P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

Supervisory Hybrid Control Systems in Intelligent Control

Panos Antsaklis, Jim Stiver,
Xiaojun Yang, Chris Bett, and Mike Lemmon
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

Abstract

Supervisory hybrid control systems arise when a discrete event system is used to supervise the behaviour of a continuous-state system plant. Hybrid control systems can be used as a modeling framework for intelligent control systems. This chapter discusses approaches to modeling, analysis, and synthesis for supervisory hybrid control systems that were developed under sponsorship of the NSF/EPRI intelligent control initiative.

1 Introduction

Intelligent control often refers to a collection of control methods used to control highly complex systems. Examples of high complexity systems requiring some form of intelligent control include flexible manufacturing, chemical process control, electric power distribution and generation, computer communication networks. Intelligent control systems for continuous processes contain interacting subsystems of discrete/symbolic and continuous dynamics; see for example [48, 5].

Clear understanding of the interactions between such subsystems is essential in developing advanced theories and methodologies for the control of continuous processes. Therefore the study of such hybrid control systems is necessary for the design of intelligent control systems with high degree of autonomy. It is also essential in the development of supervisory control theories and methodologies for continuous systems.

In this chapter, the modeling, analysis and synthesis of a general class of hybrid control systems is presented. Further details can be found in [5, 49, 4, 50, 23, 21, 22, 39, 46] Also in [35, 51, 37, 52, 53, 54, 55, 36, 56, 57, 58, 59, 60, 38, 61, 45, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 24, 72, 73, 74].

Supervisory hybrid control systems arise when continuous-state plants are controlled by discrete event systems. There has been considerable recent interest in these systems for a variety of reasons. Chief of these reasons is that they provide an underlying mathematical model which may provide a sound mathematical foundation upon which to base the design of intelligent control systems.

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

A good example of hybrid control is found in the aero-space industry. Spacecraft, such as commercial communication statellites, represent enormously complex systems having several operational modes. These satellites have their normal stationkeeping modes, modes associated with momentum unloading, and acquisition modes which are invoked when normal station-keeping has been lost. No single feedback control system is capable of achieving these diverse control objectives, so a mechanism for automatically switching between these different control models needs to be implemented by the on-board computer. Switching between operational modes is usually initiated by the occurrence of specific (discrete) events. For example, the loss of Earth point forces the sateelite the initiate a sequence of actions leading to Earth limb re-acquisition. The logic coordinating these actions must not only function with respect to the "symbolic" domain of legal actions, it must also initiate control actions which are well-behaved from the continuous-state side of the hybrid system.

A hybrid control system can be seen as consisting of three parts, the original continuous plant, a discrete-event system (DES) supervisor, and the interface between these systems. The interface plays a pivotal role since it gives meaning to the symbols used by the DES supervisor. A key component, therefore, of hybrid control system (and hence intelligent control system) synthesis is the determination of well-posed interfaces.

In our model, the interface has been chosen to be simply a partitioning of the state space and this is done without loss of generality. If memory is necessary to derive an effective control law, it is included in the DES controller and not in the interface. Also the piecewise continuous command signal issued by the interface is simply a staircase signal, not unlike the output of a zero-order hold in a digital control system. Including an appropriate continuous system at (the input of) the plant, signals such as ramps, sinusoids, etc., can be generated if desired. The simple interface used in our model allows analysis of the hybrid control system, and in particular development of properties such as controllability [36], stability [21], and determinism, in addition to synthesis results and the development of controller design methodologies [39, 40]. The simplicity of our interface with the resulting benefits in identifying central issues and concepts in hybrid control systems is perhaps the main characteristic of our approach. It is also what has been distinguishing our approach from other approaches (with more complex interfaces, or with restrictions on the class of systems studied) since early versions of our model first appeared in 1991 [33, 34].

The hybrid control system model presented in this paper is close to the model of [25] which was also developed for control purposes (in particular control design; our model is for analysis and design). Other models include [10]; see also the earlier references therein. The models in [6, 41] are more general but they are developed primarily for simulation purposes. The model of [28] is developed for control, but the interface is rather complex. Other approaches include [7, 13, 14, 42, 19, 17, 20, 47]. The paper by Branicky et al. [9] presents a rather detailed comparison of some of the models. Early work also included [27, 16]. For recent developments in hybrid systems research see [18].

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

The following chapter describes recent efforts sponsored by the NSF/EPRI intelligent control initiative at using supervisory hybrid control systems as a foundation for the systematic synthesis of intelligent control systems. The following sections first deal with the basic model used by our group and then discusses an overall synthesis process which guided most of our investigations. This synthesis process can be seen as an iterative procedure in which the specification, interface, and supervisor for a hybrid are iteratively refined until a desirable class of behaviours are generated. The following sections discuss the progress made in developing systematic frameworks for the analysis and synthesis of hybrid system interfaces and actuators.

2 Hybrid Control Systems

A hybrid control system, can be divided into three parts, the plant, interface, and controller as shown in Figure 1.

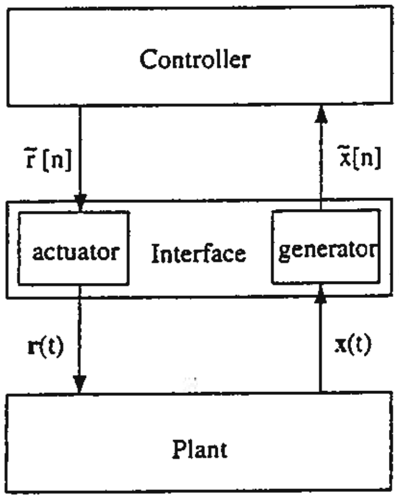

Figure 1: Hybrid Control System

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

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

For certain synthesis approaches the following model is used.

$$\dot{\bar{x}} = f_0(\bar{x}) + \sum_{i=1}^{M} r_i f_i(\bar{x}) \tag{2}$$

where $\bar{r}' = (r_1, \ldots, r_M)$ is called the reference vector and $\bar{x}$ is the state vector. The functions f_i $(i = 0, \ldots, M)$ are assumed to form a nonsingular involutive distribution, $\Delta = \{f_0, f_1, \ldots, f_M\}$, of vector fields mapping the state space back onto itself. The various vector fields are referred to as *control policies*.

The supervisor, S, is a discrete event system which is modeled as a deterministic automaton, $S = (\tilde{S}, \tilde{X}, \tilde{R}, \delta, \phi)$, where $\tilde{S}$ is the set of states, $\tilde{X}$ is the set of *plant symbols*, $\tilde{R}$ is the set of *control directives*,

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

$\delta : \tilde{S} \times \tilde{X} \to \tilde{S}$ is the state transition function, and $\phi : \tilde{S} \to \tilde{R}$ is the output function. The symbols in set $\tilde{R}$ are called control directives because they symbolize high level supervisory directives issued by the DES supervisor. The symbols in set $\tilde{X}$ are called plant symbols and are generated based on events in the plant. The action of the controller is described by the equations

$$\begin{aligned} \tilde{s}[n] &= \delta(\tilde{s}[n-1], \tilde{x}[n]) \\ \tilde{r}[n] &= \phi(\tilde{s}[n]) \end{aligned}$$

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

The interface consists of two simple subsystems, the *generator* and the *actuator*. The generator converts the continuous-time output (state) of the plant to an asynchronous, symbolic input for the controller. The set of plant events recognized by the generator is determined by a set of smooth functionals, $\{h_i : \Re^n \to \Re, i \in I\}$, whose null spaces form $n-1$ dimensional smooth hypersurfaces in the plant state space. Whenever the plant state crosses one of these hypersurfaces, a plant symbol is generated according to

$$\tilde{x}[n] = \alpha_i(\bar{x}(\tau_e[n])) \tag{3}$$

where i identifies the hypersurface which was crossed and $\tau_e[n]$ is the time of the crossing. Define a *goal set*, g_i, by the following

$$\mathrm{g}_i = \{\bar{x} : h_i(\bar{x}) < 0\} \tag{4}$$

The issuance of event $\tilde{x}_i$ marks the entry of the plant state trajectory into goal set g_i. The generator can therefore be characterized by the collection, $\mathrm{G} = \{\mathrm{g}_1, \ldots, \mathrm{g}_m\}$, of goal sets.

