On the Supremal Controllable Sublanguage in the Discrete-Event Model of Nondeterministic Hybrid Control Systems

Xiaojun Yang, Michael D. Lemmon, and Panos J. Antsaklis

Abstract—**This paper is concerned with the logical control of hybrid control systems (HCS). It is assumed that a discrete-event system (DES) plant model has already been extracted from the continuous-time plant. The problem of hybrid control system design can then be solved by applying logical DES controller synthesis techniques to the extracted DES plant. Traditional DES synthesis methods, however, are not always applicable since the extracted plant DES will often exhibit nondeterministic transitions. This paper presents an extension of certain DES controller synthesis techniques to the nondeterministic control automaton found in HCS. In particular, this paper derives a formula computing the supremal controllable sublanguage of a given specification language under the assumption that the DES plant exhibits nondeterministic transitions.**

I. Introduction

In many industrial applications, control systems can be represented by a continuous-time plant controlled by a discrete-event system. Such systems are called hybrid control systems (HCS). There has recently been considerable activity investigating the modeling and supervision of HCS's [1]–[9]. This paper uses the HCS modeling framework proposed by [7] and [8]. In this framework, a discrete-event system (DES) is extracted from the continuous-state system (CSS) plant. This DES is called the DES plant. Once such a logical model of the plant has been extracted, a DES supervisor for the system can be obtained using conventional DES controller synthesis techniques such as the Ramadge–Wonham (RW) [10] synthesis. DES plants obtained in this way, however, are often nondeterministic, and the RW synthesis does not pertain to nondeterministic DES's. Earlier work [7], [8] extended an iterative supervisor synthesis algorithm [10] to the design of HCS controllers. This paper follows a different approach in which a formula is derived to compute the supremal controllable sublanguage. The result is an extension of prior work in [11] to the class of nondeterministic DES plants found in hybrid control systems.

The remainder of this paper is organized as follows. Section II discusses the HCS modeling framework and identifies the way in which nondeterministic DES's arise in HCS. Section III introduces some necessary technical definitions which are used in the remainder of the paper. Section IV characterizes the supremal live sublanguage of a nondeterministic finite automaton associated with the DES plant. Section V derives the formula for the supremal controllable sublanguage of the DES plant. Section VI provides an example, and concluding remarks will be found in Section VII.

II. Hybrid Control Systems

Hybrid control systems arise when a continuous-state (CSS) plant is controlled by a DES controller. An important HCS modeling framework was first discussed in [7] and [8]. In this modeling framework, a simplified interface is used to facilitate communication between the CSS plant and the DES controller. A block diagram of the assumed connection is shown in Fig. 1. The CSS plant accepts a continuous reference input trajectory, $\bar{r}(t)$, and outputs the system state vector trajectory $\bar{x}(t)$. The interface consists of two subsystems: the actuator and generator. The generator transforms the plant state trajectory, $\bar{x}(t)$ into a sequence of discrete output symbols denoted as $\tilde{x}[n]$. The output sequence, $\tilde{x}[n]$, draws its symbols from a finite alphabet, $\tilde{X}$, of output symbols. The DES controller uses this sequence of output symbols as inputs and outputs a sequence of control symbols, $\tilde{r}[n]$ drawn from an alphabet $\tilde{R}$. The interface actuator transforms this sequence of control symbols into the reference trajectory, $\bar{r}(t)$. Note that in the remainder of this paper, tilded small letters will denote symbols and barred letters will denote real vectors.

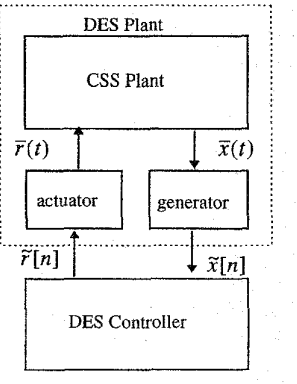

Fig. 1. Hybrid control system.

The CSS plant is assumed to be represented by a differential equation

$$\frac{d\bar{x}}{dt} = f(\bar{x}, \bar{r}) \tag{1}$$

where $\bar{x} \in \Re^n$, $\bar{r} \in \Re^m$, and $f : \Re^n \times \Re^m \rightarrow \Re^n$. The hybrid modeling framework in [7] and [8] has the interface generator issue output symbols with respect to an assumed partition of the CSS plant's state space. In particular, let $\mathcal{B}$ denote a collection of open sets which partition the CSS plant's state space. This event partition, $\mathcal{B}$, is denoted as

$$\mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_N\} \tag{2}$$

where $\mathbf{b}_i$ is an open subset of $\Re^n$ $(i = 1, \cdots, N)$. The set $\mathbf{b}_i$ is referred to as the event set. Associated with each event set is a state symbol $\tilde{p}_i$ $(i = 1, \cdots, N)$ drawn from a finite alphabet $\tilde{P}$. We will associate the output symbols $\tilde{X}$ with the boundaries of the event sets. The generator will issue the jth output symbol, $\tilde{x}_j$, when the CSS plant's state trajectory, $\bar{x}(t)$, crosses the jth boundary of an event set and enters into a new event set. The output symbol sequence, $\tilde{x}[n]$, therefore marks the entrance of the CSS plant state into event sets. Fig. 2 illustrates the situation outlined above; this figure shows three event sets with associated state symbols, $\tilde{p}_1$, $\tilde{p}_2$, and $\tilde{p}_3$. Two CSS plant trajectories (labeled A and B) are shown. Both trajectories start in the event set marked by state symbol $\tilde{p}_1$. The solid circles in the figure mark where these trajectories cross the event set boundaries. The resulting state symbol sequences $\tilde{p}[n]$ and output symbol sequences $\tilde{x}[n]$ are shown for both trajectories.

Manuscript received May 12, 1994; revised December 27, 1994 and July 8, 1995. This work was supported in part by National Science Foundation Grant MSS92-16559.

The authors are with the Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN 46556 USA.

IEEE Log Number 9415791.

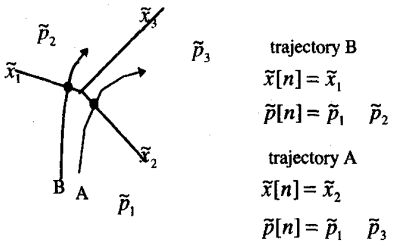

Fig. 2. Output symbols are generated with respect to a partition of the CSS plant's state space.

