ACKNOWLEDGMENT

The authors are grateful to the referees and the Associate Editor for their valuable comments and suggestions.

REFERENCES

[1] J.-Q. Hu, P. Vakili, and G.-X. Yu, "Optimality of hedging point policies in the production control of failure prone manufacturing systems," *IEEE Trans. Automat. Contr.*, vol. 39, pp. 1875–1880, 1994.

[2] J.-Q. Hu and D. Xiang, "Optimal control for systems with deterministic production cycles," *IEEE Trans. Automat. Contr.*, vol. 40, pp. 782–786, Apr. 1995.

[3] J. W. Cohen, *The Single Server Queue.* Amsterdam, The Netherlands: North-Holland, 1982.

[4] J.-Q. Hu, "Production rate control for failure-prone production systems with no backlog permitted," *IEEE Trans. Automat. Contr.*, vol. 40, pp. 291–295, Feb. 1995.

[5] J. G. Kimemia and S. B. Gershwin, "An algorithm for the computer control of production in flexible manufacturing systems," *IIE Trans.*, vol. 15, pp. 353–362, 1983.

[6] R. Akella and P. R. Kumar, "Optimal control of production rate in a failure prone manufacturing system," *IEEE Trans. Automat. Contr.*, vol. 31, pp. 116–126, Feb. 1986.

[7] T. Bielecki and P. R. Kumar, "Optimality of zero-inventory policies for unreliable manufacturing systems," *Ops. Res.*, vol. 36, no. 4, pp. 532–541, July/Aug. 1988.

Online Supervisor Synthesis for Partially Observed Discrete-Event Systems

Joseph H. Prosser, Moshe Kam, and Harry G. Kwatny

***Abstract*—A partial information supervisor that generates a class of closed controllable and observable sublanguages of a specified "legal" language is presented. This supervisor has the following features: 1) it can be implemented *online* (i.e., the disabled event set need only be computed once upon each event observation); 2) the computations of the disabled event set can be performed in $O(mn)$ worst case complexity, where n is the number of states in the legal language generator and m is the number of events; 3) an online supervisor presented previously by Heymann and Lin (1993) is a special case of the new supervisor; and 4) all the languages generated by the new supervisor contain the supremal closed controllable and normal (supCCN) sublanguage of the legal language (in fact, they contain a language developed by Fa *et al.* (1993) that was shown to contain the supCCN sublanguage).**

***Index Terms*—Discrete-event systems, online control, partial observation, supervisory control.**

I. INTRODUCTION

The design of supervisors for partially observed discrete-event systems requires the properties of *language controllability* and *language observability.* It has been shown that if a language K is a prefix-closed sublanguage of the plant language, then a supervisor exists such that the closed-loop language is equal to K if and only if K is both controllable and observable.

When a specified closed-loop language is not both controllable and observable, it is desired to approximate it in some sense by a controllable and observable sublanguage (thereby ensuring that a supervisor for the sublanguage can be realized.) Often, supervisors are computed that generate a (controllable and observable) subset of a specified "legal" language L. Since many such sublanguages can exist (and be incomparable in terms of set inclusion) it is of interest to find them, classify them, and identify their properties. Here, we assume that the legal language L is closed and regular and generate a new class of (not necessarily regular) closed controllable and observable sublanguages of L. This class of languages has, as two of its members, languages that were previously proposed in [9] and [11].

One of the most important properties of a potential closed-loop language is the computational effort required for synthesis of the corresponding supervisor. Until recently, the computational complexity of synthesizing partial-information supervisors has limited their practicality. For a nonregular closed-loop language, the number of supervisor states is (in the worst case) infinite, even when both the legal language and the plant language are regular. The reason is that a generator for a nonregular language can have an infinite number of states.

In [9], it was suggested that some of the complexity limitations can be bypassed by an *online* supervisor synthesis procedure. Rather than computing the supervisor *a priori* (i.e., determining *a priori* which controllable events to disable for every possible sequence of event observations) the disabled event set can sometimes be computed online just after the occurrence of observed events. The only assumption needed is that a minimum time-period elapse between event occurrences in the system, and that this period be longer than the online computation time. These limitations are reasonable in systems whose events do not occur very frequently, or where computational resources are plentiful.

