Hybrid System Modeling and Event Identification

Technical Report of the ISIS Group
at the University of Notre Dame
ISIS-93-002
January, 1993

Panos J. Antsaklis, Michael D. Lemmon, and James A. Stiver
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

Interdisciplinary Studies of Intelligent Systems

Hybrid System Modeling and Event Identification

Panos J. Antsaklis, Michael Lemmon *, and James A. Stiver

Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556 USA

Abstract. Hybrid control systems contain two distinct types of systems, continuous state and discrete-state, that interact with each other. Their study is essential in designing sequential supervisory controllers for continuous-state systems, and it is central in designing control systems with high degree of autonomy.

After an introduction to intelligent autonomous control and its relation to hybrid control, models for the plant, controller, and interface are introduced. The important role of the interface is discussed at length. System theoretic issues are addressed and the concepts of determinism and quasideterminism are introduced and studied. The relation to the theory of logical discrete event systems is shown and discussed. When the system changes, online identification supervisory control is desirable. To meet the demanding computing requirements, event identification is performed using inductive inference algorithms.

* The partial financial support of the National Science Foundation (IRI91-09298) is acknowledged

Table of Contents

1 Introduction

Hybrid control systems contain two distinct types of systems, continuous and discrete-state that interact with each other. An example of such a system is the heating and cooling system of a typical home. Here the furnace and air conditioner together with the home's heat loss dynamics can be modeled as continuous-state, (continuous-time) system which is being controlled by a discrete-state system, the thermostat. Other examples include systems controlled by bang-bang control or via methods based on variable structure control. Hybrid control systems also appear as part of *Intelligent Autonomous Control Systems*. Being able to control a continuous-state system using a discrete-state supervisory controller is a central problem in designing control systems with high degrees of autonomy. This is further discussed below. The analysis and design of hybrid control systems requires the development of an appropriate mathematical framework. That framework must be both powerful and simple enough so it leads to manageable descriptions and efficient algorithms for such systems. Recently, attempts have been made to study hybrid control systems in a unified, analytical way and a number of results have been reported in the literature [Benveniste 1990] [Gollu 1989] [Grossman 1992] [Holloway 1992] [Kohn 1992] [Lemmon 1993b] [Nerode 1992] [Passino 1991b] [Peleties 1988] [Peleties 1989] [Stiver 1991a] [Stiver 1991b] [Stiver 1991c] [Stiver 1992] [Stiver 1993].

In this chapter, a novel approach to hybrid systems modeling and control is described. Descriptions of the plant to be controlled, the controller and the interface are given in Section 2. The important role of the interface is discussed at length. In Section 3, certain system theoretic questions are addressed. In particular, the concepts of determinism and quasideterminism are introduced and results are given. It is then shown how logical Discrete Event System (DES) models can be used to formulate the hybrid control problem, thus taking full advantage of existing results on DES controller design [Cassandras 1990] [Ozveren 1991] [Passino 1989a] [Passino 1989b] [Passino 1991a] [Passino 1992a] [Ramadge 1987] [Ramadge 1989] [Wonham 1987] for hybrid control systems. When the system to be controlled is changing, these fixed controllers may not be adequate to meet the control goals. In this case it is desirable to identify the plant and derive the control law on line, and this is addressed in Section 4. Inductive inference methods are used to identify plant events in a computationally efficient manner.

In the rest of the introduction the important role hybrid control systems play in the design of Intelligent Autonomous Control systems is discussed and explained. In this way, the hybrid control problem can be seen in the appropriate setting so that its importance in the control of very complex systems may be fully understood and appreciated. Further discussion can be found in [Passino 1993]; see [Albus 1981] [Antsaklis 1989] [Antsaklis 1991] [Antsaklis 1993b] [Antsaklis 1993a] [Antsaklis 1993c] [IEEE Computer 1989] [Passino 1993] [Saridis 1979] [Saridis 1985] [Saridis 1987] [Saridis 1989a] [Zeigler 1984] for more information on intelligent control.

It is appropriate to first explain what is meant by the term Intelligent Autonomous Control used above. 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, a high degree of autonomy is desired. To achieve this autonomy, high level decision making techniques for reasoning under uncertainty must be utilized. These techniques, if used by humans, may be attributed to intelligence. 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, 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 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.

1.1 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 the late 18th century 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 signaled the end of the era of intuitive invention. The performance of the flyball governor was sufficient to meet the control needs of the day. 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 where systems based on intuitive invention rather than quantitative theory can be rather limited. 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 a need is quite apparent in intelligent autonomous control systems and in particular in hybrid control systems.

Conventional control design methods: 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. When the uncertainties in the plant and environment are large, the fixed 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.

1.2 Intelligent Control for High Autonomy Systems

There are cases where we need to significantly increase the operating range of control systems. 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 unmodeled 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. It is useful to keep in mind an example, the Houston Control example . It is an example that sets goals for the future and it also teaches humility as it indicates how difficult demanding and complex autonomous systems can be. Currently, if there is a problem on the space shuttle, the problem is addressed by the large number of engineers working in Houston Control, the ground station. When the problem is solved the specific detailed instructions about how to deal with the problem are sent to the shuttle. Imagine the time when we will need the tools and expertise of all Houston Control engineers aboard the space shuttle, space vehicle, for extended space travel.

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 control design approach taken here is a bottom-up approach. One turns to more sophisticated controllers only if simpler ones cannot meet the required objectives. 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 indeed.*

1.3 An Intelligent High Autonomy Control System Architecture For Future Space Vehicles

To illustrate the concepts and ideas involved and to provide a more concrete framework to discuss the issues, a hierarchical functional architecture of an intelligent controller that is used to attain high degrees of autonomy in future space vehicles is briefly outlined; full details can be found in [Passino 1993]. This hierarchical architecture has three levels, the Execution Level, the Coordination Level, and the Management and Organization Level. The architecture exhibits certain characteristics, which have been shown in the literature to be necessary and desirable in autonomous intelligent systems.

It is important at this point to comment on the choice for a hierarchical architecture. Hierarchies offer very convenient ways to describe the operation of complex systems and deal with computational complexity issues, and they are used extensively in the modeling of intelligent autonomous control systems. Such hierarchical approach is taken here (and in [Passino 1993]) to study intelligent autonomous and hybrid control systems.

Architecture Overview: The overall functional architecture for an autonomous controller is given by the architectural schematic of the figure below. This is a functional architecture rather than a hardware processing one; therefore, it does not specify the arrangement and duties of the hardware used to implement the functions described. Note that the processing architecture also depends on the characteristics of the current processing technology; centralized or distributed processing may be chosen for function implementation depending on available computer technology.

Fig. 1. Intelligent Autonomous Controller Functional Architecture

The architecture in Figure 1 has three levels; this is rather typical in the Intelligent Control literature. At the lowest level, the Execution Level, there is the interface to the vehicle and its environment via the sensors and actuators. At the highest level, the Management and Organization Level, there is the interface to the pilot and crew, ground station, or onboard systems. The middle level, called the Coordination Level, provides the link between the Execution Level and the Management Level. Note that we follow the somewhat standard viewpoint that there are three major levels in the hierarchy. It must be stressed that the system may have more or fewer than three levels. Some characteristics of the system which dictate the number of levels are the extent to which the operator can intervene in the system's operations, the degree of autonomy or level of intelligence in the various subsystems, the hierarchical characteristics of the plant. Note however that the three levels shown here in figure are applicable to most architectures of autonomous controllers, by grouping together sublevels of the architecture if necessary. As it is indicated in the Figure, the lowest, Execution Level involves conventional control algorithms, while the highest, Management and Organization Level involves only higher level, intelligent, decision making methods. The Coordination Level is the level which provides the interface between the actions of the other two levels and it uses a combination of conventional and intelligent decision making methods. The sensors and actuators are implemented mainly with hardware. Software and perhaps hardware are used to implement the Execution Level. Mainly software is used for both the Coordination and Management Levels. There are multiple copies of the control functions at each level, more at the lower and fewer at the higher levels. See [Passino 1993] for an extended discussion of the issues involved.

Hybrid control systems do appear in the intelligent autonomous control system framework whenever one considers the Execution level together with control functions performed in the higher Coordination and Management levels. Examples include expert systems supervising and tuning conventional controller parameters, planning systems setting the set points of local control regulators, sequential controllers deciding which from a number of conventional controllers is to be used to control a system, to mention but a few. One obtains a hybrid control system of interest whenever one considers controlling a continuous-state plant (in the Execution level) by a control algorithm that manipulates symbols, that is by a discrete-state controller (in Coordination and/or Management levels).

1.4 Quantitative Models

For highly autonomous control systems, normally the plant is so complex that it is either impossible or inappropriate to describe it with conventional mathematical system models such as differential or difference equations. Even though it might be possible to accurately describe some system with highly complex nonlinear differential equations, it may be inappropriate if this description makes subsequent analysis too difficult or too computationally complex to be useful. The complexity of the plant model needed in design depends on both the complexity of the physical system and on how demanding the design specifications are. There is a tradeoff between model complexity and our ability to perform analysis on the system via the model. However, if the control performance specifications are not too demanding, a more abstract, higher level, model can be utilized, which will make subsequent analysis simpler. This model intentionally ignores some of the system characteristics, specifically those that need not be considered in attempting to meet the particular performance specifications. For example, a simple temperature controller could ignore almost all dynamics of the house or the office and consider only a temperature threshold model of the system to switch the furnace off or on.

The quantitative, systematic techniques for modeling, analysis, and design of control systems are of central and utmost practical importance in conventional control theory. Similar techniques for intelligent autonomous controllers do not exist. This is mainly due to the hybrid structure (nonuniform, nonhomogeneous nature) of the dynamical systems under consideration; they include both continuous-state and discrete-state systems. Modeling techniques for intelligent autonomous systems must be able to support a macroscopic view of the dynamical system, hence it is necessary to represent both numeric and symbolic information. The nonuniform components of the intelligent controller all take part in the generation of the low level control inputs to the dynamical system, therefore they all must be considered in a complete analysis. Therefore the study of modeling and control of hybrid control systems is essential in understanding highly autonomous control systems.

2 Hybrid Control System Modeling

The hybrid control systems considered here consist of three distinct levels; see Figure 2. The controller is a discrete-state system, a sequential machine, seen as a Discrete Event System (DES). The controller receives, manipulates and outputs events represented by symbols. The plant is a continuous-state system typically modeled by differential/difference equations and it is the system to be controlled by the discrete-state controller. The plant receives, manipulates and outputs signals represented by real variables that are typically (piecewise) continuous. The controller and the plant communicate via the interface that translates plant outputs into events for the controller to use, and controller output events into command signals for the plant input. The interface can be seen as consisting of two subsystems: the event generator that senses the plant outputs and generates symbols representing plant events, and the actuator that translates the controller symbolic commands into piecewise constant plant input signals.

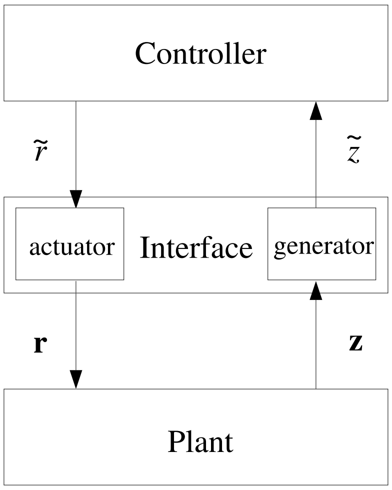

Fig. 2. Hybrid Control System

To be able to develop a useful mathematical framework we keep the interface as simple as possible; this is further discussed below. The interface determines the events the controller sees that uses to decide the appropriate control action. If the plant and the interface are taken together the resulting system is a DES, called the *DES Plant*, that the controller sees and attempts to control. Another way of expressing this is that the DES controller only sees a more abstract model of the plant; a higher level less detailed plant model than the differential / difference equation model. The complexity of this more abstract DES plant model depends on the interface. It is therefore very important to understand the issues involved in the interface design so that the appropriate DES model is simple enough so to lead to a low complexity controller. It should be noted that this lower complexity is essential for real time adaptation of hybrid control systems. All these issues pointed out here are discussed in detail later in this chapter.

It is important to identify the important concepts and develop an appropriate mathematical framework to describe hybrid control systems. Here the logical DES theory and the theory of automata is used. The aim is to take advantage as much as possible of the recent developments in the analysis and control design of DES. These include results on controllability, observability, stability of DES and algorithms for control design among others. We first present a flexible and tractable way of modeling hybrid control systems. Our goal is to develop a model which can adequately represent a wide variety of hybrid control systems, while remaining simple enough to permit analysis. We then present a methods which can be used to analyze and aid in the design of hybrid control systems. These methods relate to the design of the interface which is a necessary component of a hybrid system and its particular structure reflects both the dynamics of the plant and the aims of the controller.

Below the plant, interface and controller are described first. The assumptions made and the generality of the models are discussed. In Section 3, the DES plant model is then derived and the concepts of determinism and quasideterminism are introduced and certain results are shown. The description of the generator in the interface via covers is discussed. Controllability of the DES plant model is studied. The selection of the interface is discussed at length and the fundamental issues are identified. Connections to Ramadge-Wonham model are shown, the difficulties involved are indicated, and some recent results are outlined. Simple examples are used throughout to illustrate and explain. Note that most of these results can be found in [Stiver 1992].

A hybrid control system, can be divided into three parts as shown in Figure 2. The models we use for each of these three parts, as well as the way they interact are now described.

2.1 Plant

The system to be controlled, called the plant, is modeled as a time-invariant, continuous-time system. This part of the hybrid control system contains the entire continuous-time portion of the system, possibly including a continuous-time controller. Mathematically, the plant is represented by the familiar equations

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

$$\mathbf{z} = g(\mathbf{x}) \tag{2}$$

where $\mathbf{x} \in \Re^n$, $\mathbf{r} \in \Re^m$, and $\mathbf{z} \in \Re^p$ are the state, input, and output vectors respectively. $f : \Re^n \times \Re^m \to \Re^n$ and $g : \Re^n \to \Re^p$ are functions. For the purposes of this work we assume that $\mathbf{z} = \mathbf{x}$. Note that the plant input and output are continuous-time vector valued signals. Bold face letters are used to denote vectors and vector valued signals.

2.2 Controller

The controller is a discrete event system which is modeled as a deterministic automaton. This automaton can be specified by a quintuple, $\{\tilde{S}, \tilde{Z}, \tilde{R}, \delta, \phi\}$, where $\tilde{S}$ is the (possibly infinite) set of states, $\tilde{Z}$ is the set of plant symbols, $\tilde{R}$ is the set of controller symbols, $\delta : \tilde{S} \times \tilde{Z} \to \tilde{S}$ is the state transition function, and $\phi : \tilde{S} \to \tilde{R}$ 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 $\tilde{Z}$ are called plant symbols and are generated by the occurrence of events in the plant. The action of the controller can be described by the equations

$$\tilde{s}[n] = \delta(\tilde{s}[n-1], \tilde{z}[n]) \tag{3}$$

$$\tilde{r}[n] = \phi(\tilde{s}[n]) \tag{4}$$

where $\tilde{s}[n] \in S$, $\tilde{z}[n] \in \tilde{Z}$, 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 a sequence. The input and output signals associated with the controller are asynchronous sequences of symbols, rather than continuous-time signals. Notice that there is no delay in the controller. The state transition, from $\tilde{s}[n-1]$ to $\tilde{s}[n]$, and the controller symbol, $\tilde{r}[n]$, occur immediately when the plant symbol $\tilde{z}[n]$ occurs.