The output symbols are used as inputs to the DES controller. In response to these symbolic inputs, the DES controller outputs a control symbol, $\tilde{r}$. The interface actuator maps this symbol onto a specific control reference input vector, $\bar{r}$. Assume that the DES controller outputs the symbol $\tilde{r}_i$, and assume that the actuator maps this onto a control reference vector $\bar{r}_i \in \Re^m$. The CSS plant's state equation under the direction of control policy $\tilde{r}_i$ is therefore given by the differential equation $\dot{\bar{x}} = f(\bar{x}, \bar{r}_i)$. The behavior of the hybrid system can therefore be viewed as a system that switches between different constant reference inputs. The "switching" between multiple controllers is a distinctive feature of the hybrid systems found in [7] and [8]. The following results are therefore of general interest in the use and design of switched multiple controllers.

The symbolic behavior of the plant is explicitly seen from the DES controller's perspective. In particular, Fig. 1 shows that, from the controller's perspective, the plant is a discrete-event system. The combination of CSS plant and interface accepts the control symbol sequence $\tilde{r}[n]$ as an input and outputs the output symbol sequence $\tilde{x}[n]$. It is therefore possible to define a DES plant whose language represents the symbolic behavior of the CSS plant. To define such a DES plant, note that the state symbol sequence, $\tilde{p}[n]$, represents the symbolic evolution of the CSS plant over the state-space partition $\mathcal{B}$. It therefore makes sense to model the DES plant as a controlled finite automaton. The automaton states consist of the state symbols in $\tilde{P}$, the automaton's output is a language generated over the alphabet of output symbols, $\tilde{X}$, and the controlled inputs are drawn from the control symbol alphabet $\tilde{R}$.

An interesting aspect of the DES plant's behavior is that it is distinctly nondeterministic. This fact is illustrated in Fig. 2. The figure shows two trajectories generated by the same control reference input. Both trajectories originate in the same DES plant state, $\tilde{p}_1$. Precisely to which DES states are transitioned will depend upon where the CSS plant state, $\bar{x}$, is when the control symbol is issued. Fig. 2 shows that for a given control symbol, there are at least two possible plant states that the given trajectory can reach from DES plant state $\tilde{p}_1$. Nondeterminism in the DES plant therefore arises due to uncertainty in the DES states reached under a controlled transition. This type of nondeterminism is characteristic of HCS. In fact, transitions within a DES plant will usually be nondeterministic unless the boundaries of the event sets correspond to invariant manifolds of the CSS plant. This fact is significant with regard to the "logical stability" of HCS. See [9] for related discussions.

On the basis of the preceding discussion, it is apparent that the DES plant can be modeled as a nondeterministic controlled finite automaton. In particular, the DES plant, G, will be represented as the five-tuple, $G = \{\tilde{P}, \tilde{X}, \tilde{R}, \xi, \psi\}$. $\tilde{P}$ is the set of DES plant states, $\tilde{X}$ is the alphabet of output symbols, and $\tilde{R}$ is the control symbol alphabet. The enabling function is $\xi : \tilde{P} \times \tilde{R} \rightarrow P_{\tilde{X}}$ where $P_{\tilde{X}}$ is the power set of alphabet $\tilde{X}$. $\psi : \tilde{P} \times \tilde{X} \rightarrow \tilde{P}$ is the state transition function. Note that the DES plant's nondeterminism arises through the enabling function.

There have been many prior papers dealing with nondeterministic logical DES [12]–[19], and it is appropriate to place this paper's work in the context of that earlier research. Most prior works [12]–[16], [18] treat nondeterministic DES's in which one event may transition the DES from one state to one of many states. This type of nondeterminism has been dealt with in discussing event observability issues. This prior work, however, does not explicitly deal with nondeterministic control problems which arise when one control signal cannot enable events individually so that one control signal may cause several potential discrete state transitions. Cases in which the controller depends on the language generated by the DES plant do not require a separate consideration of nondeterminism as any language which can be accepted by a nondeterministic finite automaton (NFA) is also accepted by deterministic finite automaton (DFA) [20]. References [17] and [19] address issues associated with nondeterministic controllers. There has been relatively little work, however, pertaining to nondeterminism in the DES plant as it arises in HCS. In our systems we are concerned with nondeterministic control mechanism in which one control command may lead to several discrete state transitions. In our framework, the enabling function is fixed by the HCS's state space partition, $\mathcal{B}$, and the dynamics of the CSS plant. The enabling function is state dependent in the sense that it may vary with different DES plant states. This form of nondeterminism arises in a natural manner in HCS and does not appear to have been studied in detail by previous DES researchers.

III. Preliminary Definitions

$\tilde{X}^*$ denotes the set of all finite strings of the output symbols in $\tilde{X}$ including the empty string ϵ. A subset $L \subseteq \tilde{X}^*$ is called a language defined on $\tilde{X}$. The closure of a language, $\bar{L}$, is the set of all prefixes of strings in L. L is closed if $L = \bar{L}$. The closure of a string $s \in L$ generates a prefix closed sublanguage, $\bar{s}$, in $\bar{L}$. The behavior of the DES plant, G, is represented by the language, $L(G) \subset \tilde{X}^*$. Since $L(G)$ is generated by a finite automaton it is known to be prefix closed and regular.

To deal with nondeterminism induced by the controlled DES plant's enabling function, we will introduce two other automatons, G_x and G_r. The automaton G_x is a deterministic finite automaton (DFA) describing the DES plant's output behavior. The automaton G_r is a nondeterministic finite automaton (NFA) describing the plant's response to input commands. These two automatons explicitly show how nondeterminism enters the hybrid control system. G_x describes how the plant symbol behavior is generated from the plant automaton, while G_r is the control mechanism embedded in the dynamics of the controlled system. The use of these two automatons allow explicit representations for the enabling sets and provides physical insight into the hybrid control system's internal operation. That insight is instrumental in deriving the results presented below.

The automaton G_x is given by the four-tuple

$$G_x = (\tilde{P}, \tilde{X}, \psi, \tilde{p}_0) \tag{3}$$

where $\tilde{P}$ is the plant state alphabet, $\tilde{X}$ is the alphabet of output symbols, $\tilde{p}_0$ is an initial plant state, and $\psi : \tilde{P} \times \tilde{X} \rightarrow \tilde{P}$ is the state transition function. The language generated by G_x is denoted as $L(G_x) \subset \tilde{X}^*$.