The computational complexity of online synthesis procedures is expressed in terms of their stepwise complexity, i.e., the number of computations required each time an event is observed. Some existing procedures (e.g., [9] and [11]) have been shown to have complexity $O(mn)$, where n is the number of states in the generator of the legal language and m is the number of events.

Potential closed-loop languages are also evaluated in terms of their relative "size" when compared (in terms of set inclusion) to existing sublanguages or classes of sublanguages. One of the first approaches to finding a closed controllable and observable sublanguage uses the property of *language normality* [2]. Language normality implies language observability, and normality (like controllability) is closed under union. Hence there exists a supremal closed controllable and normal sublanguage of L, which we shall denote $\sup CCN(L)$. Studies of normal sublanguages can be found in [3]–[7].

One common approach to supervisor synthesis has been used to find closed controllable and observable sublanguages of L that contain the supremal closed controllable and normal sublanguage. The process starts with a (nonempty) closed controllable (but not necessarily observable) sublanguage $K \subseteq L$ and uses K to generate another closed controllable language that is also an observable sublanguage of K. This process was used to produce sublanguages

Manuscript received December 30, 1996. This work was supported in part by the National Science Foundation under Grant ECS-9216588 and by the Electric Power Research Institute under Contract RP-8030-01.

J. H. Prosser and M. Kam are with the Data Fusion Laboratory, Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA 19104 USA (e-mail: kam@lorelei.ece.drexel.edu).

H. G. Kwatny is with the Department of Mechanical Engineering and Mechanics, Drexel University, Philadelphia, PA 19104 USA.

Publisher Item Identifier S 0018-9286(98)07532-1.

in [8]–[11]. One such language we denote as $\Omega(K)$, a language discovered independently by the authors of [8]–[10] which was shown to satisfy $\sup CCN(K) \subseteq \Omega(K)$. Under the assumption that K is regular, a language that contains $\Omega(K)$ was synthesized in [9] and is denoted $L(S_M/G)$ where S_M is a supervisor for the plant G. Another language that contains $\Omega(K)$ (and is incomparable to $L(S_M/G)$ in the sense of set inclusion) was presented in [11] and is denoted $L(S_3/G)$. Online techniques have been used to synthesize supervisors for $\Omega(K)$ and $L(S_M/G)$ in [9] and [12], and for $L(S_3/G)$ in [11]. All of these techniques require the assumption that K is closed, controllable, and regular.

The main result of the present study is a generalization of an online supervisor synthesis procedure given in [9], producing a *class* of closed controllable and observable sublanguages of a given regular closed and controllable language $K \subseteq L$. Each element of the new class contains the language $\Omega(K)$ [which itself contains $\sup CCN(K)$]. Under the assumption that L is regular, if we set $K = L^{\uparrow}$ (where $L^{\uparrow}$ is the supremal closed and controllable sublanguage of the regular language L) then every member of the class also contains $\sup CCN(L)$. Furthermore, the previously proposed languages $L(S_M/G)$ and $L(S_3/G)$ are members of the class. This class of languages can contain nonregular languages.

The rest of this paper is organized as follows. Section II reviews the relevant theory of supervisory control and describes the language $\Omega(K)$, where K is an (arbitrary) regular closed controllable sublanguage of L. In Section III we present the assumptions and the main results of this paper. We present some examples in Section IV. The proof of our main result is given in Appendix A.

II. Preliminaries

The system to be controlled is modeled by a deterministic finite automaton (DFA) or generator $G = (Q, \Sigma, \delta, q_0)$, where Q is a set of states containing the initial state q_0, Σ is a nonempty set of events such that the null event $\epsilon \notin \Sigma$, and δ is a transition function (a partial function) mapping $\Sigma \times Q$ into Q. The set Σ^* is the set of all strings made from concatenating any number of events in Σ, defined as $\Sigma^* = \{\epsilon\} \cup \Sigma \cup \Sigma\Sigma \cup \cdots$.

The plant's transition function is extended in its domain over $\Sigma^* \times Q$ by assigning for all $q \in Q$, $\delta(\epsilon, q) := q$, and $\delta(s\sigma, q) := \delta(\sigma, \delta(s, q))$ if it is defined. A *language* is any subset of Σ^*. A language L is said to be *prefix-closed* or just *closed* if every prefix of every string in L is also a member of L. The prefix-closure of a language L, denoted $\overline{L}$, is the set of prefixes of every string in L.