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

$$\mathrm{r}(t) = \sum_{n=0}^{\infty} \gamma(\tilde{r}[n]) I(t, \tau_c[n], \tau_c[n+1]) \tag{5}$$

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

$$\tau_c[n] = \tau_e[n] + \tau_d \tag{6}$$

where τ_d is the total delay associated with the interface and controller. Since the choice of $\tilde{r}$ and γ determines how the control policies in Δ are mixed, we can characterize this part of the interface by the control policy distribution, Δ, and the actuator mapping γ.

With the preceding notational conventions, a supervisory hybrid control system can be denoted by the 4-tuple, $\mathcal{H} = (S, \mathrm{G}, \Delta, \gamma)$ The problem of interest is the synthesis of a hybrid control system which realizes

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

a specified set of symbolic and nonsymbolic constraints. These control objectives are often referred to as *specifications.* These specifications constitute formal constraints (symbolic and nonsymbolic) on the plant's desired (legal) behaviour. From the supervisor's side of the interface, the plant can be treated as an equivalent DES plant and formal specifications can be placed on the behaviour of this DES plant. From the plant's side of the interface, there are traditional control theoretic constraints involving the minimization or bounding of specified signal or system norms. Any hybrid control system synthesis procedure will need to address both types of constraints. The following subsection discusses the DES plant formulation in detail.

DES Plant:

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

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

The set of DES plant model states, $\tilde{P}$, is based upon the set of hypersurfaces realized in the generator. Each open region in the state space of the plant, bounded by hypersurfaces, is associated with a state of the DES plant. Whenever a plant event occurs there is a state transition in the DES plant. Stating this more rigorously, an equivalence relation, $\equiv_p$, can be defined on the set $\{\xi \in \Re^n : h_i(\xi) \neq 0, i \in I\}$ as follows

$$\xi_1 \equiv_p \xi_2 \text{ iff } h_i(\xi_1)h_i(\xi_2) > 0, \forall i \in I. \tag{7}$$

Each of the equivalence classes of this relation is associated with a unique DES plant state. Thus it is convenient to index the set of states, $\tilde{P}$, with a binary vector, $b \in \{0,1\}^I$, such that b_i is the ith element of b and $\tilde{p}_b$ is associated with the set $\{\xi \in \Re^n : b_i = 1 \Leftrightarrow h_i(\xi) < 0\}$. The equivalence relation is not defined for states which lie on the hypersurfaces. When the continuous state touches a hypersurface the DES plant model remains in its previous state until the hypersurface is crossed. The state of the DES corresponds to the most recently entered region of the plant state space. The limit must be used because at exactly $\tau_e[n]$ the continuous state will be on a boundary. The reason for this definition of state for the DES plant model is that it represents how much can be known about the system by observing the plant symbols without actually calculating the trajectories. So after a plant symbol is generated nothing can be ascertained beyond the resulting region.

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

When two DES plant states are adjacent at (i, ξ) it means that the regions corresponding to these states are separated by the hypersurface $\mathcal{N}(h_i)$, and the point ξ lies on this hypersurface on the boundary of both regions. Thus ξ identifies a possible transition point between the regions.

3 Synthesis Problem

The synthesis of any complex engineering system is done as part of an iterative process. System engineers devise an initial system (prototype) and then incrementally refine and modify that prototype until the desired performance specifications are met. The process provides incremental refinements to both the system and the specification in such a way that the resulting system provides a realistic compromise between desired behaviour and innate system limitations. The supervisory hybrid control system synthesis method used in this paper will also be formulated as an iterative process. In this procedure, an initial hybrid system, $\mathcal{H}_0$, is formulated and then refined until specified sets of symbolic and nonsymbolic specifications are satisfied. This section describes the hybrid control system synthesis procedure in more detail.

As noted above, the "initial" hybrid control system

$$\mathcal{H}_0 = (S_0, \mathrm{G}_0, \Delta_0, \gamma_0) \tag{8}$$

represents an initial assumption about the desired behaviour and structure of the system. Assumptions on G_0, Δ_0, and γ_0 fix the hybrid system interface. The goal sets in G_0 will be determined by the set of sensors available to monitor the plant. The control policy distribution, Δ_0, and actuator mapping, γ_0, represent the initial set of actuators (controllers) which the system can use. On the symbolic side, we assume there exists a formal specification language, K, on the DES plant's behaviour. The DES supervisor, S_0, is then computed to realize this specification language.

The initial hybrid system is therefore a prototype for the desired system. For complex systems, however, this prototype may not be entirely satisfactory. For example, the original formal specification, K, may lack certain desirable properties such as liveness or controllability. This problem can be solved by computing a supervisor generating a controllable or live sublanguage of K. When the DES plant is represented by a finite automaton, there exist many methods for computing this DES supervisor. This approach to hybrid control system synthesis examines the problem from a purely symbolic perspective. The value with this approach is that synthesis methods for DES are widely studied [39] [46] [29] [12].

This purely symbolic approach to synthesis, however, fails to address constraints on the continuous-state portion of the hybrid system. In particular, it is quite possible that while the DES supervisor satisfies all of the required symbolic specifications, the performance of the controlled CSS plant may still be unsatisfactory, and one may be required to refine the interface and repeat the symbolic synthesis algorithm. This is similar

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

to current practices in the area of digital control where one may have to adjust the sampling rate (this corresponds to adjusting the interface) in order to improve the intersample behavior. In addition, there is the issue of whether or not a controller exists such that the specified symbolic transition can be realized in a safe or "stable" manner [21]. If such a controller does exist then there is a question concerning its performance as measured by traditional control theoretic performance measures (i.e. norm bounds on the CSS plants input and output signals). Therefore, while we can refine an existing formal specification (i.e. the DES supervisor) to exhibit a desirable set of symbolic behaviours, this does not mean that the system's "implementation" of that specification will be "desirable" from a traditional control theoretic standpoint.

This second type of requirement can be thought of as a "non-symbolic" specification on the controlled CSS plant's behaviour. Such a specification can be stated in a variety of ways. The most popular method is to frame the non-symbolic specification as a hard bound on a norm over the plant's signal space. We could, for example use a quadratic cost functional and uses the L_2 signal space norm to measure the cost of the controlled plant's input/output behaviour. Other performance measures can be chosen also. In recent gain-scheduling methods, we can use an H^∞ norm [32] [23] and in [26] [15] a traditional optimal control theoretic performance measure is used. The synthesis probem is then to determine whether or not a controller exists satisfying stated bounds on the costs and then to determine the controllers realizing these minimal costs. There is, of course, a rich literature on these methods also. The application of these methods to hybrid control synthesis is still a subject of considerable recent research.

The minimization of the "cost" associated with an enabled transition can be viewed as a problem in the synthesis of the hybrid control system *interface*. Therefore we see that the problem of hybrid control system synthesis, not only involves supervisor design, but also interface design. The interesting thing about this last aspect is that in redefining the interface to ensure safe and optimal transitions, we can introduce new events into the DES plant. The introduction of these new events extend or modify the DES plant upon which the original synthesis was based and this means that the designer may need to modify the original symbolic specification language, K, in an appropriate manner. By modifying the original specification, the process takes on an iterative nature which is summarized in the following procedural list.

1. Formulate the initial specification, K, and the initial hybrid system $\mathcal{H}_0$.
2. Synthesize a DES supervisor that generates a controllable sublanguage, K_c, of K.
3. Find an interface which minimizes the cost of enacting a legal transitions in the controllable sublanguage K_c.
4. Go back and readjust the specification, K, to be consistent with the modified interface and return to step 2.