Tildes are used to indicate that the particular set or signal is made up of symbols. For example, $\tilde{Z}$ is the set of plant symbols and $\tilde{z}$ is a sequence of plant symbols. An argument in brackets, e.g. $\tilde{z}[n]$, represents the nth symbol in the sequence $\tilde{z}$. A subscript, e.g. $\tilde{z}_i$, is used to denote a particular event from a set.

2.3 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 interface consists of two memoryless maps, γ and α. The first map, called the actuating function or actuator, $\gamma : \tilde{R} \to \Re^m$, converts a sequence of controller symbols to a piecewise constant plant input as follows

$$\mathbf{r} = \gamma(\tilde{r}) \tag{5}$$

The plant input, $\mathbf{r}$, 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. The second map, the plant symbol generating function or generator, $\alpha : \Re^n \to \tilde{Z}$, is a function which maps the state space of the plant to the set of plant symbols as follows

$$\tilde{z} = \alpha(\mathbf{x}) \tag{6}$$

It would appear from Equation 6 that, as $\mathbf{x}$ changes, $\tilde{z}$ also continuously changes. That is, there is a continuous generation of plant symbols by the interface because each state is mapped to a symbol. This is not the case due to the way α is defined as will now be explained.

The plant symbol generating function, α, is designed based on an open covering of the state space of the plant. Consider a collection of p open subsets in $\Re^n$ which form an open cover for the plant state space. Let this collection be represented as

$$C = \{c_1 c_2 ... c_p\} \tag{7}$$

The collection consists of open subsets, each subset is called a covering event. Let the ith covering event, c_i, be associated with a unique covering symbol, $\tilde{c}_i$. The "alphabet" of covering symbols can therefore be represented as

$$\tilde{C} = \{\tilde{c}_1 \tilde{c}_2 ... \tilde{c}_p\} \tag{8}$$

These covering symbols are used to define the plant symbols as follows

$$\tilde{z} = \alpha(\mathbf{x}) = \{\tilde{c}_i : \mathbf{x} \in c_i\} \tag{9}$$

As shown in Equation 9, a plant symbol is a collection of covering symbols which defines a region in the state space. It is convenient to treat this collection as a symbol. This will be done in this and the following section. A plant symbol is generated only when a new event first occurs. The overall effect is that the state space of the plant is partitioned into a number of regions and each is associated with a unique plant symbol which is generated whenever the state enters that region. Note that these regions form the equivalence classes of α.

The use of open covers to describe the state space is motivated by several things. The generator, α, must take subsets of states onto a "unique" plant symbol. The representation should be "well posed" so open sets are used so that a small change in state can be made without changing the event. The representation must also be complete, meaning every state must be contained within a plant event. These conditions suggest that α should realize an open covering of the state space.

2.4 Comments on the Generality of the Model

The model described above may appear at first to be too limited but this is not the case. The simplicity of this model is its strength and it does not reduce its flexibility when modeling a hybrid control system. It is tempting to add complexity to the interface, however this typically leads to additional mathematical difficulties that are not necessary. Consider first the function γ which maps controller symbols to plant inputs. Our model features only constant plant inputs, no ramps, sinusoids, or feedback strategies. The reasons for this are two fold. First, in order for the interface to generate a nonconstant signal or feedback signal it must contain components which can be more appropriately included in the continuous time plant, as is done in the model above. Second, making the interface more complex will complicate the analysis of the overall system. Keeping the function γ as a simple mapping from each controller symbol to a unique numeric value is the solution.

The interface could also be made more complex by generalizing the definition of a plant symbol. A plant symbol is defined solely by the current plant state, but this could be expanded by defining a plant symbol as being generated following the occurrence of a specific series of conditions in the plant. For example, the interface could be made capable of generating a symbol which is dependent upon the current and previous values of the state. However, doing this entails including dynamics in the interface which actually belong in the controller. The controller, as a dynamic system, is capable of using its state as a memory to keep track of previous plant symbols.

The key feature of this hybrid control system model is its simple and unambiguous nature, especially with respect to the interface. To enable analysis, hybrid control systems must be described in a consistent and complete manner. Varying the nature of the interface from system to system in an ad hoc manner, or leaving its mathematical description vague causes difficulties.

2.5 Examples

Example 1 - Thermostat/Furnace System This example will show how an actual physical system can be modeled and how the parts of the physical system correspond to the parts found in the model. The particular hybrid control system in this example consists of a typical thermostat and furnace. Assuming the thermostat is set at 70 degrees Fahrenheit, the system behaves as follows. If the room temperature falls below 70 degrees the furnace starts and remains on until the room temperature exceeds 75 degrees. At 75 degrees the furnace shuts off. For simplicity, we will assume that when the furnace is on it produces a constant amount of heat per unit time.

The plant in the thermostat/furnace hybrid control system is made up of the furnace and room. It can be modeled with the following differential equation

$$\dot{\mathbf{x}} = .0042(T_0 - \mathbf{x}) + 2step(\mathbf{r}) \tag{10}$$

where the plant state, $\mathbf{x}$, is the temperature of the room in degrees Fahrenheit, the input, $\mathbf{r}$, is the voltage on the furnace control circuit, and T_0 is the outside temperature. The units for time are minutes. This model of the furnace is a simplification, but it is adequate for this example.

The remainder of the hybrid control system is found in the thermostat which is pictured in Figure 3. As the temperature of the room varies, the two strips of metal which form the bimetal band expand and contract at different rates thus causing the band to bend. As the band bends, it brings the steel closer to one side of the glass bulb. Inside the bulb, a magnet moves toward the nearest part of the steel and opens or closes the control circuit in the process. The bimetal band effectively partitions the state space of the plant, $\mathbf{x}$, as follows

$$\alpha(\mathbf{x}) = \begin{cases} \tilde{z}_1 \text{ if } \quad \mathbf{x} < 70 \\ \tilde{z}_2 \text{ if } 70 < \mathbf{x} < 75 \\ \tilde{z}_3 \text{ if } \quad \mathbf{x} > 75 \end{cases}, \tag{11}$$

where the three symbols correspond to 1) steel is moved against the left side of the bulb, 2) band is relaxed, and 3) steel is moved against the right side of the bulb.

Inside the glass bulb is a magnetic switch which is the DES controller. It has two states because the switch has two positions, on and off. The DES controller input, $\tilde{z}$, is a magnetic signal because the symbols generated by the generator are conveyed magnetically. The state transition graph of this simple controller is shown in Figure 4. The output function of the controller is essentially the following

$$\phi(\tilde{s}_1) = \tilde{r}_1 \Leftrightarrow \text{close control circuit} \tag{12}$$

$$\phi(\tilde{s}_2) = \tilde{r}_2 \Leftrightarrow \text{open control circuit} \tag{13}$$

Fig. 3. Thermostat

Fig. 4. Controller for Thermostat/Furnace System

The contacts on the switch which open and close the control circuit can be thought of as the actuator, although there is no logical place to separate the actuator from the DES controller. The commands from the controller to the actuator are basically a formality here because the controller and actuator are mechanically one piece. With this in mind, the actuator operates as

$$\gamma(\tilde{r}_1) = 0 \tag{14}$$

$$\gamma(\tilde{r}_2) = 24 \tag{15}$$

Example 2 - Surge Tank This is another example to illustrate how a simple hybrid control system can be modeled. The system consists of a surge tank which is draining through a fixed outlet valve, while the inlet valve is being controlled by a discrete event system. The controller allows the tank to drain to a minimum level and then opens the inlet valve to refill it. When the tank has reached a maximum level, the inlet valve is closed. The surge tank is modeled by a differential equation,

$$\dot{\mathbf{x}} = \mathbf{r} - \mathbf{x}^{1/2} \tag{16}$$

where $\mathbf{x}$ is the liquid level and $\mathbf{r}$ is the inlet flow. The interface partitions the state space into three regions as follows

$$\alpha(\mathbf{x}) = \begin{cases} \tilde{z}_1 \textbf{ if } \quad \mathbf{x} > max \\ \tilde{z}_2 \textbf{ if } min < \mathbf{x} < max \\ \tilde{z}_3 \textbf{ if } \quad \mathbf{x} < min \end{cases}, \tag{17}$$

Thus when the level exceeds max, plant symbol $\tilde{z}_1$ is generated, and when the level falls below min, plant symbol $\tilde{z}_3$ is generated. The interface provides for two inputs corresponding to the two controller symbols $\tilde{r}_1$ and $\tilde{r}_2$ as follows

$$\gamma(\tilde{r}) = \begin{cases} 1 \text{ if } \tilde{r} = \tilde{r}_1 \\ 0 \text{ if } \tilde{r} = \tilde{r}_2 \end{cases}, \tag{18}$$

Since $\mathbf{r} = \gamma(\tilde{r})$, this means the inlet valve will be open following controller symbol $\tilde{r}_1$, and closed following controller symbol $\tilde{r}_2$.

The controller for the surge tank is a two state automaton which moves to state $\tilde{s}_1$ whenever $\tilde{z}_3$ is received, moves to state $\tilde{s}_2$ whenever $\tilde{z}_1$ is received and returns to the current state if $\tilde{z}_2$ is received. Furthermore $\phi(\tilde{s}_1) = \tilde{r}_1$ and $\phi(\tilde{s}_2) = \tilde{r}_2$.

3 System Theoretic Issues

3.1 The DES Plant Model

If the plant and interface of a hybrid control system are viewed as a single component, this component behaves like a discrete event system. It is advantageous to view a hybrid control system this way because it allows it to be modeled as two interacting discrete event systems which are more easily analyzed than the system in its original form. The discrete event system which models the plant and interface is called the DES Plant Model and is modeled as an automaton similar to the controller. The automaton is specified by a quintuple, $\{\tilde{P}, \tilde{Z}, \tilde{R}, \psi, \lambda\}$, where $\tilde{P}$ is the set of states, $\tilde{Z}$ and $\tilde{R}$ are the sets of plant symbols and controller symbols, $\psi : \tilde{P} \times \tilde{R} \to \tilde{P}$ is the state transition function, and $\lambda : \tilde{P} \to \tilde{Z}$ is the output function. The behavior of the DES plant is as follows

$$\tilde{p}[n+1] = \psi(\tilde{p}[n], \tilde{r}[n]) \tag{19}$$

$$\tilde{z}[n] = \lambda(\tilde{p}[n]) \tag{20}$$

where $\tilde{p}[n] \in \tilde{P}, \tilde{r}[n] \in \tilde{R}$, and $\tilde{z}[n] \in \tilde{Z}$. There are two differences between the DES plant model and the controller. First, as can be seen from Equation 19, the state transitions in the DES plant do not occur immediately when a controller symbol occurs. This is in contrast to the controller where state transitions occur immediately with the occurrence of a plant symbol. The second difference is that the automaton which models the DES plant may be non-deterministic, meaning $\tilde{p}[n+1]$ in Equation 19 is not determined exactly but rather is limited to some subset of $\tilde{P}$. The reason for these differences is that the DES plant model is a simplification of a continuous-time plant and an interface. This simplification results in a loss of information about the internal dynamics, leading to non-deterministic behavior.

The set of states, $\tilde{P}$, of the DES plant is based on the open covering realized in the interface. Specifically, each state in $\tilde{P}$ corresponds to a region, in the state space of the continuous-time plant, which is equivalent under α. Thus there is a one-to-one correspondence between the set of states, $\tilde{P}$, and the set of plant symbols, $\tilde{Z}$. It is this relationship between the states of the DES plant model and the plant symbols which forms the basis for the work described in this section. It can be used to develop an expression for the state transition function, ψ. Starting with the continuous-time plant, we integrate Equation 19 to get the state after a time t, under constant input $\mathbf{r} = \gamma(\tilde{r}_k)$

$$\mathbf{x}(t) = F_k(\mathbf{x}_0, t) \tag{21}$$

Here $\mathbf{x}_0$ is the initial state, t is the elapsed time, and $\tilde{r}_k \in \tilde{R}$. $F_k(\mathbf{x}_0, t)$ is obtained by integrating $f(\mathbf{x}, \mathbf{r})$, with $\mathbf{r} = \gamma(\tilde{r}_k)$. Next we define

$$\hat{F}_k(\mathbf{x}_0) = F_k(\mathbf{x}_0, t), \tag{22}$$

where

$$t = \min_t \{t | \alpha(F(\mathbf{x}_0, t)) \neq \alpha(\mathbf{x}_0)\} \tag{23}$$

Equation 22 gives the state, $\mathbf{x}$, where it will first cross into a new region. Now the dynamics of the DES plant model can be derived from Equations 5, 6, 22.

$$\tilde{z}[n+1] = \lambda(\psi(\tilde{p}[n], \tilde{r}[n])) \tag{24}$$
$$\tilde{z}[n+1] = \alpha(\hat{F}_k(\mathbf{x}_0)) \tag{25}$$
$$\psi(\tilde{p}[n], \tilde{r}[n]) = \lambda^{-1}(\alpha(\hat{F}_k(\mathbf{x}_0))) \tag{26}$$

where $\tilde{r}[n] = \tilde{r}_k$ and $\{\mathbf{x}_0 \in \{\mathbf{x}|\alpha(\mathbf{x}) = \lambda(\tilde{p}[n])\}$. As can be seen, the only uncertainty in Equation 26 is the value of $\mathbf{x}_0$. $\mathbf{x}_0$ is the state of the continuous-time plant at the time of the last plant symbol, $\tilde{z}[n]$, i.e. the time that the DES plant entered state $\tilde{p}[n]$. $\mathbf{x}_0$ is only known to within an equivalence class of α. The condition for a deterministic DES plant is that the state transition function, ψ, must be invariant to this uncertainty.

Definition 1. A DES is *deterministic* iff for any state and any input, there is only one possible subsequent state.

The following theorem gives the conditions upon the hybrid control system such that the DES plant will be deterministic.

Theorem 2. *The DES plant will be deterministic iff given any $\tilde{p}[n] \in P$ and $\tilde{r}_k \in R$, there exists $\tilde{p}[n+1] \in P$ such that for every $\mathbf{x}_0 \in \{\mathbf{x}|\alpha(\mathbf{x}) = \lambda(\tilde{p}[n])\}$ we have $\alpha(\hat{F}_k(\mathbf{x}_0)) = \lambda(\tilde{p}[n+1])$.*

Proof: Notice that the set $\{\mathbf{x}|\alpha(\mathbf{x}) = \lambda(\tilde{p}[n])\}$ represents the set of all states, $\mathbf{x}$, in the continuous-time plant which could give rise to the state $\tilde{p}[n]$ in the DES plant. The theorem guarantees that the subsequent DES plant state, $\tilde{p}[n+1]$, is unique for a given input and thus the DES plant is deterministic.

To prove that the theorem is necessary, assume that it does not hold. There must then exist a $\tilde{p}[n] \in \tilde{P}$ and $\tilde{r}_k \in \tilde{R}$ such that no $\tilde{p}[n+1]$ exists to satisfy the condition: $\alpha(\hat{F}_k(\mathbf{x}_0)) = \lambda(\tilde{p}[n+1])$ for every $\mathbf{x}_0 \in \{\mathbf{x}|\alpha(\mathbf{x}) = \lambda(\tilde{p}[n])\}$. This is not a deterministic system because there is uncertainty in the state transition for at least one state and input. □

Theorem 2 states that the DES plant will be deterministic if all the state trajectories in the continuous-time plant, which start in the same region and are driven by the same input, move to the same subsequent region.

3.2 Double Integrator

To illustrate the DES plant model, an example of a hybrid control system containing a double integrator is given. Double integrators often arise in systems. For example, a satellite equipped with a thruster will behave as a double integrator when the thrust is considered the input and the velocity and position are the two states.