Corresponding to the DFA G we define a language $L(G)$ (called the plant language) to be the set $L(G) := \{s | s \in \Sigma^* \wedge \delta(s, q_0) \text{ is defined}\}$. The language $L(G)$ is closed, and we say that the plant G *generates* $L(G)$. If $L \subseteq L(G)$ we say that L is a sublanguage of $L(G)$ or just that L is a sublanguage.

We partition Σ into two subsets: Σ_c, the set of controllable events, and $\Sigma_{uc} = \Sigma - \Sigma_c$, the set of uncontrollable events. A *supervisor* is a function γ: $L(G) \rightarrow 2^{\Sigma_c}$. Under the event disabling action of the supervisor γ the plant G generates the *closed-loop language* that is a closed sublanguage of $L(G)$. The closed-loop language generated by the supervisor γ acting on the plant G is denoted $L(G, \gamma)$, defined constructively using: 1) $\epsilon \in L(G, \gamma)$ and ii)

$$[s \in L(G, \gamma) \wedge s\sigma \in L(G) \wedge \sigma \notin \gamma(s)] \Leftrightarrow s\sigma \in L(G, \gamma). \quad (1)$$

If $L(G, \gamma) = L$ we say that the supervisor γ generates L. From the above definition of $L(G, \gamma)$, it can be seen that the function γ need not be defined for all $s \in L(G)$. It is sufficient to define it for all $s \in N$ so long as $L(G, \gamma) \subseteq N \subseteq L(G)$. We call γ a supervisor as long as it is defined over a sufficient subset of $L(G)$ so that the closed-loop language can be computed.

A prefix-closed language $L \subseteq L(G)$ is *controllable* if $(\forall s \in L, \sigma \in \Sigma_{uc})[s\sigma \in L(G) \Rightarrow s\sigma \in L]$. If L is a closed sublanguage of $L(G)$ then a supervisor exists such that $L(G, \gamma) = L$ if and only if L is controllable [1]. Every closed language $L \subseteq L(G)$ contains a supremal closed controllable sublanguage [14]. The supremal closed controllable sublanguage of L is denoted $L^{\uparrow}$.

A mask [5] is a function M: $\Sigma \rightarrow \Lambda \cup \{\epsilon\}$, where $\epsilon \notin \Lambda$ and M is defined for all $\sigma \in \Sigma$. The set Λ is another event set, and events in Λ are called *observed events.* The mask M is extended in its domain over strings by assigning $M(\epsilon) := \epsilon$ and $(\forall s \in \Sigma^*, \sigma \in \Sigma) M(s\sigma) := M(s)M(\sigma)$. The mask function is extended in its domain over sets of events and sets of strings in the natural way.

We define the *restricted inverse mask function* M_P^{-1} as a map M_P^{-1}: $2^{\Lambda^*} \rightarrow 2^P$, where P is a restricted domain of M. Let $(\forall \Delta \subseteq \Lambda^*)[M_P^{-1}(\Delta) := \{s \in P | M(s) \in \Delta\}]$. The set $M_\Sigma^{-1}(\{\lambda\})$ is the set of events in Σ whose mask value is λ. Similarly, the set $M_{\Sigma^*}^{-1}(\{l\})$ is the set of strings in Σ^* whose mask value is equal to l. The *unobservable* events, denoted Σ_ϵ, are those whose mask value is equal to ϵ. These are given by $\Sigma_\epsilon = M_\Sigma^{-1}(\{\epsilon\})$. For simplicity of notation we sometimes omit parentheses and braces and write $M\sigma$ for $M(\sigma)$, $ML(G)$ for $M(L(G))$, and $M^{-1}l$ for $M_{\Sigma^*}^{-1}(\{l\})$ wherever its meaning is clear from context.

A *partial information supervisor* is a function γ_M: $ML(G) \rightarrow 2^{\Sigma_c}$. The closed-loop language of a partial information supervisor is computed by assigning $(\forall s \in L(G))\gamma(s) := \gamma_M(Ms)$ then computing $L(G, \gamma)$ as in (1). The function γ_M actually need not be defined for all $l \in ML(G)$ but only on a sufficient subset so that the closed-loop language can be computed. We shall always use the subscript M for partial information supervisors so we are permitted to write $L(G, \gamma_M)$.

