemmon, "On the Supremal Controllable Sublanguage in the Discrete Event Model of Nondeterministic Hybrid Control Systems", Technical Report of the ISIS Group, No. ISIS-94-004, Univ of Notre Dame, March 1994. Also in IEEE Transactions on Automatic Control, to appear.

[25]. J. A. Stiver and P. J. Antsaklis, "On the Controllability of Hybrid Control Systems", Proc 32nd IEEE Conference on Decision and Control, pp. 294-299, San Antonio, TX, Dec. 15-17, 1993.

[26]. J. A. Stiver and P. J. Antsaklis, "Modeling and Analysis of Hybrid Control Systems", Proc of the 31st Conference on Decision and Control, pp.3748-3751, Tucson, AZ, Dec. 16-18, 1992.

[27]. J.A. Stiver, P.J. Antsaklis and M.D. Lemmon, "Interface Design for Hybrid Control Systems", Technical Report of the ISIS Group (Interdisciplinary Studies of Intelligent Systems) ISIS-95-001, University of Notre Dame, January 1995.

P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems," Intelligent S ystems i n P rocess E ngineerin g , pp. 37-46, AIChE Symposium Series, J.F. Davis, G. Stephanopoulos and V. Venkatasubramanian, Eds., No. 312, Vol. 92, 1996.

6.2 Part II

References

[ABM93] John Guckenheimer Allen Back and Mark Myers. A dynamical simulation facility for hybrid systems. In R. Grossman, A. Nerode, A. Ravn, and H. Rischel, editors, *Hybrid Systems*, pages 253–267. Springer-Verlag, 1993.

[ACHH93] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In R. Grossman, A. Nerode, A. Ravn, and H. Rischel, editors, *Hybrid Systems*, pages 207–229. Springer-Verlag, 1993.

[ASL93] P. Antsaklis, J. Stiver, and M. Lemmon. Hybrid system modeling and autonomous control systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 366–392. Springer-Verlag, 1993.

[BG90] A. Benveniste and P. Le Guernic. Hybrid dynamical systems and the signal language. *IEEE Transactions on Automatic Control*, 35(5):535–546, May 1990.

[BHG+93] S. Balemi, G. J. Hoffmann, P. Gyugyi, H. Wong-Toi, and G. F. Franklin. Supervisory control of a rapid thermal multiprocessor. *IEEE Transactions on Automatic Control*, 38(7):1040–1059, July 1993.

[BKS93] S. Balemi, P. Kozák, and R. Smedinga, editors. *Discrete Event Systems: Modeling and Control*, volume 13 of *Progess in Systems and Control Theory*. Birkhäuser, 1993.

[CLL92] S.-L. Chung, S. Lafortune, and F. Lin. Limited lookahead policies in supervisory control of discrete event systems. *IEEE Transactions on Automatic Control*, 37(12):1921–1935, 1992.

[CR90] C. Cassandras and P. Ramadge. Toward a control theory for discrete event systems. *IEEE Control Systems Magazine*, 00:66–68, June 1990.

[CRH93] Zhou Chaochen, Anders P. Ravn, and Michael R. Hansen. An extended duration calculus for hybrid real-time systems. In R. Grossman, A. Nerode, A. Ravn, and H. Rischel, editors, *Hybrid Systems*, pages 36–59. Springer-Verlag, 1993.

[DA94] René David and Hassane Alla. Petri nets for modeling of dynamic systems – a survey. *Automatica*, 30(2):175–202, February 1994.

[Del90] David F. Delchamps. Stabilizing a linear system with quantized state feedback. *IEEE Transactions on Automatic Control*, 35(8):916–924, August 1990.

[dSCSC94] Arturo del Sagrado Corazón Sanchez Carmona. *Formal Specification and Synthesis of Sequential/Logic Controllers for Process Systems*. Ph.d. thesis, Imperial College of Science, Technology and Medicine, 1994.

[GDS92] A. Giua, F. DeCesare, and M. Silva. Generalized mutual exclusion constraints on nets with uncontrollable transitions. In *Proceedings 1992 IEEE International Conference on Systems, Man, and Cybernetics*, pages 974–979, Chicago, Illinois, October 1992.

[GDS93] A. Giua, F. DeCesare, and M. Silva. Petri net supervisors for generalized mutual exclusion constraints. In *Proceedings 12th IFAC World Congress*, pages Volume 1, 267–270, Sydney, Australia, July 1993.

[Guc95] John Guckenheimer. A robust hybrid stabilization strategy for equilibria. *IEEE Transactions on Automatic Control*, 40(2):321–326, February 1995.

[HALL94] Nejib Ben Hadj-Alouane, Stéphane Lafortune, and Feng Lin. Variable lookahead supervisory control with state information. *IEEE Transactions on Automatic Control*, 39(12):2398–2410, December 1994.

[HK90] L. E. Holloway and B. Krogh. Synthesis of feedback control logic for a class of controlled petri nets. *IEEE Transactions on Automatic Control*, 35(5):514–523, May 1990.

[Lam93] L. Lamport. Hybrid systems in TLA$^+$. In R. Grossman, A. Nerode, A. Ravn, and H. Rischel, editors, *Hybrid Systems*. Springer-Verlag, 1993.

[LW93] Yong Li and W. M. Wonham. Control of vector discrete-event systems i — the base model. *IEEE Transactions on Automatic Control*, 38(8):1214- 1227, 1993.

[LW94] Yong Li and W. M. Wonham. Control of vector discrete-event systems ii — controller synthesis. *IEEE Transactions on Automatic Control*, 39(3):512 - 531, 1994.

[LY90] Stéphane Lafortune and Hyuck Yoo. Some results on petri net languages. *IEEE Transactions on Automatic Control*, 35(4):482–485, April 1990.

[MPBC92] Il Moon, Gary J. Powers, Jerry R. Burch, and Edmund M. Clarke. Automatic verification of sequential control systems using temporal logic. *AIChE Journal*, 38(1):67–75, January 1992.

[Pan95] C. C. Pantelides. Modelling, simulation, and control of hybrid processes. In *Workshop on Analysis and Design of Event-Driven Operations in Process Systems*, London, UK, April 1995. Centre for Process Engineering, Imperial College of Science and Technology.

[PD94] Philippos Peleties and Raymond DeCarlo. Analysis of a hybrid system using symbolic dynamics and petri nets. *Automatica*, 30(9):1421–1427, 1994.

[RO94] J. Raisch and S. O'Young. A discrete-time framework for control of hybrid systems. In *Proceedings of 1994 Hong Kong International Workshop on New Directions of Control and Manufacturing*, pages 34–40, 1994.

[RW89] P. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–89, January 1989.

[Sch88] Heinz Georg Schuster. *Deterministic Chaos*. VCH Publishers, second revised edition, 1988.

[SW93] Rainer Scheuring and Hans Wehlan. Control of discrete event systems by means of boolean differential calculus. In S. Balemi, P. Kozák, and R. Smedinga, editors, *Discrete Event Systems: Modeling and Control*, volume 13 of *Progess in Systems and Control Theory*, pages 79–93. Birkhäuser, 1993.

[TE94] Michael Tittus and Bo Egardt. Control-law synthesis for linear hybrid systems. In *Proceedings of IEEE Conference on Decision and Control*, pages 961–966, Buena Vista, Florida, December 1994.

[YMLA94] Katerina Yamalidou, John Moody, Michael Lemmon, and Panos Antsaklis. Feedback control of petri nets based on place invariants. Technical Report of the ISIS Group ISIS-94-002, Dept. of Electrical Engineering, University of Notre Dame, 1994.

1901-ISIS 95