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathbf{r} \tag{27}$$

The general control goal in this system, which motivates the design of the interface, is to move the state of the double integrator between the four quadrants of the state-space. In the interface, the function a partitions the state space into four regions as follows,

$$\alpha(\mathbf{x}) = \begin{cases} \tilde{z}_1 & \text{if} \quad x_1, x_2 > 0 \\ \tilde{z}_2 & \text{if} \;\; x_1 < 0, x_2 > 0 \\ \tilde{z}_3 & \text{if} \quad x_1, x_2 < 0 \\ \tilde{z}_4 & \text{if} \;\; x_1 > 0, x_2 < 0 \end{cases}, \tag{28}$$

and the function γ provides three set points,

$$\gamma(\tilde{r}) = \begin{cases} -10 & \text{if } \tilde{r} = \tilde{r}_1 \\ 0 & \text{if } \tilde{r} = \tilde{r}_2 \\ 10 & \text{if } \tilde{r} = \tilde{r}_3 \end{cases}, \tag{29}$$

So whenever the state of the double integrator enters quadrant 1, for example, the plant symbol $\tilde{z}_1$ is generated. When the controller (which is unspecified) generates controller symbol $\tilde{r}_1$, the double integrator is driven with an input of -10.

Now we know that the DES plant will have four states because there are four regions in the state space of the actual plant. By examining the various state trajectories given by Equation 30, we can find the DES plant which is shown in Figure 5. Equation 30 is obtained by integrating Equation 27 and adding $\mathbf{x}(0)$.

$$\mathbf{x} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \mathbf{x}_0 + \begin{bmatrix} .5t^2 \\ t \end{bmatrix} \gamma(\tilde{r}) \tag{30}$$

Fig. 5. DES Plant Model for Double Integrator

As can be seen in Figure 5, the DES plant is not deterministic. If we consider $\tilde{p}[n] = \tilde{p}_2$ and $\tilde{r}[n] = \tilde{r}_1$, there exists no uniquely defined $\tilde{p}[n+1]$, it could be either $\tilde{p}_1$

or $\tilde{p}_3$. This could present a problem in designing a controller for this system because it is not entirely predictable. In the following section a possible remedy for lack of determinism is presented and this example is revisited.

3.3 Partitioning and Quasideterminism

A particular problem in the design of a hybrid control system is the selection of the function α, which partitions the state-space of the plant into various regions. Since this partition is used to generate the plant symbols, it must be chosen to provide sufficient information to the controller to allow control without being so fine that it leads to an unmanageably complex system or simply degenerates the system into an essentially conventional control system. The partition must accomplish two goals. First it must give the controller sufficient information to determine whether or not the current state is in an acceptable region. For example, in an aircraft these regions may correspond to climbing, diving, turning right, etc. Second, the partition must provide enough additional information about the state, to enable the controller to drive the plant to an acceptable region. In an aircraft, for instance, the input required to cause the plane to climb may vary depending on the current state of the plane. So to summarize, the partition must be detailed enough to answer: 1) is the current state acceptable; and 2) which input can be applied to drive the state to an acceptable region.

In a hybrid control system, the controller needs information about the plant for two reasons. First the controller must be able to assess whether the plant is operating as desired or if some new control action is needed. Second, if control action is called for, the controller needs to know which control action will achieve the desired effect. Both of these tasks require information about the plant. Consider for example a climate control system in a building. To assess the current condition, the controller needs to know whether the temperature and humidity fall within a certain range of acceptable values. If not the controller needs additional, more detailed, information about which condition is unacceptable and how much and in which direction it must be changed to reach the desired range.

To design a partition, we can start by designing a primary partition to meet the first goal mentioned above. This primary partition will identify all the desired operating regions of the plant state space, so its design will be dictated by the control goals. The final partition will represent a refinement of the primary partition which enables the controller to regulate the plant to any of the desired operating regions, thus meeting the second goal. An obvious choice for the final partition is one which makes the DES plant deterministic and therefore guarantees that the controller will have full information about the behavior of the plant. In addition to being very hard to meet, this requirement is overly strict because the controller only needs to regulate the plant to the regions in the primary partition, not the final partition. For this reason we define quasideterminism, a weaker form of determinism. In the DES plant, the states which are in the same region of the primary partition can be grouped together, and if the DES plant is deterministic with respect to these groups, then we say it is quasideterministic. So if the DES plant is quasideterministic, then we may not be able to predict the next state exactly, but we will be able to predict its region of the primary partition and thus whether or not it is acceptable.

Definition 3. The DES plant will be quasideterministic iff given any $\tilde{p}[n] \in \tilde{P}$ and $\tilde{r}_k \in \tilde{R}$, there exists $\tilde{Q} \subset \tilde{P}$ such that for every $\mathbf{x}_0 \in \{\mathbf{x}|\alpha(\mathbf{x}) = \lambda(\tilde{p}[n])\}$ we have $\alpha_p(\hat{F}_k(\mathbf{x}_0)) = \lambda_p(\tilde{p}[n+1])$ where $\tilde{p}[n+1] \in \tilde{Q}$ and $\lambda_p(\tilde{q})$ is the same for all $\tilde{q} \in \tilde{Q}$. □

The functions α_p and λ_p are analogous to α and λ but apply to the primary partition. They are useful for comparing states but they are never implemented and their actual values are irrelevant. For example, if $\alpha_p(\mathbf{x}_1) = \alpha_p(\mathbf{x}_2)$, then $\mathbf{x}_1$ and $\mathbf{x}_2$ are in the same region of the primary partition. Or, if $\alpha_p(\mathbf{x}_1) = \lambda_p(\tilde{p}_1)$, then $\mathbf{x}_1$ is in the same region of the primary partition as $\tilde{p}_1$ in the DES plant. When used with α_p we define $\hat{F}$ as

$$\hat{F}_k(\mathbf{x}_0) = F_k(\mathbf{x}_0, t), \tag{31}$$

where

$$t = \min_t \{t | \alpha_p(F(\mathbf{x}_0, t)) \neq \alpha_p(\mathbf{x}_0)\} \tag{32}$$

We would like to find the coarsest partition which meets the conditions of Definition 1 for a given primary partition. Such a partition is formed when the equivalence classes of α are as follows,

$$E[\alpha] = \inf\{E[\alpha_p], E[\alpha_p \circ \hat{F}_k] | \tilde{r}_k \in R\} \tag{33}$$

Where we use $E[\bullet]$ to denote the equivalence classes of $\bullet$. The infimum, in this case, means the coarsest partition which is at least as fine as any of the partitions in the set.

Theorem 4. *The regions described by Equation (33) form the coarsest partition which generates a quasideterministic DES plant.*

Proof: First we will prove that the partition does, in fact, lead to a quasideterministic system. For any two states, $\mathbf{x}_1$ and $\mathbf{x}_2$, which are in the same equivalence class of α, we apply some control $\mathbf{r} = \gamma(\tilde{r}_k)$. The two states will subsequently enter new regions of the primary partition at $\hat{F}_k(\mathbf{x}_1)$ and $\hat{F}_k(\mathbf{x}_2)$ respectively. The actual regions entered are $\alpha_p(\hat{F}_k(\mathbf{x}_1))$ and $\alpha_p(\hat{F}_k(\mathbf{x}_2))$. Now according to Equation 33, if $\mathbf{x}_1$ and $\mathbf{x}_2$ are in the same equivalence class of α, then they are also in the same equivalence class of $\alpha_p \circ \hat{F}_k$. Therefore $\alpha_p(\hat{F}_k(\mathbf{x}_1)) = \alpha_p(\hat{F}_k(\mathbf{x}_2))$ and the system is quasideterministic.

Next we will prove that the partition is as coarse as possible. Assume there is a coarser partition which also generates a quasideterministic system. That is, there exists two states, $\mathbf{x}_3$ and $\mathbf{x}_4$, in the same region of the primary partition such that $\alpha(\mathbf{x}_3) \neq \alpha(\mathbf{x}_4)$, but $\alpha_p(\hat{F}_k(\mathbf{x}_3)) = \alpha_p(\hat{F}_k(\mathbf{x}_4))$ for *any* possible k. These two states would lie in the same equivalence class of $\alpha_p \circ \hat{F}_k$ for all $\tilde{r}_k \in \tilde{R}$ and therefore in the same equivalence class of $\inf\{E[\alpha_p], E[\alpha_p \circ \hat{F}_k | \tilde{r}_k \in \tilde{R}\}$. This violates the assumption that $\mathbf{x}_3$ and $\mathbf{x}_4$ do not lie in the same equivalence class of α, so two such states could not exist and therefore a coarser partition can not exist. □

Quasideterminism accomplishes its goal by causing the trajectories of the various states within a given region of the final partition, under the same control, to be invariant with respect to the regions of the primary partition.

We can return now to the double integrator discussed previously and us it to illustrate quasideterminism. The state space of the double integrator had been partitioned into the four quadrants and this gave rise to the nondeterministic DES plant shown in Figure 5. Using those four regions as the primary partition, a final partition can be obtain according to Theorem 4. This partition is shown in Figure 6 and the resulting DES plant is shown in Figure 7. The final partition refined the regions in quadrants II and IV, and the DES plant is now quasideterministic (in fact it is deterministic but unfortunately that is not generally the result).

Fig. 6. State Space Partition for Double Integrator

Note that the partition described in Equation 33 and discussed in Theorem 4 is not dependent upon any specific sequence of controller events. It is intended to yield a DES plant which is as "controllable" as possible, given the continuous-time plant and available inputs. If the specific control goals are know, it may be possible to derive a coarser partition which is still adequate. This can be done in an ad hoc fashion, for instance, by combining regions which are equivalent under the inputs which are anticipated when the plant is in those regions.

Selection of Control Action In hybrid control systems, the choice of the plant inputs which make up the range of the actuator, γ, play an important role in defining the system. At this time we have no way of systematically deriving a set of control actions which will achieve the desired control goals, either optimally or otherwise. We can assume that the control actions available are determined by the plant (positions of the various valves, switches, etc.) and thus represent a constraint on the controller design.

Fig. 7. DES Plant Model for Double Integrator

3.4 Connections to Existing DES Control Theory

A significant amount of work has been done on the analysis and design of discrete event systems, especially the design of controllers for discrete event systems. Since the controller of a hybrid control system is a DES and we can use a DES to represent the plant in a hybrid control system, we can apply many of the theories and techniques, which were developed for DES's, to hybrid control systems. In this section, we draw on some of this work.

Stability Several papers have been written dealing with the stability of discrete event systems, e.g. [Passino 1992a] and [Ozveren 1991]. In [Passino 1992a] the ideas of Lyapunov stability are applied to discrete event systems. These same techniques can be applied to the DES plant in a hybrid system. The states of the DES plant which are considered "desirable" are identified and a metric is defined on the remaining "undesirable" states. With a metric defined on the state space, finding a Lyapunov function will prove that the DES is stable. In the case of a hybrid control system, this interpretation of definition Lyapunov stability means the following. The state of the plant will remain with in the events which were deemed "desirable" and if it is perturbed from this area, the state will return to it. A detailed application of these results to hybrid control systems can be found in [Stiver 1991b].

In [Ozveren 1991] the stability of a DES is defined as the property that the state of the DES will visit a certain subset infinitely often. This subset is analogous to the "desirable" set mention above. In a hybrid control system, this would imply that the state of the plant could leave the subset of states but would eventually return.

Controllability Work has been done on the controllability of discrete event systems using the Ramadge-Wonham framework [Ramadge 1987] [Ramadge 1989] [Wonham 1987]. The DES models used in the Ramadge-Wonham framework differ from the models developed for hybrid control systems as described in this chapter, therefore the theorems and techniques cannot be applied directly, but must be adapted to work with a slightly different model.

The model developed by Ramadge and Wonham (henceforth RWM) features a generator and a supervisor, both DES's, which are analogous to the DES plant model and DES controller, respectively. There are, however, several differences which must be addressed first.

In the generator, the state transitions are divided into two sets, those which are controllable and those which are uncontrollable. The controllable state transitions, or events, can be individually enabled by a command from the supervisor, while the uncontrollable state transitions are always enabled. Also, the effect of the supervisor commands is not changed by the state of the generator. This is in contrast to our DES plant model where commands from the DES controller can enable one or more state transitions depending on the current state. The general inability to enable events individually and the dependence of DES controller commands upon the state of the DES plant model, are what differentiate the DES models used our work on hybrid control systems from the RWM.

The reason for the differences between the RWM and the model used for hybrid control systems is chiefly due to the fact that the RWM is suited to modeling actual discrete event systems, while the DES plant model is an abstraction of a continuous-time system. This means that a particular state of the DES plant corresponds to more than one state in the continuous-time plant.

Controllability in a DES can be characterized by the set of event sequences which can be made to occur in the DES plant, [Ramadge 1989]. This set is referred to as the language of the particular DES. When under control, the DES will exhibit behavior which lies in a subset of its language. A theorem has been developed to determined whether a given RWM DES can be controlled to a desired language and if not, what is the greatest portion of the desired language which can be achieved via control. With appropriate modifications this theorem can be applied to the DES plant to determine whether a given control goal is possible.

If a desired behavior (i.e. language) is not attainable for a given controlled DES, it may be possible to find a more restricted behavior which is. If so, the least restricted behavior is desirable. [Wonham 1987] provides a method for finding this behavior which is referred to as the *supremal sublanguage* of the desired language.

4 Event Identification

Hybrid dynamical systems provide a convenient tool for the analysis and design of supervisory control systems. A supervisory control system arises when a discrete event system is used to supervise the behaviour of a plant by the issuance of logical control directives. The hybrid system framework shown in figure 2 clearly illustrates this architecture. In this case, the DES or supervisor is used to control the continuous-state plant. The supervisor, of course, is a symbol manipulation system where the logical symbols have "meanings" grounded in nonsymbolic external "events". For example, a certain set of temperatures and pressure measurements may be indicative of a potential system failure. In this case, we would like to associate the "nonsymbolic" measurements with a "symbolic" label called "FAILURE". Therefore the supervisor's computations represent the manipulation of abstractions about the plant's current state. The use of such high-level abstractions (representations) of system state to control the system is sometimes called "intelligent" control.

This notion of intelligence, however, is singularly unsatisfying. Note that the action of the controller relies on the prior interpretation assigned to the plant and control symbols. Therefore the "intelligence" of the system lies in the interpretation of these symbols. The "intelligent" choices, however, were made by the human designer, not by the machine. Therefore it is the designer, rather than the machine which is intelligent. This same fundamental argument has been previously leveled against production based inference as a model for human cognition [Searle 1984]. Essentially, it asserts that the "blind" manipulation of symbols is not sufficient to render a system intelligent.

The reduction of the plant to an effective DES plant model, represents one way of designing so-called "intelligent" controllers. This approach to design was discussed briefly in the preceding section. However, this design approach represents the precise disembodiment of controller symbol and event, which was immediately discussed above. In this regard, an approach to supervisory control which assumes a priori symbol/event bindings cannot be considered an "intelligent" control system. Intelligence will only arise when the system is capable of determining its own event/symbol bindings. This requires that any intelligent system solve what may be called the event identification problem. The relationship between this "event identification" problem and more traditional issues in artificial intelligence such as the symbol grounding problem [Harnad 1990] is discussed in one of the closing subsections of this chapter.

Whether or not the symbolic manipulations of a computational system constitutes intelligence can, no doubt, be argued endlessly. There is, however, a much more pragmatic reason for considering such a system undesirable. If we consider those applications for which supervisory control systems are intended, it is immediately apparent that supervision is meant for complex and and unpredictable systems. For such systems, prior plant knowledge or complete plant knowledge may be impossible. This means that "events" which are defined with respect to an assumed plant structure, may change unexpectedly. If this is the case, then it is well within the realm of possiblity for our not-so intelligent supervisor to happily chunk away and produce of stream of nonsensical control symbols. The reason this occurs, of course, is because the supervisor really doesn't understand the significance of the symbols it is manipulating. If we wish to call this un-intelligent processing, that is fine. The end result is the same, however, a system whose autonomy is limited by the designer's initial assignment of symbol bindings. Therefore, a more pragmatic reason for requiring event identification of "intelligent" control systems is that it will undoubtably lead to increased system autonomy. The issue of autonomy in intelligent control was discussed thoroughly in the introduction. It is the need for such autonomy that really motivates the requirement for event identification in hybrid systems. As will be pointed out in one of the closing subsections, this ability is also consistent with notions of "intelligence" stemming from the symbolic and subsymbolic AI communities [Chalmers 1992].