A prefix-closed language $L \subseteq L(G)$ is *observable* (with respect to M) if

$$(\forall s, s' \in L, \sigma \in \Sigma)([Ms = Ms' \wedge s\sigma \in L \wedge s'\sigma \in L(G)] \\ \Rightarrow s'\sigma \in L). \quad (2)$$

A partial information supervisor that generates the prefix-closed sublanguage L exists if and only if L is both controllable and observable [2].

A property that found use in the synthesis of sublanguages of a given language is *normality* [15]. A prefix-closed language $L \subseteq L(G)$ is *normal* (with respect to M) if for all $s \in L(G)$ and $s' \in L$, $Ms = Ms'$ implies that $s \in L$. If a closed sublanguage is normal, then it is observable. Since normality is closed under union, for each closed sublanguage L there exists a supremal closed controllable and normal sublanguage of L [2]. This language is denoted $\sup CCN(L)$.

If the language $K \subseteq L(G)$ is closed and controllable (and nonempty), then there is a closed controllable and observable sublanguage of K denoted $\Omega(K)$ that contains $\sup CCN(K)$ [8]. We now synthesize a partial information supervisor that generates $\Omega(K)$. By assumption, there exists a supervisor γ such that $L(G, \gamma) = K$. Let γ be such a supervisor. Assume without loss of generality that $(\forall s \in K)(\sigma \in \gamma(s) \Rightarrow s\sigma \in L(G))$. In this case, we have that $(\forall s \in K)\gamma(s) = \{\sigma \in \Sigma_c | s\sigma \in L(G) - K\}$. We extend the domain of the function γ to sets of strings by assigning $(\forall P \subseteq K)\overline{\gamma}(P) := \bigcup_{s \in P} \gamma(s)$.

Now define $(\forall l \in MK)[\gamma_M^\Omega(l) := \overline{\gamma}(M_K^{-1}l)]$. Thus $\gamma_M^\Omega(l)$ is the union of $\gamma(s)$ over all $s \in K$ that satisfy $Ms = l$. We have that $\Omega(K) = L(G, \gamma_M^\Omega)$ [8]–[10].

III. An Online Supervisor for a Class of Legal Sublanguages

We assume that the legal sublanguage L is closed and regular. We develop a procedure for generating closed controllable and observable (but not necessarily regular) sublanguages of L. We assume the existence of some nonempty regular closed controllable sublanguage of L, which we denote K. Then K and $\Omega(K)$ are both closed regular languages. We generate languages that contain $\Omega(K)$ and are contained in K. We remark that we can set K to be any regular (nonempty) closed controllable sublanguage of L, e.g., we can choose $K = L^{\uparrow}$.

The controllability of K allows us to assume the existence of a supervisor γ such that $L(G, \gamma) = K$ and $(\forall s \in K)[\sigma \in \gamma(s) \Rightarrow s\sigma \in L(G)]$. (Hence $\gamma(s)$ contains no "extra" events.) Since K is regular, let $D = (X, \Sigma, \xi, x_0)$ be a DFA such that $L(D) = K$ with D *accessible*. D is accessible when for all $x \in X$, there exists an $s \in L(D)$ such that $\xi(s, x_0) = x$. Furthermore, we choose D so that D *refines* G [14]. In this case, the supervisor γ can be given equivalently by a *disable map* $\gamma_D\colon X \rightarrow 2^{\Sigma_c}$, where we define $(\forall x \in X)[\gamma_D(x) := \gamma(s)\colon \xi(s, x_0) = x]$.

We now extend the domains of γ_D and ξ to ease notation. The disable map γ_D is extended in its domain over subsets of X. We define the *extended disable map* $\overline{\gamma}_D\colon 2^X \rightarrow 2^{\Sigma_c}$ using the union of $\gamma_D(x)$ over $x \in X$.