The automaton, G_r, is an NFA whose language is over an "extension" of the control symbol alphabet $\tilde{R}$. $\tilde{R}$ is extended so that the dependence of the enabling function on the DES plant state can be explicitly noted. In particular, let i denote the index of the ith symbol

in $\tilde{P}$, and let j denote the index of the jth symbol in $\tilde{R}$. The extended control alphabet, $\tilde{R}_e$, consists of a set of symbols, $\tilde{r}_{ij}$ such that

$$\tilde{R}_e = \{\tilde{r}_{ij} | i: \text{ state index, and } j: \text{ control symbol index}\}. \quad (4)$$

The symbol $\tilde{r}_{ij}$ therefore means that the control symbol $\tilde{r}_j \in \tilde{R}$ was issued when the plant state was $\tilde{p}_i \in \tilde{P}$. The automaton G_r is given by the four-tuple

$$G_r = (\tilde{P}, \tilde{R}_e, \eta, \tilde{p}_0) \quad (5)$$

where $\tilde{P}$ is the plant state alphabet, $\tilde{R}_e$ is the extended control symbol alphabet, and $\tilde{p}_0$ is the initial state. The state transition function is a mapping $\eta : \tilde{P} \times \tilde{R}_e \to P_{\tilde{P}}$ where $P_{\tilde{P}}$ is the power set of alphabet $\tilde{P}$ and

$$\eta(\tilde{p}_i, \tilde{r}_{ij}) = \{\tilde{p} \in \tilde{P} \,|\, \forall \tilde{x} \in \xi(\tilde{p}_i, \tilde{r}_j), \tilde{p} = \psi(\tilde{p}_i, \tilde{x})\}. \quad (6)$$

This function maps the DES plant's current state and extended control symbol onto a collection of possible next plant states.

The following sections will need to map strings in $L(G_r)$ onto strings generated by G_x. In particular, we define this operator, $T_G : L(G_r) \longrightarrow L(G_x)$, by the relation

$$T_G(\tilde{r}_{i_0 j_0}) = \xi(\tilde{p}_{i_0}, \tilde{r}_{j_0}), \quad (7)$$

$$\begin{aligned} T_G(\tilde{r}_{i_0 j_0} \tilde{r}_{i_1 j_1} \cdots \tilde{r}_{i_m j_m}) = \{\tilde{x}_0 \tilde{x}_1 \cdots \tilde{x}_m \,|\, \tilde{x}_k \in \xi(\tilde{p}_{i_k}, \tilde{r}_{j_k}) \text{ and} \\ \psi(\tilde{p}_{i_k}, \tilde{x}_k) = \tilde{p}_{i_{k+1}} \\ \text{for all } 0 \le k \le m-1 \\ \text{and } \tilde{x}_m = \xi(\tilde{p}_{i_m}, \tilde{r}_{j_m})\} \end{aligned} \quad (8)$$

where k denotes sequence index, i_k discrete state index, and j_k control symbol index. For all $\tilde{r}_{\text{ps}} \in \tilde{R}_e^*$, $T_G(\tilde{r}_{\text{ps}})$ represents all the output symbol strings generated by $\tilde{r}_{\text{ps}}$ in G. The inverse image of a string $\tilde{x}_s \in L(G_x)$ onto the domain of T_G will be denoted as $T_G^{-1}(\tilde{x}_s)$. In particular

$$T_G^{-1}(\tilde{x}_s) = \{\tilde{r}_{\text{ps}} \in L(G_r) \,|\, \tilde{x}_s \in T_G(\tilde{r}_{\text{ps}})\}. \quad (9)$$

$T_G^{-1}(\tilde{x}_s)$ represents all the control symbol strings which possibly generate $\tilde{x}_s$ in G. If $s \in \tilde{X}^*$, then the pre-image of s under T_G will sometimes be denoted as $\tilde{r}_s \in T_G^{-1}(s)$.

IV. Supremal Live Sublanguages

Deadlocks occur in an automaton when it fails to generate any more output symbols. An automaton without deadlocks will be said to be live. Live sublanguages are important in the characterization of maximally permissive behaviors in controlled DES's. For this reason, a characterization of the live sublanguages of G_r will be needed. The following definition provides just such a characterization.

Definition 1: Consider a sublanguage L of $L(G_r)$ and a string $s \in L$. Let $\tilde{R}_{e|s}$ be a subset of control symbols in $\tilde{R}_e$ such that

$$\tilde{R}_{e|s} = \{\tilde{r}_{ij} \in \tilde{R}_e : s\tilde{r}_{ij} \in L(G_r)\}. \quad (10)$$

The set $\tilde{R}_{e|s}$ is called realizable controls from string s.

A string $s \in L$ is said to be live with respect to G_r if and only if there exists $\tilde{r}_{ij}$ in the realizable set, $\tilde{R}_{e|s}$, such that $s\tilde{r}_{ij} \in L$.

A string s in a sublanguage L of $L(G_r)$ causes deadlock if it is not live with respect to G_r.

A sublanguage L of $L(G_r)$ is said to be live with respect to G_r if it contains no strings causing deadlock.

Note that the notion of liveness defined above is somewhat more involved than similar notions found in deterministic finite automatons [21]. This additional complexity arises due to G_r's nondeterminism. In our systems, nondeterminism allows a string $s \in L(G_r)$ to terminate in several different DES plant states. To ensure that the string s is live, we need to ensure that all of these realized plant states are not deadlocked. That is precisely what the above definition accomplishes. It first uses $\tilde{R}_{e|s}$ to identify those DES plant states which are reached (realized) by a string s. It then defines liveness with regard to those realizable controls.

The principal result of this section concerns the existence and characterization of the supremal live sublanguage of $L(G_r)$. These results are used later to derive a formula for the supremal controllable sublanguage of G. The following two lemmas have well-known analogues in deterministic DES's [11]. The first lemma is stated without proof, and the second is stated with proof.

Lemma 1: If L_1, L_2 are closed live languages, then $L_1 \cup L_2$ is also a closed live language.

Lemma 2: For a given closed language $L \subseteq L(G_r)$, there exists a supremal closed and live sublanguage of L.