The preceding discussion therefore indicates that an important problem in hybrid system control is the identification of events. How does one choose events which are consistent with the desired control objectives? Is it possible for the system to identify its own set of "optimal" events. This section presents one example of how such **event identification** can be accomplished. The problem of event identification can be view in a variety of contexts. For example, consider a system which has the general architecture shown in figure 2. Assume that the plant uses a collection of control policies, so that the plant's differential equation has the form

$$\dot{\mathbf{x}} = \sum_{i=1}^{m} r_i f_i(\mathbf{x}) \tag{34}$$

where $\mathbf{x}$ is the state vector and $\mathbf{r}$ is an m-vector of "coordination" coefficients. The individual vector fields can be seen as "control policies" which are coordinated through the specification of the vector $\mathbf{r}$. In figure 2, it can now be seen that the binding of plant/control symbols with subsets of the state space determines the behaviour of this system. One side of the problem, involves determining plant symbol and event bindings which allow a determinisitic or quasideterministic plant DES (see section 3). The solution of this problem yields a design for the interface's event generator. A system which can learn a set of bindings consistent with deterministic behaviour will have gone a long way in learning to control itself. Another side of the problem focuses on learning the symbol bindings between the control symbols, $\tilde{r}$, and vectors $\mathbf{r}$. The solution to this problem yields a design for the interface's actuator. System's which are capable of forming the event/symbol bindings consistent with with control objectives (i.e. determinism or controllability) will go a long way towards realizing "intelligent" control systems exhibiting a high degree of autonomy.

The following subsections provide a specific example of a hybrid system which can automatically learn event/symbol bindings. The example system is a variable structure system and the symbol bindings are learned with regard to invariant sets generated by the plant's dynamics. Early work on this was done in [Lemmon 1992] and refined in [Lemmon 1993a] with regard to the binding of plant symbols. Considerations on the binding of control symbols was discussed in [Lemmon 1993b]. In all of this work it was shown that bindings could be learned in finite time with a sample complexity that scales in a polynomial manner with plant complexity. The remainder of this section is organized as follows. Subsection 4.1 discusses the example problem which is referred to in this section as the invariant subspace identification (ISID) problem. Subsection 4.2 introduces the learning algorithm. This algorithm consists of two procedures called the oracle and the update procedure. These procedures are derived in subsections 4.3 and 4.4. The convergence and complexity properties of this learning procedure are discussed in section 4.5. An example of this algorithm's use is illustrated in section 4.6. The importance of the following example is that it provides a concrete example of a hybrid system which learns to "identify" its own events in a computationally efficient manner. Some issues and concerns associated with this example are discussed in subsection 4.7. The presented algorithm also provides a slightly different perspective on the relationship between intelligence and control. The central issue in this perspective is the so-called "symbol grounding" problem [Harnad 1990]. This novel perspective on "intelligence" will be discussed in subsection 4.8.

4.1 Invariant Subspace Identification (ISID) Problem

The hybrid system under consideration is assumed to have a very special form. Specifically, it will be assumed that the plant's dynamics are represented by the following differential equations.

$$\dot{\mathbf{x}} = \sum_{i=1}^{2} r_i f_i(\mathbf{x}) \tag{35}$$

where $\mathbf{x} \in \Re^n$ is the state vector, f_1 and f_2 are smooth mappings from $\Re^n$ onto $\Re^n$. It is also assumed that the vector $\mathbf{r} = (r_1, r_2)^t$ takes on the values of $(0,0)$,$(1,0)$ or $(0,1)$. The resulting plant is therefore a variable structure system [Utkin 1977] [DeCarlo 1988].

The interface generator for this hybrid system will be formed with respect to a covering collection consisting of two events, c^+ and c^-.

$$c^+ = \left\{\mathbf{x} \in \Re^n : \mathbf{s}^t\mathbf{x} > -|\alpha|\right\} \tag{36}$$

$$c^- = \left\{\mathbf{x} \in \Re^n : \mathbf{s}^t\mathbf{x} < |\alpha|\right\} \tag{37}$$

where α is a real number and $\mathbf{s}$ is an n-dimensional real vector. These two events form overlapping linear halfspaces. The covering generates three distinct plant events. The plant state either lies in the deadzone formed by the intersection of c^+ and c^- or else it lies only in one of the halfspaces. Therefore the plant symbols issued by the generator will be either $\{\tilde{c}^+, \tilde{c}^-\}$, $\{\tilde{c}^+\}$, or $\{\tilde{c}^-\}$.

It will be assumed that the supervisor is an identity mapping which simply passes on the plant symbol to the interface actuator. The actuator will then associate each symbol with a control vector $\mathbf{r}$ as follows

$$\mathbf{r} = \begin{cases} (1\ 0)^t & \text{if } \{\tilde{c}^+\} \\ (0\ 0)^t & \text{if } \{\tilde{c}^+, \tilde{c}^-\} \\ (0\ 1)^t & \text{if } \{\tilde{c}^-\} \end{cases} \tag{38}$$

These assumptions for the plant, actuator, generator, and supervisor yield a hybrid system which is, essentially, a variable structure control system. The dynamics of the plant under the supervisor's control are represented by the following set of switching differential equations

$$\dot{\mathbf{x}} = \begin{cases} f_1(\mathbf{x}) \text{ if } \mathbf{s}^t\mathbf{x} < -|\alpha| \\ \qquad\ \text{if } |\mathbf{s}^t\mathbf{x}| < |\alpha| \\ f_2(\mathbf{x}) \text{ if } \mathbf{s}^t\mathbf{x} > |\alpha| \end{cases} \tag{39}$$

One objective in variable structure control is to drive the plant state onto the hyperplane, $H_{\mathbf{s}}$, and keep it in the neighborhood of that surface. Define the surface $H_{\mathbf{s}}$ as

$$H_{\mathbf{s}} = \left\{ \mathbf{x} \in \Re^n : \mathbf{s}^t \mathbf{x} = 0 \right\} \tag{40}$$

This is a hyperplane passing through the origin, with normal vector $\mathbf{s}$. The neighborhood of this surface is represented by the set formed by intersecting the two events c^+ and c^-. Since the control objective is to drive the system state into $c^+ \cap c^-$ and keep it there, it is important that this set be an attracting invariant set with respect to the controlled plant's dynamics as shown in equation 39. The $H_{\mathbf{s}}$ which is invariant with respect tot he plant's dynamics will be referred to as a sliding mode.

An invariant subset with respect to a transformation group $\{\Phi_t\}$ is defined as follows.

Definition 5. The set $H \subset \Re^n$ will be a $\underline{\Phi\text{-invariant}}$ of the transformation group $\{\Phi_t : \Re^n \rightarrow \Re^n\}$ if and only if for any $\mathbf{x} \in H$, $\Phi_t(\mathbf{x}) \in H$ for all $t > 0$.

Of more interest are sets which are attracting invariants of the flow.

Definition 6. The set $H \subset \Re^n$ will be an $\underline{\text{attracting}}$ $\underline{\Phi\text{-invariant}}$ of the transformation group $\{\Phi_t : \Re^n \rightarrow \Re^n\}$ if and only if for any $\mathbf{x} \in H$, there exists a finite $T > 0$ such that $\Phi_t(\mathbf{x}) \in H$ for all $t > T$.

In our example, the transformation groups are the family of transition operators generated by the differential equations 39. These transformations, Φ_t, represent a collection of automorphisms over the state space which are sometimes called the "flow" of the dynamical system.

Unfortunately, not all choices of $\mathbf{s}$ will leave the target event invariant. Those hyperplanes which yield invariant target events can be determined directly from the set of vector fields $\{f_1, f_2\}$ representing the system's control policies. Examples of this computation can be found in nonlinear systems theory [Olver 1986]. However, this computation requires explicit equations for these control policies and there are numerous applications where such prior knowledge is unavailable. Uncertainty in the precise form of the control policies can arise from unpredicted variations in the plant's structure. Uncertainty can also arise in highly complex systems where the state space's high dimensionality precludes complete prior knowledge of the distributions. In such situations, it is necessary that the invariants be determined directly from the system's observed behaviour. Since the hybrid system's event covering is defined with respect to these invariants, we see that the problem of finding such invariants is essentially the problem of event identification. In other words, we need to identify a collection of covering events which are invariant with respect to the available control policies f_1 and f_2. This problem is referred to in this chapter as the invariant subspace identification (ISID) problem. The algorithms discussed in the following subsections provide one way of solving the ISID problem by direct (active) experimentation.

4.2 Invariant Subspace Identification Algorithm

Inductive inference is a machine learning protocol in which a system learns by example. It has found significant practical and theoretical uses in learning Boolean functions by example [Angluin 1983], proving poly-time complexity of linear programming [Khachiyan 1979] and combinatorial optimization [Groetschel 1988] algorithms, developing finite-time procedures for training linear classifiers [Rosenblatt 1962] [Ho-Kashyap 1965], and estimating sets bounding unknown parameters [Dasgupta 1987]. In this section, an inductive protocol for learning an $n-1$-dimensional invariant subspace of a variable structure system is formally stated.

The inductive protocol developed in this chapter can be seen as consisting of three fundamental components;

- an experiment for generating examples,
- a query to an algorithm called the membership oracle,
- and an update algorithm for modifying the system's current controller (i.e. switching surface).

These components are used to iteratively adjust the system's current estimate for the invariant subspace $H_{\mathbf{S}}$. Figure 8 illustrates the relationship between these three algorithm components.

[h]

Fig. 8. Flow chart for an inductive inference protocol solving the ISID problem

The algorithm begins by forming an initial hypothesis about the system's sliding mode. This hypothesis takes the form of an n-dimensional vector $\mathbf{s}$ and an n by n symmetric matrix, $\mathbf{Q}$. The vector represents a unit normal to a switching surface, $H_{\mathbf{s}}$, which is hypothesized to be a sliding mode. The matrix represents a convex cone which is known to contain those vectors normal to all sliding modes of the system. Because the matrix, $\mathbf{Q}$, is associated with a convex cone in $\Re^n$, it will have 1 negative eigenvalue and $n-1$ positive eigenvalues. For the purposes of this section it will therefore be convenient to make the following notational conventions. Let $\mathbf{e}_i$ be the ith eigenvector of $\mathbf{Q}$ and let λ_i be its associated eigenvalue. Assume that the eigenvalues and eigenvectors are ordered so that $\lambda_i > \lambda_{i+1}$ for all i. Define an n by $n-1$ matrix, $\mathbf{E}$, whose columns are the eigenvectors of $\mathbf{Q}$ with positive eigenvalues. This matrix will be called $\mathbf{Q}$'s positive eigenvector matrix. Also form an $n-1$ by $n-1$ diagonal matrix, $\mathbf{L}$, from the positive eigenvalues of $\mathbf{Q}$. This matrix will be called the positive eigenvalue matrix. Both matrices are shown below.

$$\mathbf{E} = \begin{pmatrix} \mathbf{e}_1 \; \mathbf{e}_2 \cdots \mathbf{e}_{n-1} \end{pmatrix}, \; \mathbf{L} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ . & . & \cdots & . \\ 0 & 0 & \cdots & \lambda_{n-1} \end{pmatrix}. \qquad (41)$$

The normalized eigenvalue matrix will be defined as $\mathbf{R} = \mathbf{L}/|\lambda_n|$.

After forming the initial hypothesis, the algorithm's first component, the experiment, is performed. This component involves the active measurement of the system's state and state velocity. The second algorithm component uses these experimental measurements to make a declaration on the validity of the hypothesis that the switching surface $H_{\mathbf{s}}$ is indeed a sliding mode. The declaration is made by a Boolean functional called the invariance oracle. The oracle's response is a MAYBE or FALSE declaration. If the answer is MAYBE then nothing is done. If the answer is FALSE, however, then the current hypothesis is modified using the algorithm's third component, the update algorithm.

The update algorithm uses a modification of the central-cut ellipsoid method [Shor 1977] to recompute the symmetric matrix $\mathbf{Q}$ and the vector $\mathbf{s}$. In modifying the hypothesis after the oracle's FALSE declaration, the update procedure attempts to generate a new hypothesis which is consistent with prior experimental data. This basic cycle of experiment, query, and update continues until an attracting invariant subspace is found.

The ISID algorithm can now be formally stated.

Invariant Subspace Identification (ISID) Algorithm

1. **Initialize:** Initialize an n by n symmetric matrix, $\mathbf{Q}$, which has $n-1$ positive eigenvalues and 1 negative eigenvalue such that if $H_{\mathbf{z}}$ is a sliding mode, then $\mathbf{z}^t Q \mathbf{z} < 0$. Compute the eigendecomposition of $\mathbf{Q}$.
2. **Form Hypothesis:** Set the system's current switching surface, $\mathbf{s}$, equal to the negative eigenvector, $\mathbf{e}_n$, of $\mathbf{Q}$.
3. **Experiment:** Measure the system's state and state velocity, $\mathbf{x}$ and $\dot{\mathbf{x}}$.
4. **Query:** Compute the invariance oracle's response,

$$I_1(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{s}) = \begin{cases} 0 & \text{if } (\mathbf{s}^t\mathbf{x})(\mathbf{s}^t\dot{\mathbf{x}}) < 0 \\ 1 & \text{otherwise} \end{cases} \qquad (42)$$

5. **Update Hypothesis:** If the oracle returns 1, then recompute $\mathbf{Q}$ using the following equations,

$$\mathbf{c} = \mathbf{E}^t\mathbf{x}, \tag{43}$$

$$\mathbf{b} = \frac{\mathbf{R}^{-1}\mathbf{c}}{\sqrt{\mathbf{c}^t\mathbf{R}^{-1}\mathbf{c}}}, \tag{44}$$

$$\mathbf{a} = -\frac{1}{n}\mathbf{b}, \tag{45}$$

$$\overline{\mathbf{R}}^{-1} = \frac{(n-1)^2}{(n-1)^2-1}\left(\mathbf{R}^{-1} - \frac{2}{n}\mathbf{b}\mathbf{b}^t\right), \tag{46}$$

$$\mathbf{x}_a = \mathbf{E}\mathbf{a} + \mathbf{e}_n, \tag{47}$$

$$\overline{\mathbf{Q}} = (\mathbf{I} - \mathbf{e}_n\mathbf{x}_a^t)\mathbf{E}\overline{\mathbf{R}}\mathbf{E}^t(\mathbf{I} - \mathbf{x}_a\mathbf{e}_n^t), \tag{48}$$

   Set $\mathbf{Q}$ equal to $\overline{\mathbf{Q}}$ and recompute the eigendecomposition of $\mathbf{Q}$.
6. If the oracle returns 0, then do nothing.
7. **Loop:** go to step 2.

4.3 Invariance Oracles

This subsection derives the oracle used by the ISID algorithm and formally stated in equation 42. The oracle is a Boolean functional which evaluates a sufficient condition for the set $H_{\mathbf{S}}$ to be attracting and invariant.

Consider a set X called the sample set and let M be a measurable subset of X. The membership oracle is defined as follows.