For simplicity of notation we sometimes omit parentheses and write $\overline{\gamma}_D P$ for $\overline{\gamma}_D(P)$. The *extended transition function* is similarly defined by assigning

$$(\forall J \subseteq \Sigma^*,\, P \subseteq X)\overline{\xi}(J, P) := \{x \in X | (\exists\, s \in J,\, x' \in P)\, x = \xi(s, x')\}. \tag{3}$$

The partial information supervisor γ_M^{Ω} can now be equivalently given by $\gamma_M^{\Omega}(l) = \overline{\gamma}_D\overline{\xi}(M^{-1}l, \{x_0\})$.

Our main theorem (Theorem 1) places a sufficient condition on the events that a partial information supervisor disables, so that the resulting closed-loop language contains $\Omega(K)$ and is contained in K. A class of languages is generated by choosing such supervisors that meet the sufficient condition. We show several ways of choosing supervisors in the propositions following the theorem. The theorem uses the definition of an *unobserved reach function* (denoted $\hat{\xi}$) that provides information about the states of D (the DFA that generates K) that can be reached via strings of the closed-loop language. The unobserved reach function takes into account the event-disabling actions of the partial information supervisor, and is used, along with the disable map $\overline{\gamma}_D$, to determine those events that: 1) must be disabled to ensure that the closed-loop language is a sublanguage of K and 2) must not be disabled to ensure that the closed-loop language contains $\Omega(K)$.

Theorem 1: Let $\gamma_M\colon ML(G) \rightarrow 2^{\Sigma_c}$ be a partial information supervisor. Let $\hat{\xi}(\epsilon) := \{x_0\}$ and

$$(\forall l\lambda \in MK)\hat{\xi}(l\lambda) := \overline{\xi}(M_{\Sigma}^{-1}\lambda - \gamma_M(l), \hat{\xi}_+(l)) \tag{4}$$

where $\hat{\xi}_+(l) = \overline{\xi}([\Sigma_\epsilon - \gamma_M(l)]^*, \hat{\xi}(l))$. Then the following:

$$\forall\, l \in ML(G, \gamma_M)[\overline{\gamma}_D\hat{\xi}_+(l) \subseteq \gamma_M(l) \subseteq \overline{\gamma}_D\overline{\xi}(\Sigma_\epsilon^*, \hat{\xi}(l))] \tag{5}$$

implies that $\Omega(K) \subseteq L(G, \gamma_M) \subseteq K$.

The proof of the theorem is given in Appendix I. The functions $\hat{\xi}_+$ and $\hat{\xi}$ have physical meaning related to the language K and its DFA D. As shown in Lemma A.1, if a string s is in the closed-loop language (and (5) is satisfied) then the corresponding state $\xi(s, x_0)$ is a member of $\hat{\xi}_+(Ms)$. In fact (as demonstrated in the proof) if the last event of s is observable, then the corresponding state $\xi(s, x_0)$ is a member of $\hat{\xi}(Ms)$. The functions $\hat{\xi}_+$ and $\hat{\xi}$ take into account the event-disabling action of γ_M, hence it is also true that for $l \in ML(G, \gamma_M)$, $x \in \hat{\xi}_+(l)$ implies that there exists a string $s \in L(G, \gamma_M)$ such that $Ms = l$ and $\xi(s, x_0) = x$.

We now present a series of propositions that show how to select the supervisor γ_M so that the sufficient condition (5) is satisfied. In each proposition we select the set $\gamma_M(l)$ for each $l \in ML(G, \gamma_M)$, and it is assumed that $\gamma_M(l')$, where l' is a (strict) prefix of l, has already been selected. The following proposition uses an "initial guess" of $\gamma_M(l)$ in order to compute one that satisfies (5).

Proposition 1.1: For each $l \in ML(G, \gamma_M)$, let $\Gamma_l' \subseteq \Sigma_c$. Let

$$\Gamma_l = \Gamma_l' \cap \overline{\gamma}_D\overline{\xi}([\Sigma_\epsilon - \Gamma_l']^*, \hat{\xi}(l)) \tag{6}$$
$$\gamma_M(l) = \overline{\gamma}_D\overline{\xi}([\Sigma_\epsilon - \Gamma_l]^*, \hat{\xi}(l)). \tag{7}$$

Then $\Omega(K) \subseteq L(G, \gamma_M) \subseteq K$.