Proof: For all $L_1 \subseteq L$ and $L_2 \subseteq L$ such that L_1 and L_2 are closed live languages, then $L_1 \cup L_2 \subseteq L \subseteq L(G_r)$. According to Lemma 1, $L_1 \cup L_2$ is also a closed and live language. Therefore closed and live languages are closed under disjunction, and we can conclude that the disjunction of all the closed live sublanguages of L is the supremal closed and live sublanguage of L. □

Lemma 2 establishes the existence of a supremal live sublanguage of $L(G_r)$. Let L_{live} denote this supremal sublanguage. To characterize L_{live}, it will be convenient to define the following operator

$$\begin{aligned} D_{nl}(L) = \{s \in L \text{ which are deadlocked} \\ \text{with respect to } G_r\} \end{aligned} \quad (11)$$

$$D(L) = L - D_{nl}(L)\tilde{R}_e^* \quad (12)$$

$$D^2(L) = D[D(L)] \quad (13)$$

$$D_{\text{lock}}(L) = \lim_{k \to \infty} D^k(L). \quad (14)$$

The operator $D_{nl}(L)$ picks out all strings in L which cause deadlock in L (i.e., string which are not live in L). The operator $D(L)$ is therefore used to remove strings that cause deadlock in L. The following lemmas show that $D(L)$ and $D_{\text{lock}}(L)$ are closed languages. These lemmas are then used to prove the main result of this section which states that $D_{\text{lock}}(L)$ is the supremal live sublanguage, L_{live}.

Lemma 3: $L \subseteq L(G_r)$ and $L = \bar{L}$. If $D(L)$ is given by (11)–(12), then $D(L) \subseteq L$, $D(L) = \overline{D(L)}$.

Proof: Since $D(L) = L - D_{nl}(L)\tilde{R}_p^*$ and $\bar{L} = L$, $D(L)$ is closed [11], and $D(L) \subseteq L$. □

Lemma 4: $L \subseteq L(G_r)$ and $L = \bar{L}$. If D_{lock} is given by (11)–(14), then $D_{\text{lock}}(L) = \overline{D_{\text{lock}}(L)}$.

Proof: This is easily derived from Lemma 3 by mathematical induction. □

Theorem 1: $L \subseteq L(G_r)$, $L = \bar{L}$ and L is regular. If D_{lock} is given by (11)–(14), then $D_{\text{lock}}(L) = L_{\text{live}}$.

Proof: Lemma 4 guarantees that $D_{\text{lock}}(L)$ is closed. It therefore remains to show that $D_{\text{lock}}(L)$ is live and supremal. The proof is divided into two parts. The first part proves that the language is live, and the second part shows that it is supremal.

Part 1: Assume that $D_{\text{lock}}(L)$ is not live language. This means $\exists s \in D_{nl}[D_{\text{lock}}(L)]$. Since G_r is a finite automaton and by our prior definition on operator D_{lock}, there is a minimal finite n such that $D_{\text{lock}}(L) = D^n(L) = D^{n+1}(L)$. Since $D^{(n+1)}(L) = D^n(L) - D_{nl}[D^n(L)]\tilde{R}_e^*$ and $s \in D_{nl}[D_{\text{lock}}(L)]$, then $D^{(n+1)}(L)$

$= D_{\text{lock}}(L) - D_{nl}[D_{\text{lock}}(L)]\tilde{R}_e^*$. So $D^{n+1}(L) \subset D_{\text{lock}}(L)$, which contradicts with $D_{\text{lock}}(L) = D^{n+1}(L)$. Therefore the language, $D_{\text{lock}}(L)$ must be live.

Part 2: Assume that $D_{\text{lock}}(L)$ is not supremal. This means that $D_{\text{lock}}(L) \subset L_{\text{live}}$. Therefore there must exist a string $s \in L_{\text{live}}$ such that $s \notin D_{\text{lock}}(L)$. Clearly, $s \in L_{\text{live}} \subseteq L$, so there exists a finite m_0 such that $s \in D^{m_0}(L)$ and $s \notin D^{m_0+1}(L)$ with $D^{m_0+1}(L) = D^{m_0}(L) - D_{nl}[D^{m_0}(L)]\tilde{R}_p^*$. This implies that $s \in D_{nl}[D^{m_0}(L)]\tilde{R}_p^*$. Hence, there exists t s.t. $t \in \bar{s}$ such that $t \in D_{nl}[D^{m_0}(L)]$. Therefore t causes deadlock in $D^{m_0}(L)$. This means that for all k there exists i such that $\tilde{r}_{ik} \in \tilde{R}_{e|t}$ and t causes deadlock (i.e., $t\tilde{r}_{ik} \notin D^{m_0}(L)$). Let i_1 be the "state" index on this control symbol resulting in deadlock. Since $t \in \bar{s} \subseteq \bar{L}_{\text{live}}$, we can conclude that $t \in L_{\text{live}}$. More precisely, this means that for all i there exists j such that $\tilde{r}_{ij} \in \tilde{R}_{e|t}$ (realizable) and $t\tilde{r}_{ij} \in L_{\text{live}}$. Since this holds for all i, we can conclude that there exists k such that $t\tilde{r}_{i_1 k} \in L_{\text{live}}$.

The above discussion concludes that there exists string $s_0 = t\tilde{r}_{ij}$ such that $s_0 \in L_{\text{live}} \subseteq L$ and $s_0 \notin D^{m_0}(L)$. Since $s_0 \in L$, there exists $m_1 < m_0$ such that $s_0 \notin D^{m_1+1}(L)$ and $s_0 \in D^{m_1}(L)$. Reasoning as was done in the preceding paragraph, a string s_1 can be constructed such that $s_1 \in L_{\text{live}}$ and $s_1 \notin D^{m_1}(L)$. Proceeding as before, we can construct a monotone decreasing sequence of integers m_k which eventually converge to zero after N finite steps. This implies that there exists a string $s_N \in L_{\text{live}}$ such that $s_N \notin D^0(L) = L$ which is a clear contradiction since $L_{\text{live}} \subseteq L$. It is therefore concluded that L_{live} is indeed the supremal live sublanguage. □.

This theorem indicates that the supremal closed and live sublanguage of a given closed language defined on $\tilde{R}_e$ in HCS can be computed by using the D_{lock} operator. Since D_{lock} is a finite recursion on the simple D operator and since all computations involving D are basic language operations, D_{lock} can be determined using a well-defined computational procedure.

V. Formula for the Supremal Controllable Sublanguage

As described above, the DES plant differs significantly from the traditional logical DES. Since it is generally impossible to enable DES plant events individually, the traditional formal definition of DES controllability is no longer appropriate. The following definition extends the concept of a specified sublanguage's controllability to the nondeterministic DES plants arising in hybrid control systems. Early versions of this definition first appeared in [22].