Definition 7. Given a sample set X and a measurable set $M \subset X$, the membership oracle for M is a mapping, $O : X \to \{0, 1\}$, such that for any $\mathbf{x} \in X$,

$$O(\mathbf{x}) = \begin{cases} 0 \text{ if and only if } \mathbf{x} \in M \\ 1 \text{ if and only if } \mathbf{x} \notin M \end{cases}. \tag{49}$$

The membership or M-oracle can be thought of as a decision machine determining whether or not an example is a member of set M. An example which is an element of M will be called a positive M-example. If M^c is the complement of M, then a positive M^c-example will sometimes be called a negative example. In this regard, the M-oracle's response can be interpreted as a TRUE or FALSE declaration concerning the membership of the example.

In certain cases, complete membership information may not be practical. It is therefore desirable to consider a weaker form of the M-oracle.

Definition 8. Given a sample set X and a measurable set $M \subset X$, the mapping, $O : X \to \{0, 1\}$, is called an incomplete M-oracle if there exists another measurable set N such that $M \subseteq N$ and the mapping, O, is an N-oracle.

The incomplete M-oracle is a weaker version of the M-oracle since it only declares that the example is not an element of M. It does not make any declaration about an example's membership in M. In this regard, an incomplete oracle's response can be interpreted as a FALSE or MAYBE declaration on the example's membership in M.

An invariance oracle will be a Boolean functional which declares whether or not a given subspace, $H_{\mathbf{s}}$, is attracting and Φ-invariant. Therefore the first step in defining an "invariance" oracle is to determine a test by which invariance can be determined. Sufficient conditions for attracting Φ-invariant sets form the basis of these tests. The following theorem provides a specific example of such a test.

Theorem 9. *Let* $\mathbf{s}$ *be a given n-dimensional real vector and let* $\dot{\mathbf{x}}$ *be given by equation 35. If the following condition*

$$\left(\mathbf{s}^t\mathbf{x}\right)\left(\mathbf{s}^t\dot{\mathbf{x}}\right) < 0, \tag{50}$$

is satisfied for all $\mathbf{x} \notin H_{\mathbf{s}}$, *then the subspace,* $H_{\mathbf{s}}$, *is an attracting* Φ*-invariant set.*

Proof: Define the functional $V(\mathbf{x}) = \frac{1}{2}\left(\mathbf{s}^t\mathbf{x}\right)^2$. Clearly, $V > 0$ for all $\mathbf{x}$. By the theorem's assumption, $\dot{V} < 0$, for all $\mathbf{x} \notin H_{\mathbf{s}}$. Therefore by theorem 8 in [Utkin 1977], $H_{\mathbf{s}}$ must be a sliding mode and is therefore an attracting Φ-invariant set of the flow. **QED**.

It should be apparent that equation 50 can be recast as a logical function making a declaration about the consistency of the measured state and state velocity with the hypothesis that $H_{\mathbf{s}}$ is a sliding mode. This, then motivates the following definition for an "invariance" oracle.

Definition 10. The Boolean functional, $I_1 : \Re^{3n} \rightarrow \{0, 1\}$, defined by equation 42 will be called an **invariance oracle**.

Let A denote a subset of $\Re^n$ consisting of those n-dimensional vectors $\mathbf{s}$ for which $H_{\mathbf{s}}$ is attracting and Φ-invariant. This set, A, will be referred to as the set of attracting and invariant subspaces. The following theorem states that the invariance oracle, I_1, is an incomplete A^c-oracle where A^c is the complement of set A.

Theorem 11. *Let* A *be the set of attracting invariant subspaces. If the function* $I_1 : \Re^{3n} \rightarrow \{0, 1\}$ *is an invariance oracle, then it is an incomplete* A^c*-oracle.*

Proof: Let A_1 be a set of n-dimensional vectors $\mathbf{s}$ such that $I_1 = 0$ for any $\mathbf{x}$ and $\dot{\mathbf{x}}$ given by equation 35. By definition 7, I_1 must be an A_1^c-oracle. By theorem 1, any element of A_1 must also be an attracting Φ-invariant set. Therefore $A_1 \subset A$, which implies $A^c \subset A_1^c$. This therefore establishes I_1 as an incomplete A^c-oracle according to definition 8. **QED**

The set A_1 defined in the above proof will be referred to as the set of attracting invariant subspaces which are **declarable** by the invariance oracle, I_1. Note that this set is smaller than A, the set of all attracting invariant subspaces. For this reason, the oracle is incomplete and its response is a declaration of TRUE or MAYBE concerning the membership of $\mathbf{s}$ in A.

In the remainder of this subsection, the data collection gathered by an experiment will be denoted as $\mathcal{X} = \{\mathbf{x}, \dot{\mathbf{x}}\}$. It is assumed, that these measurements have no measurement noise, so that the oracle's declarations are always correct. An invariance oracle which always makes the correct declaration for a given data collection, $\mathcal{X}$, will be called a **perfect** oracle. In practical oracle realizations, the assumption that the invariance oracle is perfect may be too optimistic due to measurement uncertainty. This realization prompts the definition of an **imperfect** oracle as an oracle whose declarations are incorrect with a given probability. The distinction between perfect and imperfect oracles is critical, because inductive protocols based on oracle queries can fail disasterously with imperfect oracles. The convergence results of subsection 4.5 only apply to perfect invariance oracles. Precisely how to manage failures due to imperfect oracles is an important issue for future study. A preliminary indication of how to handle this problem will be discussed in subsection 4.7.

4.4 Ellipsoidal Update Method

The ISID algorithm uses an update procedure which recursively adjusts an estimate for the set A_1 of attracting invariant subspaces declarable by I_1. The proposed updating procedure is therefore a set-estimation algorithm which is closely related to set-membership identification algorithms [Dasgupta 1987] [Deller 1989]. It is also related to analytical techniques used in proving polynomial oracle-time complexity for certain optimization algorithms [Groetschel 1988] [Khachiyan 1979]. The common thread between both of these related areas is the use of the ellipsoid method [Shor 1977] [Bland 1981], which the following discussion also uses to great advantage.

An important property of A_1 (the set of declarable subspaces) is provided in the following lemma.

Lemma 12. *A_1 is a convex cone centered at the origin.*

Proof: Consider a specific collection of measurements as $\mathcal{X} = \{\mathbf{x}, \dot{\mathbf{x}}\}$. Define $C_{\mathcal{X}}$ as

$$C_{\mathcal{X}} = \{\mathbf{s} \in \Re^n : I_1(\mathcal{X}, \mathbf{s}) = 0\}. \tag{51}$$

A_1 will therefore be given by

$$A_1 = \bigcap_{\mathcal{X}} C_{\mathcal{X}}. \tag{52}$$

Since the oracle's response for a given $\mathbf{s}$ is independent of the vector's magnitude, $C_{\mathcal{X}}$ must be a cone centered at the origin. Since $C_{\mathcal{X}}$ is formed by the intersection of two halfspaces (see inequality 50), it must also be convex. A_1 is therefore the intersection of a collection of convex cones centered at the origin and must therefore be one itself. **QED**

The significance of the preceding lemma is that it suggests A_1 may be well approximated by sets which are themselves convex cones centered at the origin. A particularly convenient selection of approximating cones are the so-called ellipsoidal cones.

An "ellipsoidal cone" cone is defined as follows,

Definition 13. The ellipsoidal cone, $C_e(\mathbf{Q})$, is

$$C_e(\mathbf{Q}) = \{\mathbf{s} \in \Re^n : \mathbf{s}^t\mathbf{Q}\mathbf{s} < 0\}, \tag{53}$$

where $\mathbf{Q}$ is an n by n symmetric matrix with $n-1$ positive eigenvalues and one negative eigenvalue.

In the update procedure to be derived below, an ellipsoidal cone, $C_e(\mathbf{Q})$, will be used as an initial estimate for A_1. The current hypothesis is that the subspace normal to the negative eigenvector of $\mathbf{Q}$ is an attracting invariant set. If any data collection, $\mathcal{X}$, results in the oracle, I_1, declaring a failure, then the information from that failed query can be used to identify a set of subspaces which cannot possibly lie in A_1. This set will be referred to as the "inconsistent" set of subspaces generated by $\mathcal{X}$. The following lemma provides one characterization of these sets.

Lemma 14. *Let $C_e(\mathbf{Q})$ be an ellipsoidal cone with negative eigenvector, $\mathbf{e}_n$. Let $\mathcal{X}$ be a data collection for which a perfect invariance oracle, I_1, declares a failure, $I_1(\mathcal{X}, \mathbf{e}_n) = 1$. If $A_1 \subset C_e(\mathbf{Q})$, then $A_1 \subset C_e(\mathbf{Q}) \cap H(\mathcal{X}, \mathbf{e}_n)$ where*

$$H(\mathcal{X}, \mathbf{e}_n) = \left\{ \mathbf{s} \in \Re^n : \mathbf{s}^t \mathbf{x} < \mathbf{e}_n^t \mathbf{x} \right\}. \tag{54}$$

The set $H(\mathcal{X}, \mathbf{e}_n)$ will be called the <u>inconsistent</u> <u>set</u> generated by $\mathcal{X}$.

Proof: If a perfect invariance oracle I_1 returns 1 for $\mathcal{X}$ given the subspace represented by $\mathbf{e}_n$, then the following inequality holds.

$$\left(\mathbf{e}_n^t \dot{\mathbf{x}}\right) \left(\mathbf{e}_n^t \mathbf{x}\right) > 0. \tag{55}$$

On the basis of this inequality, it is apparent that any subspace, $H_{\mathbf{z}}$, such that

$$\mathbf{z}^t \mathbf{x} \geq \mathbf{e}_n^t \mathbf{x}, \tag{56}$$

cannot possibly be an attracting invariant set. The collection of such subspaces form the complement of the halfspace $H(\mathcal{X}, \mathbf{e}_n)$ defined in the theorem. Since A_1 is assumed to lie in $C_e(\mathbf{Q})$, it must therefore lie in the intersection of $C_e(\mathbf{Q})$ with $H(\mathcal{X}, \mathbf{e}_n)$. **QED**

The significance of the preceding lemma is that the inconsistent set is an n-dimensional halfspace in $\Re^n$. To discuss this more fully, we first need to introduce the linear varieties of $n-1$-dimensional subspaces.

Definition 15. Let S be an $n-1$-dimensional subspace of $\Re^n$ and let $\mathbf{x}$ be an n-dimensional real vector. The <u>linear</u> <u>variety</u> of S generated by $\mathbf{x}$ is the set

$$V(S, \mathbf{x}) = \{\mathbf{s} + \mathbf{x} : \mathbf{s} \in S\}. \tag{57}$$

The following lemma shows that the inconsistent set forms a halfspace in the linear variety, $V(\mathrm{sp}(\mathbf{E}), \mathbf{e}_n)$.

Lemma 16. *Let $C_e(\mathbf{Q})$ be an ellipsoidal cone with negative eigevector, $\mathbf{e}_n$. Let $V(\mathrm{sp}(\mathbf{E}), \mathbf{e}_n)$ be a linear variety of the subspace spanned by the positive eigenvectors of $\mathbf{Q}$. If the inconsistent set $H(\mathcal{X}, \mathbf{e}_n)$ is as defined in lemma 14, then the set $H(\mathcal{X}, \mathbf{e}_n) \cap V(\mathrm{sp}(\mathbf{E}), \mathbf{e}_n)$ is an $n-1$ dimensional halfspace.*

[h]

Fig. 9. The set of subspaces, A_M, declarable by a perfect invariance oracle, I_M, forms a convex cone centered at the origin. The intersection of A_M with a linear variety of an $n-1$-dimensional subspace will be a bounded $n-1$-dimensional convex body, K.

Proof: This result is very straightforward and its formal proof will be found in [Lemmon 1993a].

The geometry implied by the preceding lemmas is illustrated in figure 9. The characterization of the ellipsoidal cone and inconsistent sets provided by these lemmas forms the basis for the following theorem. This theorem states the equations used in obtaining a bounding ellipsoidal cone for A_1 from a prior bounding cone and the inconsistent set generated by $\mathcal{X}$. The proof of this theorem is a straightforward application of the central-cut ellipsoid method [Shor 1977].

Theorem 17. *Let $C_e(\mathbf{Q})$ be an ellipsoidal cone with negative eigenvector $\mathbf{e}_n$ such that $A_1 \subset C_e(\mathbf{Q})$. Let $\mathcal{X}$ be a data collection for which $I_1(\mathcal{X}, \mathbf{e}_n) = 1$. There exist ellipsoidal cones, $C_e(\underline{\mathbf{Q}})$ and $C_e(\overline{\mathbf{Q}})$, such that*

$$C_e(\underline{\mathbf{Q}}) \subset H(\mathcal{X}, \mathbf{e}_n) \cap C_e(\mathbf{Q}) \subset C_e(\overline{\mathbf{Q}}). \tag{58}$$

Furthermore if $\mathbf{R} = \mathbf{L}/|\lambda_n|$ where $\mathbf{L}$ is the positive eigenvalue matrix and if $\mathbf{E}$ is the positive eigenvector matrix of $\mathbf{Q}$, then $\overline{\mathbf{Q}}$ is given by equations 43 through 48 and $\underline{\mathbf{Q}}$ is given by equations 43 through 48 where $\underline{\mathbf{R}}^{-1} = \overline{\mathbf{R}}^{-1}/(n-1)^2$ is used in place of $\overline{\mathbf{R}}^{-1}$ in equation 43.

Proof: From lemma 2, the intersection of cone $C_e(\mathbf{Q})$ with $V(\mathrm{sp}(\mathbf{E}), \mathbf{e}_n)$ is an ellipsoid of the following form

$$E(\mathbf{R}^{-1}, 0) = \{\mathbf{w} \in \Re^{n-1} : \mathbf{w}^t \mathbf{R} \mathbf{w} < 1\}. \tag{59}$$

From lemma 4, the intersection of the inconsistent set, $H(\mathcal{X}, \mathbf{e}_n)$ and $V(\mathrm{sp}(\mathbf{E}), \mathbf{e}_n)$ will be an $n-1$-dimensional halfspace, H, given by

$$H = \{\mathbf{w} \in \Re^{n-1} : \mathbf{w}^t \mathbf{c} < 0\}, \tag{60}$$

where $\mathbf{c} = \mathbf{E}^t \mathbf{x}$. Therefore the intersection of $C_e(\mathbf{Q})$, $V(\mathrm{sp}(\mathbf{E}), \mathbf{e}_n)$, and $H(\mathcal{X}, \mathbf{e}_n)$ will be an $n-1$-dimensional convex body, K.

It is well known that any bounded convex body can be contained within a unique ellipsoid of minimal volume called the Lowner-John ellipsoid [John 1984]. For convex bodies formed by single cuts of an ellipse, however, the Lowner-John ellipse can be computed in closed form [Groetschel 1988] [Bland 1981]. In particular, let K be the convex body formed by the intersection of an ellipse

$$E(\mathbf{A}, \mathbf{a}) = \left\{\mathbf{x} \in \Re^n : (\mathbf{x} - \mathbf{a})^t \, \mathbf{A}^{-1} (\mathbf{x} - \mathbf{a}) \leq 1\right\}. \tag{61}$$

with a halfspace, $\{\mathbf{x} : \mathbf{c}^t \mathbf{x} < \mathbf{c}^t \mathbf{a}\}$, then the Lowner-John ellipse, $E(\overline{\mathbf{A}}, \overline{\mathbf{a}})$ is given by

$$\overline{\mathbf{a}} = \mathbf{a} - \frac{n^2}{n+1}\mathbf{b}, \tag{62}$$

$$\overline{\mathbf{A}} = \frac{n^2}{n^2-1}\left(\mathbf{A} - \frac{2}{n+1}\mathbf{b}\mathbf{b}^t\right), \tag{63}$$

$$\mathbf{b} = \frac{\mathbf{A}\mathbf{c}}{\sqrt{\mathbf{c}^t \mathbf{A} \mathbf{c}}}. \tag{64}$$

Computing the Lowner-John ellipsoid for $K = E(\mathbf{R}^{-1}, 0) \cap H$ will yield the ellipsoid $E(\overline{\mathbf{R}}^{-1}, \mathbf{a})$ where $\overline{\mathbf{R}}$ and $\mathbf{a}$ are as given in the theorem. Figure 10 illustrates the geometry implied by the central-cut ellipsoid method.