The preceding iterative procedure, of course, is precisely the design process that is followed in the development

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

of many complex engineering system. It provides a systematic method for hybrid control system synthesis and hence intelligent control system synthesis. Various aspects of this iterative procedure have been deeply investigated by the NSF/EPRI intelligent control initative. The following section summarizes the progress of these investigations.

4 Synthesis Methods

This section summarizes a variety of results concerning synthesis methods for supervisory hybrid control sytems. As noted in the preceding sections, synthesis methods can be viewed in terms of interface and supervisor synthesis. Supervisor synthesis for hybrid systems is very similar to traditional logical DES synthesis methods, though there are some differences due to the inherent non-determinism of the DES plant. Interface synthesis consists of two parts. One first needs to establish the existence of interfaces ensuring that the supervisor's enabled actions are appropriate, for example they are "safe" or T-stable, and they satisfy certain cost requirements used in combinatorial optimization of the supervisor. We must then determine the interface that minimizes the cost of this appropriate transition in terms of the continuous system dynamics. The following subsections discuss these issues in greater detail.

4.1 Supervisor Synthesis

In this section, we use the language generated by the DES plant to examine the controllability of the hybrid control system. This work builds upon the work done by Ramadge and Wonham on the controllability of discrete event systems in a logical framework [31, 29, 30, 43, 44]. Here we adapt several of those results and apply them to the DES plant model obtained from a hybrid control system.

Before existing techniques, developed in the logical DES framework can be extended, certain differences must be dealt with. The Ramadge-Wonham model (RWM) consists of two interacting DES's called here the *RWM generator* and *RWM supervisor*. The RWM generator is analogous to the DES plant and the RWM supervisor is analogous to the DES supervisor. The RWM generator shares its name with the generator found in the hybrid control system interface but the two should not be confused. In the RWM, the plant symbols are usually referred to as "events", but we will continue to call them plant symbols to avoid confusion. The plant symbols in the RWM are divided into two sets, those which are controllable and those which are uncontrollable: $\tilde{X} = \tilde{X}_c \cup \tilde{X}_u$. A plant symbol being controllable means that the supervisor can prevent it from being issued by the RWM generator. When the supervisor prevents a controllable plant symbol from being issued, the plant symbol is said to be *disabled*. The plant symbols in $\tilde{X}_c$ can be individually disabled, at any time and in any combination, by a command from the RWM supervisor, while the plant symbols in $\tilde{X}_u$ can never be disabled. This is in contrast to our DES plant where each command (controller symbol)

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

from the DES supervisor disables a particular subset of $\tilde{X}$ determined by the complement of the set given by the transition function, ψ. Furthermore, this set of disabled plant symbols depends not only on the controller symbol but also the present state of the DES plant. In addition, there is no guarantee that any arbitrary subset of $\tilde{X}$ can be disabled while the other plant symbols remain enabled.

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

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

It is possible to determine whether a given RWM generator can be controlled to a desired language [31]. That is, whether it is possible to design a controller such that the RWM generator will be restricted to some target language K. Such a controller can be designed if K is prefix closed and

$$\bar{K}\tilde{X}_u \cap L \subset \bar{K} \tag{9}$$

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

When equation 9 is true for a given RWM generator, the desired language K is said to be controllable, and provided K is prefix closed, a controller can be designed which will restrict the generator to the language K. This condition requires that if an uncontrollable symbol occurs after the generator has produced a prefix of K, the resulting string must still be a prefix of K because the uncontrollable symbol cannot be prevented.

Since the DES plant model belongs to a slightly different class of automata than the RWM, we present another definition for controllable language which applies to the DES plant. We assume in this section that we are dealing with observable DES plant models, that all languages are prefix closed, and that q_0 is the initial state.

Definition 1 *A language, K, is controllable with respect to a given DES plant if $\forall \tilde{x} \in K$, there exists $\rho \in \tilde{R}$ such that*

$$\tilde{x}\lambda(q, \psi(q, \rho)) \subset K, \tag{10}$$

where $q = \text{obs}(q_0, \tilde{x})[N]$.

This definition requires that for every prefix of the desired language, K, there exists a control, ρ, which will enable only symbols which will cause string to remain in K.

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

Proposition 1 *If the language K is controllable according to (1), then a controller can be designed which will restrict the given DES plant to the language K.*

Since the DES plant can be seen as a generalization of the original RWM, the conditions in (10) reduce to those of (9) under appropriate restrictions; namely that the plant symbols fall into a controllable/uncontrollable dichotomy and a control policy exists to disable any combination of controllable plant symbols. This is indeed the case.

If the desired language is not attainable for a given DES, it may be possible to find a more restricted language which is. If so, the least restricted behavior is desirable. [31] and [43] describe and provide a method for finding this behavior which is referred to as the *supremal controllable sublanguage*, $K^{\dagger}$, of the desired language. The supremal controllable sublanguage is the largest subset of K which can be attained by a controller. $K^{\dagger}$ can be found via the following iterative procedure.

$$K_0 = K \tag{11}$$

$$K_{i+1} = \{w : w \in \bar{K}, \bar{w}\tilde{X}_u \cap L \subset \overline{K_i}\} \tag{12}$$

$$K^{\dagger} = \lim_{i \to \infty} K_i \tag{13}$$

Once again, this procedure applies to the RWM. For hybrid control systems, the supremal controllable sublanguage of the DES plant can be found by a similar iterative scheme.

$$K_0 = K \tag{14}$$

$$K_{i+1} = \{w \in K : \forall \tilde{x} \in \bar{w} \; \exists \; \rho \in \tilde{R} \text{ such that } \tilde{x}\lambda(q, \psi(q, \rho)) \subset K_i\} \tag{15}$$

$$K^{\dagger} = \lim_{i \to \infty} K_i \tag{16}$$

This result yields the following proposition.

Proposition 2 *For a DES plant and language K, $K^{\dagger}$ is controllable and contains all controllable sublanguages of K.*

Formula for the Supremal Controllable Sublanguage

A formula for the supremal controllable sublanguage similar to that found in [8] has been derived. This formula assumes that the DES plant's nondeterminism enters solely through the plant's enabling function. In particular, the DES plant, G, will be represented as the 5-tuple, $G(\{\tilde{P}, \tilde{X}, \tilde{R}, \xi, \psi\}$, where the alphabets

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

$\tilde{P}$, $\tilde{X}$, and $\tilde{R}$ have the customary interpretation. The enabling function is $\xi : \tilde{P} \times \tilde{R} \rightarrow P_{\tilde{X}}$ where $P_{\tilde{X}}$ is the power set of alphabet $\tilde{P}$ and $\psi : \tilde{P} \times \tilde{X} \rightarrow \tilde{P}$ is the state transition function. In [46], the notion of a controllable sublanguage for this call of nondeterministic DES plant is defined as follows:

Definition 2 *A language, $K \subset L(G)$, generated by the DES plant is controllable if for all $\omega \in \overline{K}$,*

- *there exists $\tilde{r} \in \tilde{R}$ such that $\omega\xi(\psi(\tilde{p}_0, \omega), \tilde{r}) \subseteq \overline{K}$,*
- *and if $\omega = \omega_b\alpha$ ($\alpha \in \tilde{X}$) then there exists $\tilde{r}_b \in \tilde{R}$ such that $\alpha \in \xi(\psi(\tilde{p}_0, \omega_b), \tilde{r}_b)$ and $\omega_b\xi(\psi(\tilde{p}_0, \omega_b), \tilde{r}_b) \subseteq \overline{K}$.*

This definition for controllability consists of two specific parts. The first part obviously requires that there exist control symbols which ensure that no deadlocked strings are generated. However, it will also be necessary to ensure that the strings in K are generated by control symbols which do not generate illegal behaviours. This is precisely what thesecond condition ensure. It guarantees that a string in K is indeed realized by a switching control policy in our nondeterministic plant.

DES Plant Identification and Learning

The synthesis methods used above require that the DES plant be known. In certain situations, this prior knowledge may not be available. In this case, it may be necessary to identify the DES plant by observing the system's behaviour. We have investigated the use of computationally efficient inductive learning methods for the identification of DES plants [22] [45]. These methods are based on a modification of the L^*-learning procedure [1].