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

- There exists $\tilde{r} \in \tilde{R}$ such that $\omega\xi(\psi(\tilde{p}_0, \omega), \tilde{r}) \subseteq \bar{K}$, 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 $\omega_b\xi(\psi(\tilde{p}_0, \omega_b), \tilde{r}_b) \subseteq \bar{K}$.

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. As noted earlier in our discussion on live sublanguages for G_r, however, it will also be necessary to ensure that the strings in K are generated by control symbol strings which do not generate illegal behaviors. This is precisely what the second condition ensures. It guarantees that a string in K is indeed realized by a switching control policy in our nondeterministic plant. Early definitions of DES plant controllability [22] only considered the first part of the above definition.

Since the disjunction of the controllable languages with respect to the same DES plant is still controllable with respect to the plant, we can infer that the supremal controllable sublanguage of the given language exists. The critical problem in designing the controller is to obtain the supremal controllable sublanguage of the given specification language. Let K denote the specification language of G. $L = L(G_x)$, $K \subseteq L$, $K = \bar{K}$. $K^\uparrow$ is the supremal, closed, controllable sublanguage of K. The following discussion first proves two technical lemmas. We then state and prove the main result of this paper which provides a formula for the supremal controllable sublanguage, $K^\uparrow$, of the DES plant behavior. This result is an extension of earlier formulas obtained in [11].

Lemma 5: $L_r \subseteq L(G_r)$, and $L_r = \bar{L}_r$, then $T_G(L_r) = \overline{T_G(L_r)}$.

Proof: Since $\forall\omega \in T_G(L_r)$, $\exists\tilde{r}_\omega \in L_r$, such that $\omega \in T_G(\tilde{r}_\omega)$. Since $\forall s \in \bar{\omega} - \omega$, $\exists\tilde{r}_s \in \overline{\tilde{r}_\omega}$, such that $s \in T_G(\tilde{r}_s)$. $\tilde{r}_\omega \in L_r$, $\Longrightarrow \overline{\tilde{r}_\omega} \subseteq \overline{L_r}$, $\Longrightarrow \overline{\tilde{r}_\omega} \subseteq L_r$ $(L_r = \overline{L_r})$. So, $\tilde{r}_s \in L_r$, $\Longrightarrow \underline{T_G(\tilde{r}_s)} \subseteq T_G(L_r)$. Since $s \in T(\tilde{r}_s)$, $s \in T_G(L_r)$, $\Longrightarrow T_G(L_r) = \overline{T_G(L_r)}$. □

Lemma 6: Let $T_G\{*\} = T_G\{T_G^{-1}L - [T_G^{-1}(L-K)]\tilde{R}_e^*\}$. $L = L(G_x)$ and $K \subseteq L$, then $T_G\{*\} \subseteq K$.

Proof: $\forall\omega \in T_G\{*\}$, $\exists\tilde{r}_\omega \in \{*\}$, such that $\omega \in T(\tilde{r}_\omega)$. Obviously, $\tilde{r}_\omega \in T_G^{-1}L$ and $\tilde{r}_\omega \notin [T_G^{-1}(L-K)]\tilde{R}_e^* \Longrightarrow \tilde{r}_\omega \notin T_G^{-1}(L-K)$. This implies that $T_G(\tilde{r}_\omega) \cap (L-K) = \emptyset$ (according to the meaning of T_G^{-1}). Since $L = L(G_x)$, $\Longrightarrow T_G(\tilde{r}_\omega) \subseteq L$, $\Longrightarrow T_G(\tilde{r}_\omega) \subseteq K$. Since $\omega \in T_G(\tilde{r}_\omega)$, $\Longrightarrow \omega \in K$. So, $T_G\{*\} \subseteq K$. □

Theorem 2: $K^\uparrow = T_G\{D_{\text{lock}}\{T_G^{-1}L - [T_G^{-1}(L-K)]\tilde{R}_e^*\}\}$.

Proof: Denote $\{*\} = \{T_G^{-1}L - [T_G^{-1}(L-K)]\tilde{R}_e^*\}$. First, we need to prove $T_G\{D_{\text{lock}}\{*\}\} \subseteq K$. $D_{\text{lock}}\{*\} \subseteq \{*\}$, $\Longrightarrow T_G\{D_{\text{lock}}\{*\}\} \subseteq T_G\{*\}$. $T_G\{*\} \subseteq K$ (Lemma 6), $\Longrightarrow T_G\{D_{\text{lock}}\{*\}\} \subseteq K$. This means $T_G\{D_{\text{lock}}\{*\}\}$ is the sublanguage of K.

Next, we need to prove $T_G\{D_{\text{lock}}\{*\}\}$ is closed. Since $L = L(G_x)$, it is clear $T_G^{-1}L$ is the control symbol behavior of the DES plant, $L(G_r)$. So $T_G^{-1}L$ is closed. Clearly, $T_G^{-1}L - [T_G^{-1}(L-K)]\tilde{R}_e^*$ is closed; according to Lemma 4, $D_{\text{lock}}\{*\}$ is also closed. Hence, $T_G\{D_{\text{lock}}\{*\}\}$ is closed by Lemma 5.

In the following, we will prove $T_G\{D_{\text{lock}}\{*\}\}$ is controllable. For all $\omega \in T_G\{D_{\text{lock}}\{*\}\}$, $\exists\tilde{r}_\omega \in D_{\text{lock}}\{*\}$, such that $\omega \in T_G(\tilde{r}_\omega)$. Since $D_{\text{lock}}\{*\}$ is closed, $T_G(\overline{\tilde{r}_\omega}) \subseteq T_G\{D_{\text{lock}}\{*\}\}$. Denote $\tilde{p}_i = \psi(\tilde{p}_0, \omega)$. Since $\tilde{r}_\omega \in D_{\text{lock}}\{*\}$, there exists i such that $\tilde{r}_\omega\tilde{r}_{ij} \in D_{\text{lock}}\{*\}$, where $\tilde{r}_{ij} \in \tilde{R}_e$ (Theorem 1). Clearly, $T_G(\tilde{r}_\omega\tilde{r}_{ij}) \subseteq T_G\{D_{\text{lock}}\{*\}\}$. Since $\tilde{r}_\omega\tilde{r}_{ij}$ is defined on $\tilde{R}_e$, we know that $T_G(\tilde{r}_\omega\tilde{r}_{ij}) = \omega T_G(\tilde{r}_{ij})$. So, $\omega T_G(\tilde{r}_{ij}) \subseteq T_G\{D_{\text{lock}}\{*\}\}$, and $\omega\xi[\psi(\tilde{p}_0, \omega), \tilde{r}_{ij}] \subseteq T_G\{D_{\text{lock}}\{*\}\}$ (according to the definition of T_G). Hence, $T_G\{D_{\text{lock}}\{*\}\}$ is controllable, i.e., $T_G\{D_{\text{lock}}\{*\}\} \subseteq K^\uparrow$.