The $n-1$-dimensional Lowner-John ellipsoid generates an n-dimensional ellipsoidal cone. Let $\mathbf{s}$ be any point in the cone generated by the ellipsoid $E(\overline{\mathbf{R}}^{-1}, \mathbf{a})$. There exists an $\alpha \in \Re$ such that $\alpha\mathbf{s}$ is in the linear variety, $V(\mathrm{sp}(\mathbf{E}), \mathbf{e}_n)$. The α for which this is true must satisfy the orthogonality condition,

$$0 = \mathbf{e}_n^t (\alpha\mathbf{s} - \mathbf{e}_n) \tag{65}$$

$$= \alpha\mathbf{e}_n^t \mathbf{s} - 1, \tag{66}$$

which implies that $\alpha = 1/\mathbf{e}_n^t \mathbf{s}$.

Since, $\mathbf{s} = \mathbf{E}\mathbf{w} + \mathbf{e}_n$, the ellipsoid equation for $E(\overline{\mathbf{R}}^{-1}, \mathbf{a})$ is

$$1 > (\mathbf{w} - \mathbf{a})^t \overline{\mathbf{R}} (\mathbf{w} - \mathbf{a}) \tag{67}$$

$$> (\mathbf{s} - \mathbf{x}_a)^t \mathbf{E}^t \overline{\mathbf{R}} \mathbf{E} (\mathbf{s} - \mathbf{x}_a), \tag{68}$$

where $\mathbf{x}_a = \mathbf{E}\mathbf{a} + \mathbf{e}_n$. The vector $\mathbf{s}$ in this equation must, of course, lie in the linear variety generated by $\mathbf{e}_n$, $V(\mathrm{sp}(\mathbf{E}), \mathbf{e}_n)$. From our preceding discussion, any vector in

[h]

Fig. 10. Lowner-John ellipsoid for convex body formed by a central cut ellipsoid.

the cone can be pulled back to the variety by appropriate renormalization with α. This then implies that if $\mathbf{s}$ is any vector in the cone, then

$$\left(\frac{\mathbf{s}}{\mathbf{s^t e_n}} - \mathbf{x}_a\right)^t \mathbf{E}^t\overline{\mathbf{R}}\mathbf{E}\left(\frac{\mathbf{s}}{\mathbf{e_n^t s}} - \mathbf{x}_a\right) < 1. \tag{69}$$

Multiplying through by $|\mathbf{s}^t\mathbf{e}_n|^2$, we obtain

$$\mathbf{s}^t\left[\left(\mathbf{I} - \mathbf{e}_n\mathbf{x}_a^t\right)\mathbf{E}^t\overline{\mathbf{R}}\mathbf{E}\left(\mathbf{I} - \mathbf{x}_a\mathbf{e}_n^t\right) - \mathbf{e}_n\mathbf{e}_n^t\right]\mathbf{s} < 0. \tag{70}$$

This inequality determines an ellipsoidal cone and the term within the square brackets is $\overline{\mathbf{Q}}$.

$\underline{\mathbf{Q}}$ is obtained by noting that if $E(\overline{\mathbf{R}}^{-1}, \mathbf{a})$ is a Lowner-John ellipsoid for K, then $E(\overline{\mathbf{R}}^{-1}/(n-1)^2, \mathbf{a})$ is an ellipsoid contained within K. By repeating the preceding construction with this smaller ellipsoid, the equation for $\underline{\mathbf{Q}}$ is obtained. **QED**

4.5 Convergence and Complexity

This subsection shows that the ISID algorithm generates a sequence of ellipsoidal cones whose negative eigenvectors must eventually lie in A_1. In particular, it is shown that if A_1 is non-empty then the ISID algorithm must converge after a finite number of MAYBE (1) declarations by the invariance oracle. It is further shown that under certain conditions the convergence time scales with the square of the state space dimension. The subsection therefore proves that the ISID algorithm has finite oracle-time convergence and polynomial oracle-time complexity where oracle-time is measured by the number of MAYBE declarations by the invariance oracle.

To prove the convergence of the ISID algorithm requires that there be some measure of the ellipsoidal cone's size or "volume". The set function used to define this volume is given below.

Definition 18. Let $C_e(\mathbf{Q})$ be an ellipsoidal cone and let the eigenvalues of $\mathbf{Q}$ be ordered as $\lambda_i > \lambda_{i+1}$. The **volume** of cone $C_e(\mathbf{Q})$ is defined to be

$$\text{vol}C_e(\mathbf{Q}) = \sqrt{\prod_{i=1}^{n-1} \frac{|\lambda_n|}{\lambda_i}}. \tag{71}$$

The volume of an ellipsoid, $E(\mathbf{A}, \mathbf{a})$, will be proportional to the square root of the determinant of $\mathbf{A}$. Since the determinant of $\mathbf{A}$ is simply the product of its eigenvalues, it should be clear that the preceding definition is using the volume of the $n-1$-dimensional ellipsoid contained in the linear variety $V(\text{sp}(\mathbf{E}), \mathbf{e}_n)$ as the "volume" of the cone.

The following theorem shows that the ISID algorithm must locate an attracting invariant subspace after a finite number of failed queries to a perfect invariance oracle.

Theorem 19. *Initialize the ISID algorithm with an ellipsoidal cone whose volume is unity and which is known to contain A_1. Let ϵ denote the volume of the smallest ellipsoidal cone containing A_1. If n is the state space dimension, then the ISID algorithm will determine an attracting invariant subspace after no more than $2(n-1)\ln\epsilon^{-1}$ failed queries to a perfect invariance oracle.*

Proof: Consider the ellipsoidal cone $C_e(\mathbf{Q}_i)$ after the ith failed invariance test. Let $\mathbf{E}$ and $\mathbf{L}$ be the positive eigenvector and eigenvalue matrices of $\mathbf{Q}_i$, respectively. The volume of this ellipsoid will be given by

$$\text{vol}C_e(Q_i) = \sqrt{\frac{1}{\prod_{j=1}^{n-1} \lambda_j(\mathbf{R})}}, \tag{72}$$

where $\lambda_j(\mathbf{R})$ is the jth positive eigenvalue of $\mathbf{R}$ and $\mathbf{R} = \mathbf{L}/|\lambda_n|$. Consider the ellipsoidal cone obtained using equations 43 through 48 of subsection 4.2. The symmetric matrix characterizing this cone is $\overline{\mathbf{Q}} = \mathbf{X}^t\mathbf{Y}\mathbf{X}$ where

$$\mathbf{X} = \begin{pmatrix} \mathbf{E}^t(\mathbf{I} - \beta\mathbf{e}_a\mathbf{e}_n^t) \\ \mathbf{e}_n^t \end{pmatrix}, \tag{73}$$

$$\mathbf{Y} = \begin{pmatrix} \mathbf{R} & 0 \\ 0 & -1 \end{pmatrix}. \tag{74}$$

where $\beta = ||\mathbf{x_a}||$. Applying the orthogonal transformation,

$$\mathbf{P} = \left(\mathbf{E}\ \mathbf{e}_n \right), \tag{75}$$

to $\mathbf{X}$, yields

$$\mathbf{P}^t\mathbf{X}^t = \begin{pmatrix} \mathbf{I} & 0 \\ -\beta\mathbf{e}_a^t\mathbf{E} & 1 \end{pmatrix} \tag{76}$$

where $\beta = ||\mathbf{x}_a||$ and $\beta\mathbf{e}_a = \mathbf{x}_a$. Recall that $\mathbf{x}_a$ is the center of the updated ellipsoid in the linear variety $V(\mathrm{sp}(\mathbf{E}), \mathbf{e}_n)$. For convenience, let $\mathbf{v}^t = -\beta\mathbf{e}_a^t E$.

Since the eigenvalues of $\overline{\mathbf{Q}}$ are unchanged by an orthogonal transformation, the eigenvalues of $\mathbf{P}^t\mathbf{X}^t\mathbf{Y}\mathbf{X}\mathbf{P}$ can be used to compute the volume of $\overline{\mathbf{Q}}$. This transformed matrix has the form

$$\begin{aligned} \mathbf{P}^t\overline{\mathbf{Q}}\mathbf{P} &= \mathbf{P}^t\mathbf{X}^t\mathbf{Y}\mathbf{X}\mathbf{P} & (77) \\ &= \begin{pmatrix} \overline{\mathbf{R}} & \overline{\mathbf{R}}\mathbf{v} \\ \mathbf{v}^t\overline{\mathbf{R}} & \mathbf{v}^t\overline{\mathbf{R}}\mathbf{v} - 1 \end{pmatrix}. & (78) \end{aligned}$$

Note that $\overline{\mathbf{R}}$ is an $n-1$ by $n-1$ leading principal submatrix of $\mathbf{P}^t\overline{\mathbf{Q}}\mathbf{P}$, so the eigenvalues of the two matrices satisfy the following interlacing property [Golub 1983].

$$\lambda_n(\overline{\mathbf{Q}}) \leq \lambda_{n-1}(\overline{\mathbf{R}}) \leq \lambda_{n-1}(\overline{\mathbf{Q}}) \leq \cdots \leq \lambda_2(\overline{\mathbf{Q}}) \leq \lambda_1(\overline{\mathbf{R}}) \leq \lambda(\overline{\mathbf{Q}}). \tag{79}$$

Since it is known that $\lambda_n(\overline{\mathbf{Q}})$ is negative (by the definition of an ellipsoidal cone), it can be shown that

$$\lambda_n(\mathbf{P}^t\mathbf{X}^t\mathbf{Y}\mathbf{X}\mathbf{P}) \leq \sigma_n^2(\mathbf{P}^t\mathbf{X}^t)\lambda_n(\mathbf{Y}), \tag{80}$$

where $\sigma_n(\mathbf{P}^t\mathbf{X}^t)$ is the smallest singular value of $\mathbf{P}^t\mathbf{X}^t$ and $\lambda_n(\mathbf{Y})$ is the negative eigenvalue of $\mathbf{Y}$ [Golub 1983]. Note that this eigenvalue must be negative one (by construction of $\mathbf{Y}$). Also note that the singular value must satisfy the following inequality for any $\mathbf{x} \in \Re^n$,

$$\sigma_n^2(\mathbf{P}^t\mathbf{X}^t) \leq \frac{\mathbf{x}^t\mathbf{P}^t\mathbf{X}^t\mathbf{X}\mathbf{P}\mathbf{x}}{\mathbf{x}^t\mathbf{x}}. \tag{81}$$

In particular, if we let $\mathbf{x} = (0\cdots01)^t$, then the smallest singular value must be less than unity. It can therefore be concluded that $|\lambda_n(\overline{\mathbf{Q}})| < 1$.

With the preceding results, it can be concluded that

$$\begin{aligned} \mathrm{vol}C_e(\overline{\mathbf{Q}}) &= \sqrt{\prod_{j=1}^{n-1} \frac{\lambda_n(\overline{\mathbf{Q}})}{\lambda_j(\overline{\mathbf{Q}})}} & (82) \\ &\leq \sqrt{\prod_{j=1}^{n-1} \frac{1}{\lambda_j(\overline{\mathbf{R}})}} & (83) \\ &\leq e^{-\frac{1}{2(n-1)}}\mathrm{vol}C_e(\mathbf{Q}). & (84) \end{aligned}$$

Inequality 84 is a consequence of the bound on the absolute value of the negative eigenvalue as well as the interlacing property (Eq. 79). This inequality is simply the volume of an ellipsoid $E(\overline{\mathbf{R}}^{-1}, \mathbf{a})$. Recall, however, that $\overline{\mathbf{R}}$ is obtained from $\mathbf{R}$ using the central-cut ellipsoid method. The relationship between the volumes of these ellipsoids is given by the last line of the inequality. This last inequality [Groetschel 1988] is

$$\frac{\mathrm{vol}E(\overline{\mathbf{A}}, \overline{\mathbf{a}})}{\mathrm{vol}E(\mathbf{A}, \mathbf{a})} = e^{-1/2n}. \tag{85}$$

which bound the rate at which ellipsoid volumes decrease when the central-cut ellipsoid method is used.

Since the initial ellipsoidal cone's volume is unity, then the ellipsoidal cone's volume after the Lth failed query must be bounded as follows,

$$\mathrm{vol}C_e(\mathbf{Q}_L) \leq e^{-\frac{L}{2(n-1)}}. \tag{86}$$

However, $C_e(\mathbf{Q}_L)$ cannot be smaller than ϵ by assumption, therefore the number of failed queries, L, must satisfy

$$\epsilon \leq e^{-\frac{L}{2(n-1)}}. \tag{87}$$

Rearranging this inequality to extract L shows that the number of failed invariance queries can be no larger than the bound stated by the theorem. **QED**

The following corollary for the preceding theorem establishes the polynomial oracle-time complexity of the ISID algorithm.

Corollary 20. *Assume that A_1 is a set which is contained within an ellipsoidal cone characterized by a matrix, $\mathbf{Q}$, whose normalized positive eigenvalues satisfy the inequality*

$$\frac{|\lambda_n|}{\lambda_i} > \gamma \tag{88}$$

for $1 > \gamma > 0$ and $i = 1, \ldots, n-1$. Under the assumptions of theorem 19, the ISID algorithm will determine an attracting invariant subspace after no more than $2(n-1)^2 \ln(n-1) + (n-1)^2 \ln \gamma^{-1}$ MAYBE declarations by the invariance oracle.

Proof: Because of the constraints on $\mathbf{Q}$, the volume of the smallest bounding ellipsoid will be no greater than $\gamma^{(n-1)/2}(n-1)^{-n+1}$. Inserting this into the bound of theorem 19 yields the asserted result. **QED**

The significance of the preceding corollary is apparent when we consider how such restrictions on the eigenvalues of $\mathbf{Q}$ might arise. In particular, if the ISID algorithm is realized in finite precision arithmetic, then γ is proportional to the least significant bit of the realization. In this regard, the result shows that for finite precision implementations, there is an upper bound on the number of queries which the system can fail before exceeding the realization's precision. In particular, this result then shows that the bound scales with the square of the state space dimension, thereby establishing the polynomial oracle-time complexity of the algorithm.

4.6 Example: AUV Stabilization

This section discusses an application of the ISID algorithm to the stabilization of an autonomous underwater vehicle's (AUV) dive plane dynamics. This problem represents an example of the ISID algorithm's use as an adaptive variable structure control algorithm.

AUV dynamics are highly nonlinear and highly uncertain systems. Nonlinearities arise from hydrodynamic forces, uncompensated buoyancy effects, as well as cross-state dynamical coupling. Uncertainties arise due to environmental effects such as changing current and water conditions as well as poorly known mass and hydrodynamic properties. When an AUV retrieves a large object, for example, the drag and mass of this object may substantially modify the vehicle's buoyancy and drag coefficients. Such changes cannot be accurately modeled beforehand and can therefore have a disasterous effect on the success of the AUV's mission. In these situations, it would be highly desirable to develop an algorithm which can quickly and efficiently relearn the stabilizing controller for the system. The ISID algorithm represents one method for achieving this goal.