A flowchart of the L^*-procedure is shown below. The algorithm constructs an data structure called an *observation table*. The formulation of a *complete* observation table allows us to identify the Nerode equivalence classes of the DES to be identified. From this information, then, the MyHill-Nerode construction can be used to construct a possibly non-deterministic automaton consistent with samples of the DES plant's behaviour.

For our application, direct application of the L^*-procedure is not possible. This is because of uncontrollable events in the DES plant. To deal with uncontrollable events, we modify the oracles used by the algorithm. Whereas the original L^*-procedure used a static membership oracle, we will use a membership oracle which is conditioned on a list of observed illegal plant behaviours. See [45] for details on the oracle formulation. With this reformulated L^*-algorithm it is then possible to determine computationally efficient methods for the identification of partially specified DES plants. Several examples in [22, 45] illustrate this method.

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

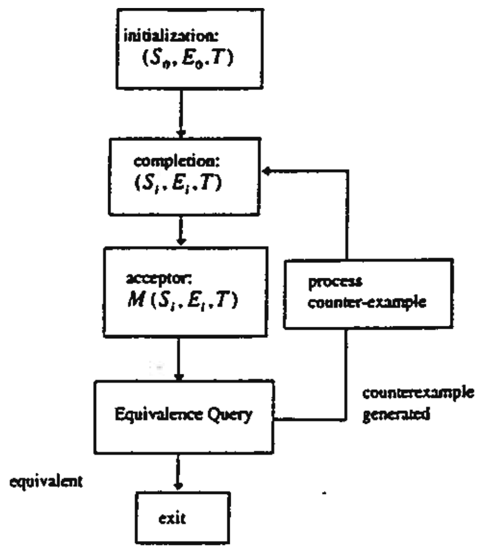
initialization: (S_0, E_0, T)
completion: (S_i, E_i, T)
acceptor: $M(S_i, E_i, T)$
process counter-example
Equivalence Query
counterexample generated
equivalent
exit

Figure 2: Flowchart for L^* Algorithm

4.2 Transition Stability

The synthesis of a supervisor enforcing controllable plant behaviours is only useful if there exists an interface capable of realizing the enabled transitions. One important issue concerns the verification that such transitions are realizable. In [21], a specific notion of safety called T-stability was introduced. For certain classes of systems, this safety criterion was easily testable using an inductive learning procedure. In particular, we will say that the specified language, $L(\mathcal{K})$, is *realizable* if all symbolic transitions occur in a "stable" manner. By "stable", we mean that the symbol sequences issued by the generator are "invariant" to small perturbations of the CSS plant's initial state. The following definition [21] formally states this notion of transition or T-stability.

Definition 3 *Let $\Phi_T(\bar{x}, \tilde{r})$ be the state of the plant assuming initial condition $\bar{x}$ under the action of control directive $\tilde{r}$. A transition from goal set g_0 to g_T under control directive, $\tilde{r}$, is said to be* stable *if and only if for all $\bar{x}_0 \in g_0$ there exists an open neighborhood, $N(\bar{x}_0, \epsilon)$, and a finite time $T < \infty$ such that*

- *$\Phi_T(\bar{x}, \tilde{r})$ is an open set in g_T*
- *and the plant symbols, $\tilde{x}$, issued by the transition are identical*

for all $\bar{x} \in N(\bar{x}_0, \epsilon)$

Figure 3 illustrates an unstable transition. This figure shows a state trajectory which transfers the initial plant state, $\bar{x}_0$, between goal sets g_i and g_j. As shown in the figure, the resulting state trajectory $\bar{x}(t)$ is

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

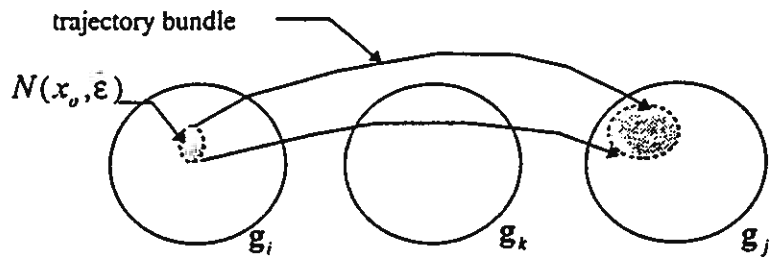

Figure 3: Example of an unstable (unrealizable) symbolic transition

tangential to the boundary of another goal set g_k. If this transition were to occur in a stable manner, then the transfer must occur so that the time between successive events, τ, is bounded and so that the symbols issued by the interface are unchanged by small changes in initial state. Stability therefore requires that the "bundle" of state trajectories originating from an open neighborhood , $N(\bar{x}_0, \epsilon)$, not intersect the set g_k. The trajectories in the figure do not possess this property, so this transition is not T-stable.

One interesting aspect of the preceding definition, is that it can be easily tested for when the plant is affine in the available control policies. In this case, it is possible to state a sufficient condition for T-stability which takes the form of a set of linear-inequalities. The ith inequality constraint has the following form

$$\left(L_{f_0}V_i \quad L_{f_1}V_i \cdots \quad L_{f_M}V_i \right) \begin{pmatrix} 1 \\ r_1 \\ \cdot r_M \end{pmatrix} < 0 \tag{17}$$

where V_i is a positive definite function associated with the i goal set constraining the transition and L_{f_j} is the Lie derivative of the jth control policy.

The preceding condition is a sufficient condition on the transition's T-stability. Feasible points in $\bar{R}$ satisfiying the inequality system represent constant control vectors guaranteeing the T-stability of the arc. A finite-time algorithm determining whether or not the feasible set is empty or not was introduced in [21]. This algorithm made used the central-cut ellipsoid method as part of an inductive learning procedure. In certain cases, this method could be shown to declare the feasible set empty or not after a finite number of updates that scaled in a polynomial manner with the number of available control policies. The other useful thing about this procedure is that if the feasible set was non-empty, then it would determine controller (interface actuator) that was within this feasible set. The procedure therefore provided a method for on-line interface synthesis.

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

4.3 Interface Synthesis

The preceding section noted that the verifying the existence of a T-stable interface could sometimes be used to identify actual controllers realizing the required transition. These methods, therefore, provided an approach to interface synthesis. The approach, unfortunately, does not optimise performance because it only seeks a feasible controller, not an optimal controller. In the following section, we discuss recent work investigating the synthesis of "optimal" interfaces.

One approach to optimal interface synthesis was originally proposed by the Kohn-Nerode group [26]. This method constructs a chattering control policy that approximates a time-optimal path from the plant's initial state to the desired termnal goal set. The method had a computational simplicity due to the fact that the chattering control problem can often be cast as a linear programming problem.

The following subsections discuss two other approaches to interface synthesis. The first method uses a gain-scheduling approach. The second method uses invariants of the control policy distribution to determine event boundaries.

4.4 Gain-Scheduled Interface Synthesis

One disadvantage of the Kohn-Nerode method [26], however, is its reliance on an ill-defined adaptation unit that would improve system robustness. Partly in response to this deficiency, another approach based on H^∞ gain-scheduling concepts was proposed in [23]. The basic ideas is as follows: Assume that there exists an ideal reference trajectory, $\bar{x}_r(t)$, between an initial goal set g_i and the terminal goal set g_j. Define a sequence of goal sets g_m $(m = 1, \ldots, N_g)$ which are all bounded subsets of the state space. Associated with the mth goal set will be the mth cntroller. This controller will be selected to ensure T-stable transitions between intermediate goal sets. These new control polices are then added to the interface actuator and the new "intermediate" goal sets are added to the interface generator.