Finally, we need to prove $T_G\{D_{\text{lock}}\{*\}\}$ is the supremal controllable sublanguage of K. Since we have shown that $T_G\{D_{\text{lock}}\{*\}\} \subseteq K^\uparrow$, it suffices to show that $K^\uparrow \subseteq T_G\{D_{\text{lock}}\{*\}\}$. $\forall s \in K^\uparrow$ there exists $\tilde{r}_s \in \tilde{R}_e^*$ such that $s \in T_G(\tilde{r}_s)$ and $T_G(\overline{\tilde{r}_s}) \subseteq K^\uparrow \subseteq K \subseteq L$ (according to the definition of controllable language). It is therefore concluded that $\tilde{r}_s \in T_G^{-1}(L)$. Since $T_G(\tilde{r}_s) \cap (L-K) = \emptyset$, then $\tilde{r}_s \notin T_G^{-1}(L-K)$. Hence $\tilde{r}_s \in T_G^{-1}L - T_G^{-1}(L-K)$; or in other words, $\tilde{r}_s \in \{*\}$.

Assume that $\tilde{r}_s \notin D_{\text{lock}}(*)$. Since $\tilde{r}_s \in \{*\}$, there exists m_0 such that $\tilde{r}_s \in D^{m_0}\{*\}$ and $\tilde{r}_s \notin D^{m_0+1}\{*\}$. This means that there exists $\tilde{r}_t \in \overline{\tilde{r}_s}$ such that $\tilde{r}_t \in D_{nl}\{D^{m_0}\{*\}\}$. Or in other words, there exists i for all j such that $\tilde{r}_t\tilde{r}_{ij} \notin D^{m_0}\{*\}$ where $\tilde{r}_t\tilde{r}_{ij} \in L(G_r)$. Since $\tilde{r}_t \in \overline{\tilde{r}_s}$, however, and since $T_G(\tilde{r}_t) \subseteq K^\uparrow$, there exists $t \in T_G(\tilde{r}_t) \subseteq K^\uparrow$ such that $\tilde{p}_i = \psi(\tilde{p}_0, t)$. According to the definition of controllability, there exists k such that $\tilde{r}_{ik} \in \tilde{R}_e$ and $tT_G(\tilde{r}_{ik}) \subseteq K^\uparrow$. In the same way as was done above, it can therefore be inferred that $\tilde{r}_t\tilde{r}_{ik} \in \{*\}$.

Now let $\tilde{r}_{1s} = \tilde{r}_t\tilde{r}_{ik}$. From the above results we know that $\tilde{r}_{1s} \in \{*\}$, $\tilde{r}_{1s} \notin D^{m_0}\{*\}$, and $T_G(\tilde{r}_{1s}) \subseteq K^\uparrow$. So we can assume that there exists $m_1 < m_0$ such that $\tilde{r}_{1s} \in D^{m_1}\{*\}$ and $\tilde{r}_{1s} \notin D^{m_1+1}\{*\}$. Reasoning as was done in Theorem 1, it can be concluded that there exists $\tilde{r}_{2s} \notin D^{m_1}\{*\}$, $\tilde{r}_{2s} \in \{*\}$, and

$T_G(\tilde{r}_{2s}) \subseteq K^{\uparrow}$. Proceeding inductively, we construct a monotone decreasing sequence of m_k of length N such that $m_N = 0$ and such that $\tilde{r}_{Ns} \notin D^0\{*\} = \{*\}$ and $\tilde{r}_{Ns} \in \{*\}$. This last result generates a contradiction arising out of our assumption that $\tilde{r}_s$ was not in $D_{\text{lock}}\{*\}$. It is therefore concluded that $\tilde{r}_s$ must be in $D_{\text{lock}}\{*\}$. By the definition of T_G, this means that $s \in T_G\{D_{\text{lock}}\{*\}\}$ which implies that $K^{\uparrow} \subseteq T_G\{D_{\text{lock}}\{*\}\}$. With our preceding result that $T_G\{D_{\text{lock}}\{*\}\} \subseteq K^{\uparrow}$, we can immediately infer the final formula of the theorem. □

Remark 1: This formula can be used to compute the supremal closed and controllable sublanguage of a given specification language. Since the formula consists of some basic language and automaton operations, it can be realized computationally. In addition to this, the formula can be applied to nondeterministic control DES plants found in HCS, hence the formula is helpful in the general synthesis of HCS controllers. Furthermore, the formula clearly indicates the procedure required to obtain the supremal closed and controllable sublanguage, $K^{\uparrow}$. This procedure first requires us to forbid all of the undesired control symbol sequences which may generate illegal behaviors, and then it requires us to delete those control sequences which may lead to deadlock or make the desired control unrealizable due to nondeterministic transition. So, in some cases, this formula may guide us into directly finding the supremal closed and controllable sublanguage $K^{\uparrow}$ by simply observing the structural features of L and K.

Remark 2: Since the DES plant is assumed to have been extracted from a hybrid control system, it will generally be different from that used in traditional logical DES. In the HCS framework, we need to pay attention to nondeterministic control mechanisms and active control functions. So in our framework, when an event occurs, at least one control command is allowed to be issued. It is impossible to confine the controlled logical behaviors to the desired region by simply inhibiting all illegal behaviors. In contrast, the traditional DES framework only considers that uncontrollable events cannot be forbidden from occurring. This means that it will generally be easier to handle controller synthesis for traditional DES than for DES plant arising in a hybrid control context.