The following simulation results illustrate how the ISID algorithm can quickly stabilize an AUV's dive plane dynamics. The simplified equations of motion for vehicle (pitch) angle of attack, θ, in the dive plane as a function of velocity, v, may be written as

$$\ddot{\theta} = K_1\dot{\theta} + K_2\theta|\theta| + K_3\theta|v| + u_\theta, \tag{89}$$

$$\dot{v} = -v + K_4|\theta|v + u_v, \tag{90}$$

where K_1, K_2, K_3, and K_4 are hydrodynamic force coefficients. u_v and u_θ represent control forces applied in the velocity and angle of attack channels, respectively. These equations clearly show how nonlinearities enter the dynamics through the hydrodynamic cross coupling between θ and v. Uncertainty arises from the simple fact that the hydrodynamic coefficients may be poorly known. In general, these coefficients will be complex functions of vehicle geometry, speed, orientation, and water conditions. Consequently, they can never be completely characterized because there are too many degrees of freedom. Figure 11 illustrates the geometry implied by the equations of motion.

Figure 12 illustrates the behaviour of an AUV without active attitude control. The figure shows the 3-d state space trajectory for a vehicle with initial condition $\theta = 1$ and $v = 1$. The commanded state is $\theta = 0$ and $v = 2$. Without active attitude control, $u_\theta = 0$, and $u_v = -v + 2$. In this example, the system is hydrodynamically stable so that natural system damping can be used to eventually null the angle of attack. The figure shows that by using this control strategy, the vehicle exhibits large oscillations in θ and v before settling to the commanded state. For this particular system, the results therefore indicate that the angle of attack should be actively nulled to improve trajectory tracking.

Variable structure control (VSC) has emerged as a powerful technique for controlling AUV's with uncertain dynamics [Yoerger 1985]. In the following simulations, a hierarchical variable structure controller with boundary layer was designed. The

[hbpt]

Fig. 11. Autonomous Underwater Vehicle Diveplane Dynamics

controls, u_θ and u_v, have the following form

$$u_\theta = \begin{cases} 1 & \text{if } \mathbf{s}_\theta^t \mathbf{x} \leq -\epsilon \\ \frac{\mathbf{s}_\theta^t \mathbf{x}}{\epsilon} & \text{if } -\epsilon < \mathbf{s}_\theta^t \mathbf{x} < \epsilon \\ -1 & \text{if } \mathbf{s}_\theta^t \mathbf{x} > \epsilon \end{cases}, \tag{91}$$

$$u_v = \begin{cases} h(\mathbf{s}_\theta^t \mathbf{x}) & \text{if } \mathbf{s}_v^t \mathbf{x} \leq -\epsilon \\ \frac{\mathbf{s}_v^t \mathbf{x}}{\epsilon} & \text{if } -\epsilon < \mathbf{s}_v^t \mathbf{x} < \epsilon \\ -h(\mathbf{s}_\theta^t \mathbf{x}) & \text{if } \mathbf{s}_v^t \mathbf{x} > \epsilon \end{cases}, \tag{92}$$

where $\epsilon > 0$ denotes the width of the boundary layer and $\mathbf{x} = (\theta, \dot{\theta}, v)^t$ is the state vector. The function $h : \Re \rightarrow \{0, 1\}$ is assumed to have the following form

$$h(x) = \begin{cases} 0 & \text{if } |x| > \epsilon \\ 1 & \text{otherwise} \end{cases}, \tag{93}$$

and is used to implement a control hierarchy in which the system nulls angle of attack prior to nulling commanded velocity errors. The n-dimensional vectors $\mathbf{s}_\theta$ and $\mathbf{s}_v$ represent hyperplanes called switching surfaces just as was originally shown in equation 35 of the introduction.

The initial design of variable structure controllers can usually be partitioned into two phases. The first phase consists of determining the switching surfaces on which the system trajectories exhibit the desired transient response. The second phase determines the control strategies (gain levels) which insure that the switching surfaces are attracting invariant sets. Such switching surfaces are called sliding modes, since the system state is eventually captured by and slides along the switching surface. The need for adaptive identification of these surfaces arises when the system's structure changes in an unpredictable manner as when the vehicle retrieves a bulky package.

[h]

Fig. 12. Simulated AUV dive with no active nulling of angle of attack, θ. **A**: 3-d phase space trajectory, **B**: angle of attack, θ, time history, **C**: velocity, v, time history.

In order to preserve system autonomy, the two phase design procedure cannot be followed, since the system's control strategies were fixed at the system's initial design. Consequently, the only part of the controller which can be modified is the switching surface and this modification must be done adaptively on the basis of the system's observed behaviour.

The simulation results shown in figure 13, 14, and 15 illustrate precisely how the ISID algorithm can be used to "relearn" the system's sliding modes. Figure 13 shows the AUV's performance (same initial conditions as shown in figure 12) with the hierarchical sliding mode controller after a system failure causes the initially chosen switching surfaces to no longer be invariant sets. As can be seen, the sliding controller is actually unstable with the system exhibiting large oscillations in θ. Figures 14 and 15 show the system's behaviour during two "learning" sessions with the ISID algorithm. A learning session involves starting the vehicle at the initial condition and then commanding it over to the desired state. The first learning session is shown in figure 14. This particular example exhibited four adjustments to the sliding surface. On the last adjustment, the sliding condition is satisfied and the system slides easily to the commanded state. Figure 15 shows the system's response during the second training session. In this case, it is clear that learning is complete. There are no readjustments of the sliding surface and the system wastes little effort in bringing the system to its commanded state.

Perhaps the most remarkable thing about this example is the apparent speed with which the sliding surface is learned. In these simulations, only 4 failed invariance tests were required before finding a sliding mode. This low number of failed tests was observed in other simulation runs where the system's initial conditions were randomly varied. When compared with existing methods for learning nonlinear controllers [Narendra 1990] [Barto 1983] [Jacobs 1991], this approach appears to be exceptionally fast.

[h]

Fig. 13. Simulated AUV dive with hierarchical sliding control in which sliding mode constraints are violated. **A**: 3-d phase space trajectory, **B**: angle of attack, θ, time history, **C**: velocity, v, time history.

4.7 Significant Issues

The final theorem of subsection 4.5 is significant for two reasons. First it shows that the invariant subspaces can be located after a finite number of of failed queries. In sliding mode control, such subspaces are used to stabilize the system as was shown in the preceding example. Therefore, theorem 19 says that a system only needs to perceive itself as "unstable" a finite number of times before system stability is re-established. This result stands in stark contrast to other results [Barto 1983] [Narendra 1990] [Jacobs 1991] where system stability can only be iteratively "learned" after a prohibitively long training period. The second important aspect of the preceding results is that the theorem's bound implies that the algorithm has polynomial time complexity. This means that as systems become more and more complex (i.e. larger state spaces), the time required to learn the system invariants will grow at a modest rate. In other words, the proposed ISID algorithm may represent a practical method for adaptive control and identification of highly complex nonlinear dynamical systems.

[h]

Fig. 14. Simulated AUV dive where ISID algorithm is used to relearn hierarchical sliding mode controller (First Learning Session). **A**: 3-d phase space trajectory, **B**: angle of attack, θ, time history, **C**: velocity, v, time history.

[h]

Fig. 15. Simulated AUV dive where ISID algorithm is used to relearn hierarchical sliding mode controller (Second Learning Session). **A**: 3-d phase space trajectory, **B**: angle of attack, θ, time history, **C**: velocity, v, time history.

It should be noted, however, that these bounds are <u>not</u> with respect to system time, but rather with respect to failed oracle time. This is an important distinction for it is quite possible that there may be a long period of time between consecutive oracle declarations of failure. Consequently, convergence of the ISID algorithm can be extremely long in "system" time and may, in fact, never converge at all. At first glance, this observation may seem to cast doubt upon the value of theorem 19. Upon closer consideration, however, it provides further insight into the method. Recall that the oracle will always declare failures if the system trajectory is diverging from the current subspace, $H_{\mathbf{S}}$. In other words, if the system is exhibiting "unstable" behaviour, the switching surface is modified. For the times between failures, the system appears to be stable and there is, therefore, no reason to change the switching surfaces. From this viewpoint, it can be seen that the bound of theorem 19 is very meaningful since it is measured with respect to the only quantity of physical interest to the system; the number of times the system "stumbles".

This point should be contrasted to parameter and set-membership identification [Dasgupta 1987] [Deller 1989] algorithms. In these cases, the important measure of parameter convergence is system time (i.e., the total number of experiments), since we are interested in obtaining accurate estimates as quickly as possible. Obviously for the parameter identification problem, the bounds computed by the preceding theorem would be useless unless the time between consecutive failures could be bounded also. That is not the situation, however, in the ISID problem which is primarily an adaptive "control" problem. The fact that these oracle-time bounds are meaningful for the ISID problem is an important point distinguishing this application from other more traditional applications of inductive inference protocols.

Finally, it must be observed that the preceding theorem assumes a perfect invariance oracle. In practice, oracles will not be perfect and the question is then what can be done to minimize the problems generated by an imperfect oracle. The answer is also provided by the preceding theorem. Theorem 19 provides a hard bound on the number of failed oracle queries. If the system generates more failures than implied by the bound, then a failure must either have occured in the oracle or else in the system itself. In either case, the theorem's finite time bound provides a natural criterion for failure detection and the subsequent reinitialization of the identification process. If the rate of oracle failure is known to be small (i.e. failure probability is small), then the natural course of action is to reinitialize the ISID algorithm and try again. The preceding discussion therefore implies the existence of effective and practical methods for dealing with the identification failures caused by imperfect oracles. In particular, if we model an oracle's imperfection as a probabilistic failure rate, then it should be possible to discuss the ISID algorithm's learning abilities within the so-called "probably almost correct" (PAC) framework used by a variety of researchers in the inductive inference community [Valiant 1984]. A full study of techniques for optimally managing the failures introduced by an imperfect oracle is well beyond the scope of the current chapter and represents an important topic for further inquiry.

4.8 Symbol Grounding and Event Identification

Formal computational systems are often interfaced to the external world. Such "hybrid" systems are used to control or interpret events within that external world. The example discussed in the section is one example of such a hybrid system. Since the supervisor uses high-level abstractions to control the plant, such controllers are often referred to as "intelligent".

As noted in the section's introduction, this notion of intelligence is somewhat limited. If high-level decision making is to constitute intelligence, then this would imply that many symbol systems would be intelligent systems. This notion is, of course, at the heart of symbolic Artificial Intelligence research and it has its detractors. John Searle [Searle 1984] disputed the AI notion of intelligent machines with his now famous Chinese room argument. In this thought experiment it was noted that a prerequisite for "intelligence" is semantic content and that such content is unavailable to a purely symbolic system. For this reason, a computer can never be intelligent thereby debunking the traditional AI assumptions concerning the computational basis of human cognition.

At the heart of Searle's complaint is the notion of a symbol's meaning. This problem is also referred to as the symbol grounding problem [Harnad 1990]. Symbol grounding refers to methods by which symbols of a formal system acquire semantic content or "meaning". Such meaning generally has two interpretations. Meaning can be acquired through the association of symbols with nonsymbolic items in the external world. This acquired meaning is referred to as a symbol's extrinsic meaning. However, symbols also acquire content through internal associations with other symbols. This form of content might be referred to as intrinsic meaning.

An example of extrinsic meaning is seen in the association of the symbolic token "ELEPHANT" with those sensory inputs produced by seeing an elephant. The association of these sensory experiences with the symbolic token is determined by experience. The "meaning" of the symbol is determined by its nonsymbolic associations and is therefore external to the symbol system, itself. Consequently, we refer to such meaning as "extrinsic".

A symbol system, as Searle asserts, is not sufficient for an intelligent machine. Extrinsic meaning simply replaces the symbolic token with nonsymbolic tokens. The system has no real "understanding" of what those tokens signify so that the resulting computation is "meaningless". A good example of this type of "unintelligent" association is seen in superivsory control systems which make extensive use of DES models. In these cases, the "meaning" of the logical constructs is determined in an a priori way by the DES modeler. These intelligent choices for symbol/event bindings therefore imply that it is the modeler, rather than the system, which is intelligent.

In order for a system to be intelligent it must not only use high level abstractions, it must be able to generate them from internally generated construction principles. Symbols which arise in this manner may be grounded in nonsymbolic entities, but the meaning of these entities is determined internally, i.e. with respect to some intrinsic systems principle. In this regard, the symbols of such a system are intrinsically grounded. It is this form of "intrinsic" semantically meaning, which Searle asserted as a prerequisite for intelligence.

Clearly, conventional symbolic AI does not intrinsically ground its symbols. It has been argued that the more recent connectionist or subsymbolic AI concepts does embody some form of internal grounding[Chalmers 1992]. In view of these preceding remarks concerning symbol grounding and intelligence, it might now be appropriate to discuss the preceding ISID algorithm in light of the symbol grounding problem. Does the ISID algorithm produce event/symbol bindings which are "intrinsically" or "extrinsically" grounded. If the bindings are wholly external, then the resulting control system cannot be "intelligent" in the sense proposed by Searle.

In reviewing the modeling framework used in this paper, it is apparent that all plant events, $\tilde{x}$, are grounded with respect to a specific subset of the state space. At first glance, one might conclude then that this is an external grounding. However, the true test of external grounding is to see what happens if the event/symbol bindings change. In other words, if we shuffle the associations between symbols and nonsymbolic entities, does the operation of the supervisor change? If the ISID algorithm is not used, then clearly the bindings are unchanged. However, the ISID algorithm uses a computational algorithm (i.e. the invariance oracle) to decide whether or not the current event/symbol bindings satisfy or are consistent with the "internal" principle of control invariance. Therefore, if the initial event/symbol bindings change so that the resulting symbol groundings are inconsistent with the invariance oracle, then the supervisor changes the symbol bindings by redefining the "events". In other words, there is an internally coded principle guiding the symbol grounding process. Under this viewpoint, we can then assert that the ISID algorithm produces symbols with intrinsic semantic content.

The intrinsic content embodied by the invariance oracle is, of course, hardwired into the system. The choice of what this oracle is, represents a choice by the system designer. There can be other oracle structures used, in which different internal event principles are used. Therefore, in some sense, it is still through the ingenuity of the designer that this system appears to have "intelligent" processing. However, the fact remains that this system is endowing its symbols with a meaning which is derived by the system internally. This fact is still true regardless of where that "internally" coded principle came from. In this regard, we could consider the use of the ISID algorithm as resulting in an "intelligent" control system in the sense proposed by J. Searle.

These notions also provide some additional perspective on what is "intelligent" control. Intelligent control is often a vaguely defined concept referring to the use of high-level decision making processes in control. In the preceding section, it has been argued that this is not sufficient. Intelligence is not a behaviour, but a property of a system. For intelligence, a system must not only use symbolic abstractions, it must formulate its own symbol bindings with regard to specific internal principles. The ISID algorithm provides a way by which traditional hybrid systems might accomplish this intelligence through event identification.

5 Concluding Remarks

This chapter has introduced a model for hybrid systems which has focused on the role of the interface between the continuous-state plant and discrete-event supervisor. An especially simple form of the interface was introduced in which symbolic events and nonsymbolic state/control vectors are related to each other via memoryless transformations. It was seen that this particular choice dichotomizes the symbolic and nonsymbolic parts of the hybrid system into two cleanly separated dynamical systems which clearly expose the relationship between plant and supervisor. With the use of the proposed interface, quasi-determinism can be used to extend controllability concepts to hybrid systems. The clear separation of symbolic and nonsymbolic domains allows the formulation of hybrid controller methodologies which are directly based on equivalent DES control methods. Finally, the acknowledgement of the different roles played by symbolic and nonsymbolic processing in hybrid systems allows the proper formulation of the hybrid system's identification problem known as event identification. The solution of the identification problem sheds considerable insight into the meaning of intelligence in control.