In certain cases, linear control agents exhibiting robust stability can be used as the new controllers. The agents can only be used if the plant's perturbation over the reference trajectory is relatively smooth [32]. This notion of smoothness can be formalized as follows. Rewrite the plant's state equations in a linear state variable form,

$$\dot{\bar{x}} = A(\bar{x})\bar{x} + B(\bar{x})\bar{r} \tag{18}$$

$$\bar{y} = C(\bar{x})\bar{x}. \tag{19}$$

in which the system matrices are functions of the plant state, $\bar{x}$. At a point $\bar{x}_0$ within the plant's state space, the plant's nominal transfer function matrix is defined as

$$P(s|\bar{x}_0) = C(\bar{x}_0)\left(sI - A(\bar{x}_0)\right)^{-1} B(\bar{x}_0) \tag{20}$$

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

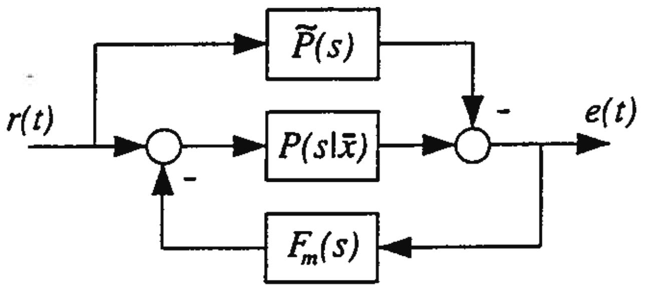

Figure 4: Hybrid Control System Control Agent

Now, consider points $\bar{x}$ near $\bar{x}_0$. We will assume that any local plant, $P(s|\bar{x})$, associated with a point $\bar{x}$ is related to $P(s|\bar{x}_0)$ through a bounded multiplicative perturbation yielding

$$P(s|\bar{x}) = (I + \Delta(s|\bar{x}, \bar{x}_0))P(s|\bar{x}_0) \tag{21}$$

with

$$\bar{\sigma}(\Delta(j\omega|\bar{x}, \bar{x}_0)) \leq \delta(\omega)||\bar{x} - \bar{x}_0||_2 \tag{22}$$

for all real ω. Here, $||\bar{x}||_2$ is a Euclidean 2-norm, $\Delta(s|\bar{x}, \bar{x}_0)$ is a stable, rational transfer function matrix, $\bar{\sigma}$ represents the maximum singular value, and $\delta(\omega)$ is a non-negative real function of ω. To model the plant in this fashion, we will assume that the multiplicative uncertainty does not change the number of closed right half plane poles of $P(s|\bar{x}_0)$. It will be further assumed that the plant state changes in a continuous manner with the output; i.e. $||\bar{x} - \bar{x}_0|| < K||\bar{y} - \bar{y}_0||$ and $K < \infty$.

The control agent architecture to be used is shown in figure 4. The controller is a model following state feedback control system. It uses an internal reference model which is represented by a stable minimum phase transfer function, $\tilde{P}(s)$. This reference represents an "ideal" trajectory, denoted $\bar{x}_r(t)$, between the two goal sets corresponding to a T-stable transition. As the reference model state moves along $\bar{x}_r(t)$, the controller that is used switches sequentially between N_g control agents. The index ($m = 1, \ldots, N_g$) will be used to specify which control agent is "active". The mth control agent uses output feedback through the linear system $F_m(s)$.

A design procedure for constructing the preceding "gain-scheduled" interface was proposed. This procedure is given below

Step 1: Generate a reference trajectory, $\bar{x}_r(t)$, and obtain a sequence of M setpoints $S = \{\bar{x}_r^j\}$ ($j = 1, \ldots, M$).

Step 2: For each setpoint in S determine the local linear transfer function $P(s|\bar{x}_r^j)$.

Step 3: Initialize the iteration with i=1, N=0, and $m = 1$.

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

Step 4: Compute an aggregated plant, $\hat{P}_m(s)$ for all $j \in [i, i+N]$.

Step 5: Determine the H^∞ norm controller, $F_m(s)$. If no such controller exists then set $i = i + N - 1$, N=0, and $m = m + 1$. Otherwise set $N = N + 1$.

Step 6: Go to step 4 if $i + N < M$.

Step 7: If no controller was found in the last iteration of Step 5, then declare a failure. The procedure was unable to T-stablize the transition. Otherwise declare success.

An important part of the above procedure involves the computation of the *aggregated plant.*

Determination of the aggregated plant can be accomplished by solving an appropriately formulated relative model error reduction problem. It is then possible to compute an H^∞ controller for the aggregated plant. The implication here, of course, is that by designing a controller for a single uncertain linear plant $\hat{P}_m(s)$ that meets robust stability and peformance requirements, we simultaneously satisfy the criteria for all local plants $\{P(s|\bar{x})|\bar{x} \in \mathrm{g}_m\}$ where g_m is a well-defined intermediate goal set enclosing the setpoints in $\mathcal{S}_m$. The above procedure therefore provides an additional set of goal sets and control policies which can be added to the interface's goal set collection, G, and control policy distribution, Δ_c. By construction, the resulting interface should T-stabilize the commanded transition.

4.5 Invariant Based Approach

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

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

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

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

A *common flow region* (CFR), for given target region, is a set of states which can be driven to the target region with the same control policy [40]. The following definition is used.

Definition 4 *For a plant given by Equation (1) and a target region,* $\mathbf{T}$*, the set* $\mathbf{B}$ *is a common flow region for* $\mathbf{T}$ *if for all* $\bar{x}(0) \in \mathbf{B}$*, there exists* $\tilde{r} \in \tilde{R}$ *and* $t_2, t_1 < t_2$ *such that* $x(t) \in \mathbf{B}, t \leq t_1$ *and* $x(t) \in \mathbf{T}, t_1 < t < t_2$ *subject to* $\dot{\bar{x}}(t) = f(\bar{x}(t), \gamma(\tilde{r}))$

To design the generator, it is necessary to select the set of hypersurfaces, $\{h_i : \mathbf{X} \to \Re \mid i \in I\}$ and the associated functions, $\{\alpha_i : \mathcal{N}(h_i) \to \tilde{R} \mid i \in I\}$, described above.

The following proposition gives sufficient conditions for the hypersurfaces bounding $\mathbf{B}$ and $\mathbf{T}$ to ensure that all state trajectories in $\mathbf{B}$ will reach $\mathbf{T}$.

Proposition 3 *Consider a flow generated by a smooth vector field,* f*, a target region,* $\mathbf{T} \subset \mathbf{X}$*, a set of smooth hypersurfaces,* $h_i, i \in I_B \subset 2^I$*, and a smooth hypersurface (exit boundary),* h_e*. Assume that* $\mathbf{B} = \{\xi \in \mathbf{X} : h_i(\xi) < 0, h_e(\xi) > 0, \forall i \in I_B\} \neq \emptyset$*. If*

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

then for all $\xi \in \mathbf{B}$ *there is a finite time,* t*, such that* $\bar{x}(0) = \xi, \bar{x}(t) \in \mathbf{T}$.

The preceding proposition can be used to help find the appropriate interface. A computerized procedure has been developed to identify the regions directly. This is achieved by simply calculating the state trajectories backwards from the target region and recording the states which lie on these trajectories. This identifies a common flow region. The procedure requires quantizing and bounding the state space so that the computer will have a finite number of state values to deal with.

To use the program the designer must choose the quantization levels for each state, Δx_i, and the range of each state, $x_{i,\min}$ and $x_{i,\max}$. As can be seen, the order of the computational complexity is q^n, where q is the number of quantization levels and n is the number of states. This limits the size of systems which can be handled in a reasonable amount of time. The limit depends on the computational power available and the on the designer's idea of what is "reasonable".

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

After the designer has quantized and bounded the state space, the procedure requires two additional pieces of information from the designer. The set of available plant inputs (control policies) must be provided and the target region must be specified in terms of the cells. That is, the cells which lie in the target region must be identified as such by the designer.