VI. Example

This section presents an example of how the formula for $K^{\uparrow}$ can be used in controller synthesis for DES plants. It is assumed that the DES plant has been derived from a hybrid control system and is given in Fig. 3. The DES plant's mathematical description is given as follows

$$\begin{aligned}
\tilde{P} &= \{\tilde{p}_0, \tilde{p}_1, \tilde{p}_2, \tilde{p}_3, \tilde{p}_4\},\\
\tilde{X} &= \{a, b, c, d, e\},\\
\tilde{R} &= \{\tilde{r}_1, \tilde{r}_2, \tilde{r}_3\}.\\
\psi(\tilde{p}_0, a) &= \tilde{p}_1,\ \psi(\tilde{p}_0, d) = \tilde{p}_4,\\
\psi(\tilde{p}_1, b) &= \tilde{p}_2,\ \psi(\tilde{p}_1, d) = \tilde{p}_4,\\
\psi(\tilde{p}_2, c) &= \tilde{p}_3,\ \psi(\tilde{p}_2, d) = \tilde{p}_4,\\
\psi(\tilde{p}_3, b) &= \tilde{p}_2,\ \psi(\tilde{p}_3, d) = \tilde{p}_4,\\
\psi(\tilde{p}_4, e) &= \tilde{p}_0.\\
\xi(\tilde{p}_0, \tilde{r}_1) &= \{a, d\},\ \xi(\tilde{p}_0, \tilde{r}_2) = a,\ \xi(\tilde{p}_0, \tilde{r}_3) = d,\\
\xi(\tilde{p}_1, \tilde{r}_1) &= \{b, d\}, \xi(\tilde{p}_1, \tilde{r}_2) = d,\ \xi(\tilde{p}_1, \tilde{r}_3) = b,\\
\xi(\tilde{p}_2, \tilde{r}_2) &= \{c, d\},\ \xi(\tilde{p}_2, \tilde{r}_3) = c,\\
\xi(\tilde{p}_3, \tilde{r}_1) &= \{b, d\},\ \xi(\tilde{p}_3, \tilde{r}_2) = d,\ \xi(\tilde{p}_3, \tilde{r}_3) = d,\\
\xi(\tilde{p}_4, \tilde{r}_3) &= e.\\
L(G) &= \overline{((de)^*(ade + ab(cb)^*de + ab(cb)^*cde)^*)^*}.
\end{aligned}$$

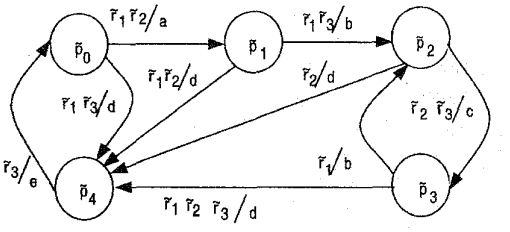

Fig. 3. DES model.

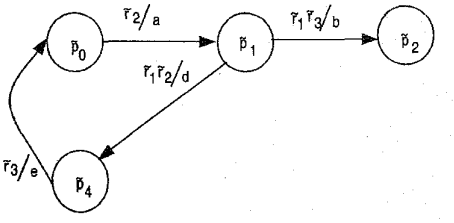

Fig. 4. Discarding undesired control sequences.

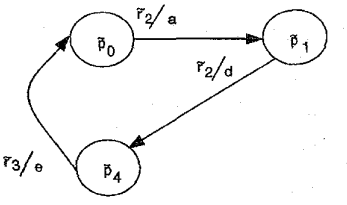

Fig. 5. Realizable behaviors.

Here, ψ, ξ are partial transition function and enabling function, respectively, of the DES plant.

Suppose we want to control the DES plant so that it generates the state paths $\overline{(\tilde{p}_0\tilde{p}_1\tilde{p}_4 + \tilde{p}_0\tilde{p}_1\tilde{p}_2\tilde{p}_4)}^*$. That is, given the control specification K, $K = \overline{(ade + abde)}^*$. K is not controllable, since, in $\tilde{p}_2$, there exists no $\tilde{r} \in R$ preventing the DES traces from entering $\tilde{p}_3$. According to the formula proposed in Theorem 2, the supremal controllable sublanguage of K is $K^{\uparrow} = \overline{(ade)}^*$. Since the control symbol strings $\tilde{r}_{s1} = (\tilde{r}_{01} + \tilde{r}_{02})(\tilde{r}_{11} + \tilde{r}_{13})(\tilde{r}_{22} + \tilde{r}_{23})$ and $\tilde{r}_{s2} = (\tilde{r}_{01} + \tilde{r}_{03})$ generate the forbidden output symbol strings abc and d we should try to eliminate the sublanguage $(\tilde{r}_{s1} + \tilde{r}_{s2})\tilde{R}_e^* \subseteq L(G_r)$. This is done by removing or relabeling arcs in the DES plant of Fig. 3. The reduced automaton resulting from this operation is shown in Fig. 4. The deadlock problem now has to be addressed to get a realizable sequence of control symbols. Since the language, $\tilde{r}_{02}(\tilde{r}_{11} + \tilde{r}_{13})$, produces deadlocks in the state $\tilde{p}_2$, we need to discard the arcs leading to this deadlocked state. The reduced automaton resulting from this operation is shown in Fig. 5. The remaining automaton will generate the supremal controllable language $K^{\uparrow}$.

VII. Conclusion

This paper derived a formula calculating the supremal controllable sublanguage for a class of nondeterministic control discrete-event systems arising in the supervision of hybrid control systems. The results show that the HCS modeling framework introduced in [7] and [8] supports the synthesis of discrete controllers in HCS. The significance of this formula is that it provides a closed form expression

for the supremal controllable sublanguage which provides significant insight into how HCS supervisors can be efficiently designed.

REFERENCES

[1] P. J. Antsaklis, M. D. Lemmon, and J. A. Stiver, "Learning to be autonomous: Intelligent supervisory control," to appear in *Intelligent Control: Theory and Control*. Piscataway, NJ: IEEE, 1995.

[2] A. Benveniste and P. Le Guernic, "Hybrid dynamical systems and the signal language," *IEEE Trans. Automat. Contr.*, vol. 35, no. 5, pp. 535–546, May 1990.

[3] A. Gollu and P. Varaiya, "Hybrid dynamical systems," in *Proc. 28th Conf. Decis. Contr.*, Tampa, FL, Dec. 1989, pp. 2780–2712.

[4] L. Holloway and B. Krogh, "Properties of behavioral models for a class of hybrid dynamical systems," in *Proc. 31st Conf. Decis. Contr.*, Tucson, AZ, Dec. 1992, pp. 3752–3757.