The work outlined in the preceding sections is indicative of the breadth of work currently being pursued in the area of hybrid systems as a means of modeling and designing supervisory and intelligent control systems. In spite of the great strides being made in this area, there are significant issues which remain to be addressed in future work. These issues include a more rigorous examination of the traditional control concepts of controllability, observability, and stability with regard to hybrid systems. To some extent, the notions of quasi-determinism and the event identification problems are preliminary efforts to codify these extensions. Future work, however, remains before these extensions are fully understood. Another area of important future study lies in the formulation of control and identification algorithms with bounded computational complexity. Since the hybrid system explicitly contains a computational system, the importance of computational complexity as it scales with plant complexity can no longer be ignored. The ISID algorithm for the event identification problem indicates that such complexity issues can be addressed realistically using simple existing algorithmic approaches. Future work, clearly needs to be done in order to extend these ideas.

References

[Acar 1990] L. Acar, U. Ozguner, "Design of Knowledge-Rich Hierarchical Controllers for Large Functional Systems", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 20, No. 4, pp. 791-803, July/Aug. 1990.

[Angluin 1983] D. Angluin, C.H. Smith, "Inductive Inference: Theory and Methods." *Computing Surveys*, 15(3):237-269, September 1983.

[Albus 1981] J. Albus, et al, "Theory and Practice of Intelligent Control", *Proc. 23rd IEEE COMPCON*, pp 19-39, 1981.

[Antsaklis 1989] P. J. Antsaklis, K. M. Passino S. J. and Wang, "Towards Intelligent Autonomous Control Systems: Architecture and Fundamental Issues", *Journal of Intelligent and Robotic Systems*, Vol.1, pp. 315-342, 1989.

[Antsaklis 1990, 1992] P. J. Antsaklis, Special Issues on 'Neural Networks in Control Systems' of the IEEE Control Systems Magazine, April 1990 and April 1992.

[Antsaklis 1991] P. J. Antsaklis, K. M. Passino and S. J. Wang, "An Introduction to Autonomous Control Systems", *IEEE Control Systems Magazine*, Vol. 11, No. 4, pp.5-13, June 1991.

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

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

[Antsaklis 1993c] P. J. Antsaklis, "Neural Networks for the Intelligent Control of High Autonomy Systems", *Mathematical Studies of Neural Networks*, J.G. Taylor, Ed., Elsevier, 1993. To appear.

[Astrom 1986] K. J. Astrom, et al, "Expert Control", *Automatica*, Vol. 22, No. 3, pp. 277-286, 1986.

[Barto 1983] A.G. Barto, R. S. Sutton, and C. W. Anderson), "Neuronlike Elements that can Solve Difficult Learning Control Problems", *IEEE Trans. on Systems, Man, and Cybernetics*, Vol 13:835-846.

[Benveniste 1990] A. Benveniste, P. Le Guernic, "Hybrid Dynamical Systems and the SIGNAL Language", *IEEE Transactions on Automatic Control*, Vol. 35, No. 5, pp. 535-546, May 1990.

[Bland 1981] R.G. Bland, D. Goldfarb, M.J. Todd, "The Ellipsoid Method: a Survey", *Operations Research*, 29:1039-1091, 1981.

[Cassandras 1990] C. Cassandras, P. Ramadge, "Toward a Control Theory for Discrete Event Systems", *IEEE Control Systems Magazine*, pp. 66-68, June 1990.

[Chalmers 1992] D. J. Chalmers, "Subsymbolic Computation and the Chinese Room", in *The Symbolic and Connectionist Paradigms: closing the gap*, ed: John Dinsmore, Lawrence Erlbaum Associates, pp. 25-48, 1992.

[Dasgupta 1987] Dasgupta and Huang, "Asymptotically Convergent Modified Recrusive Least-Squares with Data-Dependent Updating and Forgetting Factor for Systems with Bounded Noise", *IEEE Trans. on Information Theory*, IT-33(3):383-392.

[DeCarlo 1988] R. A. DeCarlo, S. H. Zak, and GP Matthews (1988), "Variable Structure Control of Nonlinear Multivariable Systems: a Tutorial", *Proceedings of the IEEE*, Vol. 76(3):212-232.

[Deller 1989] J. R. Deller, "Set Membership Identification in Digital Signal Processing", *IEEE ASSP Magazine*, Vol. 6:4-20, 1989.

[Fishwick 1991] P. Fishwick, B. Zeigler, "Creating Qualitative and Combined Models with Discrete Events", *Proceedings of The 2nd Annual Conference on AI, Simulation and Planning in High Autonomy Systems*, pp. 306-315, Cocoa Beach, FL, April 1991.

[Fukunaga 1990] K. Fukunaga), *Introduction to Statistical Pattern Recognition*, 2nd edition, Academic Press, Boston.

[Gollu 1989] A. Gollu, P. Varaiya, "Hybrid Dynamical Systems", *Proceedings of the 28th Conference on Decision and Control*, pp. 2708-2712, Tampa, FL, December 1989.

[Golub 1983] G. Golub, C. Van Loan, *Matrix Computation*, Johns Hopkins University Press, Baltimore, Maryland, 1983

[Groetschel 1988] Groetshel, Lovasz, and Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, 1988.

[Grossman 1992] R. Grossman, R. Larson, "Viewing Hybrid Systems as Products of Control Systems and Automata", *Proceedings of the 31st Conference on Decision and Control*, pp. 2953-2955, Tucson AZ, December 1992.

[Harnad 1990] S. Harnad, "The Symbol Grounding Problem", *Physica D*, vol. **42**, pp. 335-446, 1990.

[Ho-Kashyap 1965] Y. C. Ho and R. L. Kashyap (1965), "An Algorithm for Linear Inequalities and its Applications", *IEEE Trans. Electronic Computers*, EC-14:683-688.

[Holloway 1992] L. Holloway, B. Krogh, "Properties of Behavioral Models for a Class of Hybrid Dynamical Systems", *Proceedings of the 31st Conference on Decision and Control*, pp. 3752-3757, Tucson AZ, December 1992.

[IEEE Computer 1989] Special Issue on Autonomous Intelligent Machines, IEEE Computer, Vol. 22, No. 6, June
1989.

[Isidori 1989] A. Isidori, *Nonlinear Control Systems*, 2nd Edition, Springer-Verlag, Berlin, 1989

[Jacobs 1991] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive Mixtures of Local Experts", *Neural Computation*, Vol 3(1):79-87.

[John 1984] John, *Fritz John: Collected Papers (1948)*, Birkhauser, 1984.

[Khachiyan 1979] L. G. Khachiyan, "A Polynomial Algorithm in Learn Program", (english translation), *Soviet Mathematics Doklady*, 20:191-194, 1979.

[Kohn 1992] W. Kohn, A. Nerode, "Multiple Agent Autonomous Hybrid Control Systems", *Proceedings of the 31st Conference on Decision and Control*, pp. 2956-2966, Tucson AZ, December 1992.

[Lemmon 1992] M. D. Lemmon, "Ellipsoidal Methods for the Estimation of Sliding Mode Domains of Variable Structure Systems", *Proc. of 26th Annual Conference on Information Sciences and Systems*, Princeton N.J., March 18-20, 1992.

[Lemmon 1993a] M. D. Lemmon, "Inductive Inference of Invariant Subspaces", *Proceedings of the American Control Conference*, San Francisco, California, June 2-4, 1993.

[Lemmon 1993b] M. D. Lemmon, J. A. Stiver, P. J. Antsaklis, "Learning to Coordinate Control Policies of Hybrid Systems", *Proceedings of the American Control Conference*, San Francisco, California, June 2-4, 1993.

[Mendel 1968] J. Mendel and J. Zapalac, "The Application of Techniques of Artificial Intelligence to Control System Design", in *Advances in Control Systems*, C.T. Leondes, ed., Academic Press, NY, 1968.

[Mesarovic 1970] M. Mesarovic, D. Macko and Y. Takahara, *Theory of Hierarchical, Multilevel, Systems*, Academic Press, 1970.

[Narendra 1990] K. S. Narendra and K. Parthsarathy, "Identification and Control of Dynamical Systems Using Neural Networks", *IEEE Trans. Neural Networks*, Vol 1(1):4-27.

[Nerode 1992] A. Nerode, W. Kohn, "Models for Hybrid Systems: Automata, Topologies, Stability", Private Communication, November 1992.

[Olver 1986] P. J. Olver (1986), *Applications of Lie Groups to Differential Equations*, Springer-Verlag, New York, 1986

[Ozveren 1991] C. M. Ozveren, A. S. Willsky and P. J. Antsaklis, "Stability and Stabilizabilty of Discrete Event Dynamic Systems", *Journal of the Association of Computing Machinery*, Vol 38, No 3, pp 730-752, 1991.

[Passino 1989a] K. Passino, "Analysis and Synthesis of Discrete Event Regulator Systems", Ph. D. Dissertation, Dept. of Electrical and Computer Engineering, Univ. of Notre Dame, Notre Dame, IN, April 1989.

[Passino 1989b] 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.

[Passino 1991a] K. M. Passino, A. N. Michel, P. J. Antsaklis, "Lyapunov Stability of a Class of Discrete Event Systems", *Proceedings of the American Control Conference*, Boston MA, June 1991.

[Passino 1991b] K. M. Passino, U. Ozguner, 'Modeling and Analysis of Hybrid Systems: Examples", *Proc. of the 1991 IEEE Int. Symp. on Intelligent Control*, pp. 251-256, Arlington, VA, Aug. 1991.

[Passino 1992a] K. M. Passino, A. N. Michel and P. J. Antsaklis, "Ustojchivost' po Ljapunovu klassa sistem diskretnyx sobytij", *Avtomatika i Telemekhanika*, No.8, pp. 3-18, 1992. "Lyapunov Stability of a Class of Discrete Event Systems", *Journal of Automation and Remote Control*, No.8, pp. 3-18, 1992. In Russian.

[Passino 1992b] 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.

[Passino 1993] K. M. Passino and P. J. Antsaklis, ""Modeling and Analysis of Artificially Intelligent Planning Systems", *Introduction to Intelligent and Autonomous Control*, P.J.Antsaklis and K.M.Passino, Eds., Chapter 8, pp. 191-214, Kluwer, 1993.

[Peleties 1988] P. Peleties, R. DeCarlo, "Modeling of Interacting Continuous Time and Discrete Event Systems : An Example", *Proceedings of the 26th Annual Allerton Conference on Communication, Control, and Computing*, pp. 1150-1159, Univ. of Illinois at Urbana-Champaign, October 1988.

[Peleties 1989] P. Peleties, R. DeCarlo, "A Modeling Strategy with Event Structures for Hybrid Systems", *Proceedings of the 28th Conference on Decision and Control*, pp.1308-1313, Tampa FL, December 1989.

[Peterson 1981] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[Ramadge 1987] P. Ramadge, W. M. Wonham, "Supervisory Control of a Class of Discrete Event Processes", *SIAM Journal of Control and Optimization*, vol. 25, no. 1, pp. 206-230, Jan 1987.

[Ramadge 1989] P. Ramadge, W. M. Wonham, "The Control of Discrete Event Systems", *Proceedings of the IEEE*, Vol. 77, No. 1, pp. 81 - 98, January 1989.

[Rosenblatt 1962] F. Rosenblatt), *Principles of Neurodynamics*, Spartan books, Washington D.C.

[Saridis 1979] G. N. Saridis, "Toward the Realization of Intelligent Controls", *Proc. of the IEEE*, Vol. 67, No. 8, pp. 1115-1133, August 1979.

[Saridis 1983] G. N. Saridis, "Intelligent Robot Control", *IEEE Trans. on Automatic Control*, Vol. AC-28, No. 5, pp. 547-556, May 1983.

[Saridis 1985] G. N. Saridis, "Foundations of the Theory of Intelligent Controls", *Proc. IEEE Workshop on Intelligent Control*, pp 23-28, 1985.

[Saridis 1987] G. N. Saridis, "Knowledge Implementation: Structures of Intelligent Control Systems", *Proc. IEEE International Symposium on Intelligent Control*, pp. 9-17, 1987.

[Saridis 1989a] G. N. Saridis, "Analytic Formulation of the Principle of Increasing Precision with Decreasing Intelligence for Intelligent Machines", *Automatica*, Vol.25, No.3, pp. 461-467, 1989.

[Searle 1984] J. R. Searle, *Minds, brains, and science*, Cambridge, MA: Harvard University Press, 1984.

[Shor 1977] N. Z. Shor, "Cut-off Method with Space Extension in Convex Programming Problems", (english translation), *Cybernetics*, 13:94-96, 1977.

[Slotine 1988] Slotine and Li, *IEEE Trans. on Automatic Control*, Vol. AC-33:995-1003

[Stengel 1984] R. F. Stengel, "AI Theory and Reconfigurable Flight Control Systems", Princeton University Report 1664-MAE, June 1984.

[Stiver 1991a] J. A. Stiver, "Modeling of Hybrid Control Systems Using Discrete Event System Models", M.S. Thesis, Dept. of Electrical Engineering, Univ. of Notre Dame, Notre Dame, IN, May 1991.

[Stiver 1991b] J. A. Stiver, P. J. Antsaklis, "A Novel Discrete Event System Approach to Modeling and Analysis of Hybrid Control Systems", *Control Systems Technical Report #71*, Dept. of Electrical Engineering, University of Notre Dame, Notre Dame, IN, June 1991.

[Stiver 1991c] J. A. Stiver, P. J. Antsaklis, "A Novel Discrete Event System Approach to Modeling and Analysis of Hybrid Control Systems", *Proceedings of the Twenty-Ninth Annual Allerton Conference on Communication, Control, and Computing*, University of Illinois at Urbana-Champaign, Oct. 2-4, 1991.

[Stiver 1992] J. A. Stiver, P. J. Antsaklis, "Modeling and Analysis of Hybrid Control Systems", *Proceedins of the 31st Conference on Decision and Control*, pp. 3748-3751, Tucson AZ, December 1992.

[Stiver 1993] J. A. Stiver, P. J. Antsaklis, "State Space Partitioning for Hybrid Control Systems", *Proceedings of the American Control Conference*, San Francisco, California, June 2-4, 1993.

[Turner 1984] P. R. Turner, et al, "Autonomous Systems: Architecture and Implementation", Jet Propulsion Laboratories, Report No. JPL D- 1656, August 1984.

[Utkin 1977] V.I. Utkin, "Variable Structure Systems with Sliding Modes", *IEEE Transactions on Automatic Control*, Vol. AC-22:212-222.

[Valavanis 1986] K. P. Valavanis, "A Mathematical Formulation For The Analytical Design of Intelligent Machines", PhD Dissertation, Electrical and Computer Engineering Dept., Rensselaer Polytechnic Institute, Troy NY, Nov. 1986.

[Valiant 1984] L. Valiant, "A Theory of the Learnable", *Comm. of ACM*, Vol 27(11):1134-1142

[Wonham 1987] W. M. Wonham, P. J. Wonham, "On the Supremal Controllable Sublanguage of a Given Language", *SIAM Journal of Control and Optimization*, vol. 25, no. 3, pp. 637-659, May 1987.

[Yoerger 1985] D. R. Yoerger and J. J. E. Slotine, "Robust Trajectory Control of Underwater Vehicles", *IEEE Journal of Oceanic Engineering*, Vol OE-10(4):462-470.

[Zeigler 1984] B. P. Zeigler, "Multifacetted Modelling and Discrete Event Simulation", Acedemic Press, NY, 1984.

[Zeigler 1987] B. P. Zeigler, 'Knowledge Representation from Newton to Minsky and Beyond", *Journal of Applied Artificial Intelligence*, 1:87-107, 1987.

[Zeigler 1989] B. P. Zeigler, "DEVS Representation of Dynamical Systems: Event Based Intelligent Control", *Proc. of the IEEE*, Vol. 77, No. 1, pp. 72-80, 1989.