Once the program has the requisite information, it proceeds as follows. It will locate all cells from which it is possible to reach the target region via the application of any one control policy. The program resets the target region to include all these cells and then repeats the algorithm. In this way, all cells from which it is possible to reach the original target via the application of two control policies in sequence are identified. The program will repeat the algorithm as many times as the designer has specified. When the program finishes, each cell is marked with the control policy which should be used to reach the target, either directly or as the first in a sequence of control policies.

The following example involving an autonomous underwater vehicle is used to illustrate this invariant-based approach to interface design. This example uses a simplified model of a six-degree-of-freedom UUV. The three types of linear displacement are *surge*, *sway*, and *heave*, which represent translation in the x, y, and z, directions respectively. The three types of angular displacement are *roll*, *pitch*, and *yaw*. The model employed here has six states which are the time derivatives of the three linear displacements and the magnitudes of the three angular displacements. By expanding to a nine state model, the magnitudes of the linear displacements, could also be included. They are omitted here to simplify the control problem and because they do not affect the dynamics of the UUV. The UUV model has three inputs which control the rudder, stern plane, and screw.

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

$$\begin{array}{rcccccl} 0 & \leq & x_1 & \leq & 1, & \Delta x_1 & = & 0.1 \\ -0.5 & \leq & x_2 & \leq & 0.5, & \Delta x_2 & = & 0.2 \\ -0.5 & \leq & x_3 & \leq & 0.5, & \Delta x_3 & = & 0.2 \\ -\pi/2 & \leq & x_4 & \leq & \pi/2, & \Delta x_4 & = & \pi/20 \\ -\pi/2 & \leq & x_5 & \leq & \pi/2, & \Delta x_5 & = & \pi/20 \end{array}$$

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

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

The target region consists of the following interval.

$$\begin{bmatrix} 0.1 \\ -0.1 \\ -0.1 \\ -\pi/40 \\ -\pi/40 \end{bmatrix} < \bar{x} < \begin{bmatrix} 0.2 \\ 0.1 \\ 0.1 \\ \pi/40 \\ \pi/40 \end{bmatrix} \tag{23}$$

The results of a simulation is presented here. The trial has the following initial conditions.

$$\bar{x} = \begin{bmatrix} \text{surge} \\ \text{sway} \\ \text{heave} \\ \text{pitch} \\ \text{yaw} \end{bmatrix}, \bar{x}_1 = \begin{bmatrix} 0.8 \\ 0 \\ 0 \\ -1.17 \\ 1.00 \end{bmatrix} \tag{24}$$

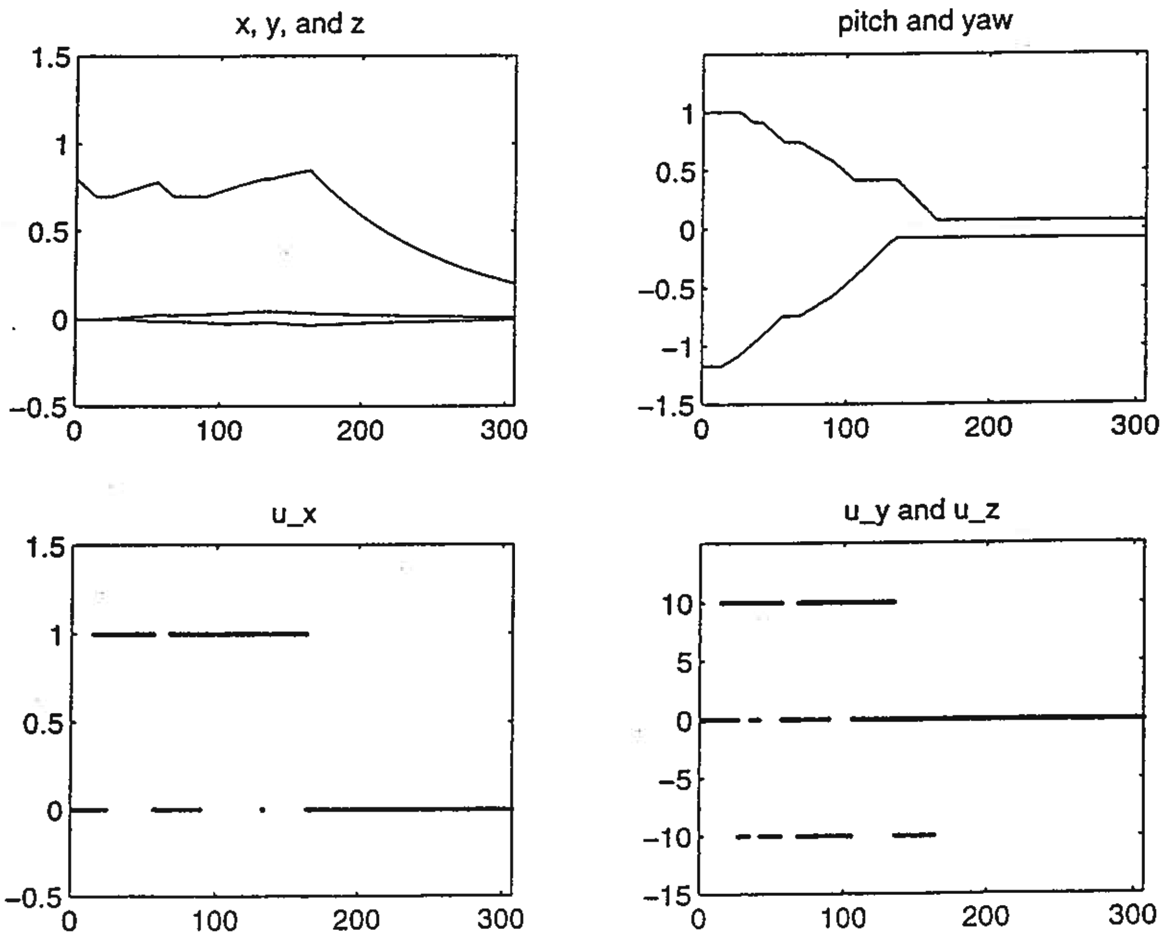

Figure 5: UUV Simulation #1

Results of the trial are shown in Figure 5, which consists of four graphs. The graph in the upper left

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

shows the surge, sway, and heave over time. The graph in the upper right shows the pitch and yaw over time. The trajectories on these two graphs can be distinguished by noting the initial conditions. The two lower graphs show the control signal. On the lower left is the propeller speed which is either 0 or 1. The segments which appear to overlap reveal the presence of chattering (due to quantization). The final graph shows the stern plane angle and the rudder angle. In Figure 5, the stern plane switches between 0 and 10 and the rudder angle switches between -10 and 0. As can be seen, the basic control strategy which developed is simply to accelerate and turn until the pitch and yaw are within the bounds of the target region, and then coast until the forward speed is also in the target.

5 Summary

This study has significantly enhanced our understanding of hybrid control systems. Our contributions to the area of hybrid control systems under the sponsorship of this grant is now briefly described.

We first developed mathematical models for hybrid control systems that allowed us to identify and study fundamental properties such as determinism and stability. We were among the first to clearly stress the importance of the interface and its impact on our ability to control the system. The characteristic of our model, which allows analysis of properties and synthesis of supervisors is the mathematical simplicity of the interface. This allows us to derive general results about system properties and general guidelines for interface design and control design methods that are not case dependent.

Specific research accomplishments are:

(a) Development of a model appropriate for analysis and control synthesis

(b) Introduction and development of important system theoretic properties such as determinism, quasi-determinism and transition stability. Recognizing the importance of the interface.

(c) Developing a control design procedure by introducing a DES model of the plant and extending existing design approaches for logical DES plants to hybrid systems.

(d) Developing an interface design procedure and a supervisor design procedure based on the invariants of the differential equation descriptions of the plant.

(e) Introducing and developing inductive learning methods to learn logically stable hybrid system interfaces, stabilizing controllers and optimal logical discrete event system controllers.

(f) Introducing and developing interior point optimization methods to develop polynomial-time algorithms for optimal collections of control agents used in the supervision of hybrid systems. This is discussed

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

in the companion chapter in this book.