[5] W. Kohn and A. Nerode, "Multiple agent autonomous hybrid control systems," in *Proc. 31st Conf. Decis. Contr.*, Tucson, AZ, Dec. 1992, pp. 2956–2966.

[6] P. J. Ramadge, "On the periodicity of symbolic observations of piecewise smooth discrete-time systems," *IEEE Trans. Automat. Contr.*, vol. 35, no. 7, pp. 807–812, 1990.

[7] J. A. Stiver, P. J. Antsaklis, and M. D. Lemmon, "A logic DES approach to the design of hybrid control systems," Univ. Notre Dame, Notre Dame, IN, Tech. Rep. of the ISIS Group ISIS-94-011, Oct. 1994, revised May 1995, to appear in *Mathematical and Computer Modeling: Special Issue on Discrete Event Systems*.

[8] J. Stiver, "Analysis and design of hybrid control systems," Ph.D. dissertation, Dept. of Elec. Eng., Univ. Notre Dame, May 1995.

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

[10] P. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Contr. Optim.*, vol. 25, no. 1, pp. 206–230, Jan. 1987.

[11] R. D. Brandt, V. Gary, R. Kumar, F. Lin, S. I. Marcus, and W. M. Wonham, "Formulas for calculating supremal controllable and normal sublanges," *Syst. Contr. Lett.*, vol. 15, no. 1, pp. 111–117, Jan. 1990.

[12] P. J. Ramadge, "Observability of discrete event systems," in *Proc. 25th Conf. Decis. Contr.*, Athens, Greece, Dec. 1986, pp. 1108–1112.

[13] C. M. Ozveren and A. S. Willsky, "Invertibility of discrete event dynamic systems," *Math. Contr., Sig., Syst.*, vol. 5, no. 4, pp. 365–390, 1992.

[14] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya, "Supervisory control of discrete-event processes with partial observations," *IEEE Trans. Automat. Contr.*, vol. 33, no. 3, pp. 249–260, 1988.

[15] H. Cho and S. I. Marcus, "On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation," *Math. Contr., Sig., Syst.*, vol. 2, no. 1, pp. 47–69, 1989.

[16] T. Ushio, "A necessary and sufficient condition for the existence of finite state supervisors in discrete event systems," *IEEE Trans. Automat. Contr.*, vol. 38, no. 1, pp. 135–138, 1993.

[17] Y. Li and W. M. Wonham, "Strict concurrency and nondeterministic control of discrete-event systems," in *Proc. 28th Conf. Decis. Contr.*, Tampa, FL, Dec. 1989, pp. 2731–2736.

[18] M. A. Shaymann and R. Kumar, "Supervisory control of nondeterministic discrete event dynamical systems," in *Proc. 33rd Conf. Decis. Contr.*, San Antonio, TX, Dec. 1993, pp. 1188–1193.

[19] C. H. Golaszewski and P. J. Ramadge, "Control of discrete event processes with forced events," in *Proc. 26th Conf. Decis. Contr.*, Los Angeles, CA, Dec. 1987, pp. 247–251.

[20] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Language, and Computation*. Reading, MA: Addison-Wesley, 1979.

[21] R. Kumar, V. K. Garg, and S. I. Marcus, "On supervisory control of sequential behaviors," *IEEE Trans. Automat. Contr.*, vol. 37, no. 12, pp. 1978–1985, Dec. 1992.

[22] J. A. Stiver and P. J. Antsaklis, "On the controllability of hybrid control systems," in *Proc. 32nd Conf. Decis. Contr.*, San Antonio, TX, Dec. 1993, pp. 294–299.

A Class of Multilinear Uncertain Polynomials to Which the Edge Theorem Is Applicable

Weining Chen and Ian R. Petersen

***Abstract*—This paper gives a class of multilinear uncertain polynomials to which the Edge Theorem is applicable. In the paper, a notion of "symmetric uncertainty structure" is introduced. The main result is as follows: If a multilinear uncertain polynomial has a symmetric uncertainty structure and the uncertain parameters are nonoverlapping, then the corresponding value sets will be convex polygons. This means that the use of the zero exclusion condition to test the robust stability of such uncertain polynomials is feasible since we need only to calculate the extreme points of the convex polygons. Furthermore, a version of the Edge Theorem can be used to test the robust stability of such uncertain polynomials.**

I. INTRODUCTION

The presence of uncertainties in control systems gives rise to robust control problems. Such system uncertainties are due to modeling errors, simplifying assumptions in the modeling process, and parameter variations. A commonly encountered problem in robust control theory is the robust stability problem for uncertain linear time-invariant systems. The connection between the stability of a linear time-invariant system and its characteristic polynomial leads to a corresponding problem: The robust stability problem for uncertain polynomials. Moreover, many robust performance problems can be transformed into equivalent robust stability problems; e.g., see [1]. Underlying all such robust stability problems is the following question: How do the roots of a polynomial depend on its coefficients? A seminal result relating to this problem is Kharitonov's Theorem; see [2].

The study of polynomial robustness results related to Kharitonov's Theorem is currently an active area of research in the robust control field. Research in this area began with Kharitonov's famous paper published in 1978; see [2]. Kharitonov's Theorem is a powerful result which enables the robust stability of an interval polynomial to be determined by checking only four fixed polynomials. For more general polynomial families, however, such as a polytope of polynomials, Kharitonov-like extreme point results do not hold. In this case, the celebrated Edge Theorem due to Bartlett *et al.* is an important result for polytopes of polynomials; see [3]. The Edge Theorem states that a polytope of polynomials is robustly $\mathcal{D}$-stable if and only if all edges of the polytope of polynomials are $\mathcal{D}$-stable (see [1] for definition of $\mathcal{D}$-stability).

Although Kharitonov's Theorem and the Edge Theorem are powerful theoretical results, the uncertainty structures which arise in typical applications usually do not allow these results to be applied directly. Hence, it is desired to derive new results which are applicable to wider classes of uncertain polynomials. Alternatively, the approach taken in this paper is to find an enlarged polynomial family to which the Edge Theorem is applicable.

Manuscript received August 3, 1994; revised July 6, 1995. This work was supported in part by the Australian Research Council.

W. Chen is with JRCASE, Macquarie University, Sydney NSW 2109, Australia.

I. R. Petersen is with the Department of Electrical Engineering, Australian Defence Force Academy, Campbell, 2600, Australia.

IEEE Log Number 9415790.