(g) The application of these new results to examples from several application areas to illustrate and for proof of concept. They include the design of a supervisor for a distillation column; the design of a supervisor for an underwater unmanned vehicle; the design of hierarchical and supervisory controllers, of radial basis neural networks and of controller colonies for linear systems, of discrete event models of manufacturing systems; and the design of a controller that autonomously stabilizes a detailed simulation model of the Olympus communication satellite of the European Space agency.

Acknowledgement

The authors would like to acknowledge the partial financial support of the National Science Foundation (MSS92-16559, IRI91-09289) and The Electric Power Research Institute (RP8030-06).

References

[1] D. Angluin (1987), "Learning regular sets form queries and counter-examples", *Int. J. Information and Computation*, Vol 75, No. 1, pp. 87-106, 1987.

[2] P. J. Antsaklis, "Defining intelligent control", *IEEE Control Systems Magazine*, vol. 14, no. 3, June 1994, Report of the Task Force on Intelligent Control, P. J. Antsaklis Chair.

[3] P. J. Antsaklis, M. D. Lemmon, and J. A. Stiver, "Hybrid system modeling and event identification", Technical Report of the ISIS Group ISIS-93-002, University of Notre Dame, Notre Dame, IN, January 1993.

[4] P. J. Antsaklis, M. D. Lemmon, and J. A. Stiver, "Learning to be autonomous: Intelligent supervisory control", Technical Report of the ISIS Group ISIS-93-003, University of Notre Dame, Notre Dame, IN, April 1993, To appear as a chapter in the IEEE Press book Intelligent Control: Theory and Applications.

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

[6] A. Back, J. Guckenheimer, and M. Myers, "A dynamic simulation facility for hybrid systems", In *Hybrid Systems*, R. L. Grossman, A. Nerode, A. P. Ravn and H. Rischel, Eds., volume 736 of *Lecture Notes in Computer Science*, pp. 255–267. Springer-Verlag, 1993.

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

[7] A. Benveniste and P. Le Guernic, "Hybrid dynamical systems and the signal language", *IEEE Transactions on Automatic Control*, vol. 35, no. 5, pp. 535–546, May 1990.

[8] R.D. Brandt, V.K. Garg, R. Kumar, F. Lin, S.I. Marcus, and W.M. Wonham Formulas for calculating supremal controllable and normal sublanguages Systems and Control Letters, 15(8):111-117, 1990.

[9] M. Branicky, V. Borkar, and K. Mitter, "A unified framework for hybrid control", In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pp. 4228–4234, Lake Buena Vista, FL, December 1994.

[10] R. Brockett, "Language driven hybrid systems", In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pp. 4210–4214, Lake Buena Vista, FL, December 1994.

[11] C. Chase and P. J. Ramadge, "Dynamics of a switched n buffer system", In *Proceedings of the Twenty-Eighth Annual Allerton Conference on Communication, Control, and Computing*, pp. 455–464, University of Illinois at Urbana-Champaign, October 1991.

[12] S-L Chung, S. Lafortune, F. Lin (1992), "Limited lookahead policies in supervisory control of discrete event systems", *IEEE Trans. Automatic Control*, Vol 37, No. 12, pp. 1921-1935, Dec. 1992.

[13] A. Deshpande and P. Varaiya, "Viable control of hybrid systems", In the Ph.D. Dissertation of the first author, June 1994, ftp: eclair.eecs.berkeley.edu.

[14] S. Di Gennaro, C. Horn, S. Kulkarni, and P. Ramadge, "Reduction of timed hybrid systems", In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pp. 4215–4220, Lake Buena Vista, FL, December 1994.

[15] X. Ge, W.Kohn, and A. Nerode (1994), "algorithms for chattering approximations to relaxed optimal controls", Technical Report 94-23, Mathematical Sciences Institute, Cornell University, April 1994.

[16] A. Gollu and P. Varaiya, "Hybrid dynamical systems", In *Proceedings of the 28th Conference on Decision and Control*, pp. 2708–2712, Tampa, FL, December 1989.

[17] R. Grossman and R. Larson, "Viewing hybrid systems as products of control systems and automata", In *Proceedings of the 31st Conference on Decision and Control*, pp. 2953–2955, Tucson, AZ, December 1992.

[18] R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, Springer-Verlag, 1993.

[19] L. Holloway and B. Krogh, "Properties of behavioral models for a class of hybrid dynamical systems", In *Proceedings of the 31st Conference on Decision and Control*, pp. 3752–3757, Tucson, AZ, December 1992.

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

[20] W. Kohn and A. Nerode, "Multiple agent autonomous hybrid control systems", In *Proceedings of the 31st Conference on Decision and Control*, pp. 2956–2966, Tucson, AZ, December 1992.

[21] M. D. Lemmon and P. J. Antsaklis, "Inductively inferring valid logical models of continuous-state dynamical systems", *Theoretical Computer Science*, vol. 138, pp. 201–210, 1995.

[22] M. Lemmon, P. Antsaklis, X. Yang, C. Lucisano (1995), "Control System Synthesis through Inductive Learning of Boolean Concepts", *IEEE Control Systems Magazine*, June 1995.

[23] M. Lemmon, C. Bett, P. Szymanski, and P. Antsaklis, "constructing hybrid control systems from robust linear control agents", in *Hybrid Systems II.* Lecture Notes in Computer Science, Springer-Verlag, to appear in 1996.

[24] M. Lemmon, C. Bett, "stable H^∞ extensions of hybrid control systems", *Proceedings of CDC*, Dec. 1995, New Orleans, Louisiana

[25] A. Nerode and W. Kohn, "Models for Hybrid Systems: Automata, Topologies, Controllability, Observability", In *Hybrid Systems*, R. L. Grossman, A. Nerode, A. P. Ravn and H. Rischel, Eds., pp. 317–356. Springer-Verlag, 1993.

[26] A. Nerode and W. Kohn (1993), "multiple agent hybrid control architecture", in [25], pp. 297-316, Springer-Verlag, 1993.

[27] P. Peleties and R. DeCarlo, "A modeling strategy with event structures for hybrid systems", In *Proceedings of the 28th Conference on Decision and Control*, pp. 1308–1313, Tampa, FL, December 1989.

[28] P. Peleties and R. DeCarlo, "Analysis of a hybrid system using symbolic dynamics and petri nets", *Automatica*, vol. 30, no. 9, pp. 1421–1427, 1994.

[29] P. Ramadge and 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, January 1987.

[30] P. Ramadge and W. M. Wonham, "The control of discrete event systems", *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81–89, January 1989.

[31] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes", Systems Control Group Report 8515, University of Toronto, Toronto, Canada, November 1985.

[32] J.S. Shamma, and M. Athans (1990), " analysis of gain scheduled control for nonlinear plants", *IEEE Trans. on Automatic Control*, Vol. AC-35, No. 8, pp. 898-907, August 1990.

[33] J. A. Stiver, "Modeling of hybrid control systems using discrete event system models, Master's thesis, Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, May 1991.

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

[34] J. A. Stiver and P. J. Antsaklis, "A novel discrete event system approach to modeling and analysis of hybrid control systems", In *Proceedings of the Twenty-Ninth Annual Allerton Conference on Communication, Control, and Computing*, University of Illinois at Urbana-Champaign, October 1991.

[35] J. A. Stiver and P. J. Antsaklis, "Modeling and analysis of hybrid control systems", In *Proceedings of the 31st Conference on Decision and Control*, pp. 3748–3751, Tucson, AZ, December 1992.

[36] J. A. Stiver and P. J. Antsaklis, "On the controllability of hybrid control systems", In *Proceedings of the 32nd Conference on Decision and Control*, pp. 3748–3751, San Antonio, TX, December 1993.

[37] J. A. Stiver and P. J. Antsaklis, "State space partitioning for hybrid control systems", In *Proceedings of the American Control Conference*, pp. 2303–2304, San Francisco, California, June 1993.

[38] J. A. Stiver, P. J. Antsaklis, and M. D. Lemmon, "Digital control from a hybrid perspective", In *Proceedings of the 33rd Conference on Decision and Control*, pp. 4241–4246, Lake Buena Vista, FL, December 1994.

[39] J.A. Stiver, P.J. Antsaklis, M.D. Lemmon, "A logical DES approach to the design of hybrid control systems", technical report of the ISIS group ISIS-94-011, University of Notre Dame, Notre Dame, IN Oct. 1994 (revised: May 1995), to appear in *Mathematical and Computer Modeling*, special issue on discrete event systems

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

[41] L. Tavernini, "Differential automata and their discrete simulators", *Nonlinear Analysis, Theory, Methods, and Applications*, vol. 11, no. 6, pp. 665–683, 1987.

[42] M. Tittus and B. Egardt, "Control-law synthesis for linear hybrid systems", In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pp. 961–966, Lake Buena Vista, FL, December 1994.

[43] W. M. Wonham and P. J. Ramadge, "On the supremal controllable sublanguage of a given language", Systems Control Group Report 8312, University of Toronto, Toronto, Canada, November 1983.

[44] W. M. Wonham and 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.

[45] X. Yang, M.D. Lemmon, P. Antaklis, "inductive inference of uncertain logical DES", *Proceedings of International Symposium on Intelligent Control*, August, 1995, Monterey, CA.

[46] Xiaojun Yang, M. Lemmon, P. Antsaklis, "On the supremal controllable sublanguage of nondeterministic plants arising in hybrid control systems", to appear in *IEEE Trans. on Automatic Control*, Jan 1996.

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

[47] B. P. Zeigler, "DEVS representation of dynamical systems: Event based intelligent control", *Proceedings of the IEEE*, vol. 77, no. 1, pp. 72–80, 1989.

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

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

[50] J.A. Stiver, P.J. Antsaklis and M.D. Lemmon , "Interface and Controller Design for Hybrid Control Systems", Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-002, Univ of Notre Dame, February 1995. Chapter in Hybrid Systems II, P. Antsaklis, W. Kohn, A. Nerode, S. Sastry Eds., Lecture Notes in Computer Science, LNCS, Springer-Verlag, 1995. To appear.

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

[52] J. A. Stiver and P. J. Antsaklis, "Extracting Discrete Event System Models from Hybrid Systems", Proc. of the 8th IEEE International Symposium on Intelligent Control, pp. 298-301, Chicago, IL, August 25-27, 1993.

[53] M. D. Lemmon and P. J. Antsaklis, "Hybrid Systems and Intelligent Control", Proc. of the 8th IEEE International Symposium on Intelligent Control, pp. 174-179, Chicago, IL, August 25-27, 1993.

[54] P. J. Antsaklis, M. D. Lemmon and J. A. Stiver, "Hybrid System Modeling and Event Identification", Proc. of the Thirty-First Annual Allerton Conference on Communication Control and Computing, pp 64-73, Univ. of Illinois at Urbana-Champaign, Sept. 29-Oct. 1, 1993.

[55] J. A. Stiver and P. J. Antsaklis, "DES Supervisor Design for Hybrid Control Systems", Proc. of the Thirty-First Annual Allerton Conference on Communication Control and Computing, pp 657-666, Univ. of Illinois at Urbana-Champaign, Sept. 29-Oct. 1, 1993.

[56] M. D. Lemmon and P. J. Antsaklis, "Event Identification in Hybrid Control Systems", Proc 32nd IEEE Conference on Decision and Control, pp. 2323-2328, San Antonio, TX, Dec. 15-17, 1993.

[57] P. J. Antsaklis, "Towards Autonomous Control Systems", Proc IEEE Meditteranean Symposium on New Directions in Control Theory and Applications, pp. 274-277, Chania, Crete, Greece, June 21-23, 1993.

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

[58] P. J. Antsaklis, "Hybrid System Modeling, Analysis and Design", Abstract, ARO-NASA Workshop on Formal Models for Intelligent Control, MIT, Cambridge, MA, Sept. 30-Oct. 2. 1993.

[59] M. D. Lemmon and P. J. Antsaklis, "A Computationally Efficient Framework for Hybrid Controller Design", Proc of the 1994 IEEE/IFAC Joint Symposium on Computer-Aided Control System Design, pp 333-338, Tucson, AZ, March 7-9, 1994.

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

[61] P. J. Antsaklis, "Intelligent Control for High Autonomy in Unmanned Underwater Vehicles", Proceedings of the NSF/ISR Workshop on 'Undersea Robotics and Intelligent Control', pp. 25-32, Lisboa, Portugal, March 2-3, 1995.

[62] P. J. Antsaklis and J. C. Kantor, "Intelligent Control for High Autonomy Process Control Systems". Proceedings of 'Intelligent Systems in Process Engineering', Snowmass Village, Colorado, July 9-14, 1995. To appear.

[63] J.A. Stiver, P.J. Antsaklis and M.D. Lemmon, "Hybrid Control System Design Based on Natural Invariants", Proc of the 34th IEEE Conference on Decision and Control, New Orleans, Louisiana, Dec 13-15, 1995. To appear.

[64] M.D. Lemmon and C. Bett, "Inductive Learning of Stable Intelligent Hybrid Control Systems", Proc of the 8th Yale Workshop on Adaptive Learning Systems, pp 227-232, Yale University, June 13-15, 1994.

[65] P.T. Szymanski and M.D. Lemmon, "Interior Point Methods in the Design of Hierarchical Controllers", Proc of the 9th Intern Symposium on Intelligent Control, pp. 33-38, Columbus, OH, August 1994.

[66] M.D. Lemmon and P.T. Szymanski, "Interior Point Implementations of Alternating Minimization Training", Proc of the Conference on Neural Information Processing Systems, Denver, CO, November 1994.

[67] P.T. Szymanski and M.D. Lemmon, "A Modified Interior Point Method for Supervisory Controller Design", Proc of the 33rd IEEE Conference on Decision and Control, Lake Buena Vista, FL, Dec 14-16, 1994. To appear.

[68] M.D. Lemmon, P.T. Szymanski and C. Bett, "Optimizing Colonies of Linear Control Agents for Hybrid Control Systems", paper presented at the Workshop on Hybrid and Autonomous Systems, Cornell University, October 1994.

[69] Bett CJ and Lemmon MD H-Infinity Gain Schedule Synthesis Of Supervisory Hybrid Control Systems, to appear in DIMACS workshop on verification and control of hybrid systems, Oct. 22-25, 1995., New Brunswick, New Jersey

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

[70] Lemmon MD and Bett CJ, Hybrid Control System Design Using Robust Linear Control Agents to appear in IEEE Conference on Decision and Control, Dec. 1995, New Orleans, LA

[71] C. Lucisano and M. Lemmon Query-Based Attitude Control Of A Low Altitude Communications Satellite American Control Conference, June 1995, Seattle Washington

[72] MD Lemmon, PT Szymanski, and CJ Bett, Interior Point Competitive Learning Of Agents In Colony-Style Systems Proceedings of SPIE conference on applications and science of artificial neural networks (editors, Rogers/Ruck), Vol 2492, pg 426-437, Orlando FL, April 1995

[73] M. Lemmon and C. Bett, Inductive Learning Of Stable Intelligent Hybrid Control Systems. Proceedings of Yale Workshop on Adaptive Learning Systems, Yale University, pp 227-232, June 13-15, 1994

[74] M. Lemmon and C. Bett Direct Adaptive Stabilization Of Linear Systems Using Query-Based Protocols, 32nd IEEE Conference on Decision and Control, San Antonio, Texas, December 1993.

P.J. Antsaklis, J.A. Stiver, X. Yang, C.J. Bett, and M.D. Lemmon , "Supervisory Hybrid Control Systems in Intelligent Control," Technical Report of the ISIS (Interdisciplinary Studies of Intelligent Systems) Group, No. ISIS-95-008, Univ of Notre Dame, September 1995.

194 - IC chapter 95