Proof: We have that $(\forall l \in ML(G, \gamma_M))\overline{\gamma}_D\overline{\xi}([\Sigma_\epsilon - \Gamma_l]^*, \hat{\xi}(l)) \subseteq \overline{\gamma}_D\overline{\xi}(\Sigma_\epsilon^*, \hat{\xi}(l))$, so all that remains to be shown is that $\overline{\gamma}_D\hat{\xi}_+(l) \subseteq \gamma_M(l)$. It is clear from the definition of Γ_l that $\Gamma_l \subseteq \Gamma_l'$, hence $\overline{\gamma}_D\overline{\xi}([\Sigma_\epsilon - \Gamma_l']^*, \hat{\xi}(l)) \subseteq \gamma_M(l)$. But it is also true that $\Gamma_l \subseteq \overline{\gamma}_D\overline{\xi}([\Sigma_\epsilon - \Gamma_l']^*, \hat{\xi}(l))$, hence $\Gamma_l \subseteq \gamma_M(l)$. It follows by the definition of γ_M and $\hat{\xi}_+$ that $\overline{\gamma}_D\hat{\xi}_+(l) \subseteq \gamma_M(l)$. By Theorem 1, $\Omega(K) \subseteq L(G, \gamma_M) \subseteq K$. □

The initial guess Γ_l' can (for each l) be any subset of Σ_c, and it can be interpreted as a set of "preferred events" to disable. Although the set $\gamma_M(l)$ can contain events not in Γ_l' [and Γ_l' can contain events not in $\gamma_M(l)$], these two sets share a common subset, namely Γ_l. The use of the set Γ_l' to generate $\gamma_M(l)$ [as in (7)] guarantees the right-hand containment in (5). To construct $\gamma_M(l)$ we require two evaluations of the function $\overline{\xi}$ and two evaluations of the function $\overline{\gamma}_D$. The computation of $\overline{\xi}$ takes in the worst case $|\Sigma_\epsilon||X|$ steps, and the computation of $\overline{\gamma}_D$ takes at most $|X|$ steps. Hence the complexity of the algorithm presented in Proposition 1.1 is $O(|\Sigma_\epsilon||X|)$.

The following ways of selecting γ_M (Propositions 1.2 and 1.3) are special cases of Proposition 1.1; they show how some languages proposed in the past can be generated. In both cases, the set Γ_l' is chosen independently of l.

Proposition 1.2: For each $l \in ML(G, \gamma_M)$, let $\Gamma_l' = \Sigma_c$ in Proposition 1.1. Then $\Omega(K) \subseteq L(G, \gamma_M) \subseteq K$ (by Proposition 1.1) and $L(G, \gamma_M) = L(S_3/G)$.

Proposition 1.3: For each $l \in ML(G, \gamma_M)$, let $\Gamma_l' = \emptyset$ in Proposition 1.1. Then $\Omega(K) \subseteq L(G, \gamma_M) \subseteq K$ (by Proposition 1.1) and $L(G, \gamma_M) = L(S_M/G)$.

The proofs of Propositions 1.2 and 1.3 are straightforward (see [9] and [11]).

The algorithms of Propositions 1.2 and 1.3 will always generate regular languages ($\gamma_M(l)$ depends only on the value of $\hat{\xi}(l)$, which can only have a finite number of values). Proposition 1.1, on the other hand, can be used to generate a nonregular language, since the value of $\gamma_M(l)$ depends not only on $\hat{\xi}(l)$ but on the values of Γ_l', which is chosen by the DES designer. It is possible for Γ_l' to be chosen so that the closed-loop language is nonregular.

The algorithms for computing the partial information supervisors in Propositions 1.1–1.3 can be implemented online. If one has already computed $\gamma_M(l)$, then the computation of $\gamma_M(l\lambda)$, where λ is an observed event, does not have to be performed until λ is observed by the supervisor.

The overall computational complexity of the supervisors presented in Propositions 1.1–1.3 is measured by examining the number of computations needed (in the worst case) each time an event is observed. The computation requires that the values of $\hat{\xi}_+(l)$, $\hat{\xi}(l\lambda)$, and $\gamma_M(l\lambda)$ each be computed once. It is easy to show that the joint computation of $\hat{\xi}_+(l)$ and $\hat{\xi}(l\lambda)$ requires at most $|\Sigma||X|$ computations. This is due to the fact that in the computation of $\hat{\xi}_+$ unobservable events are used, and in the computation of $\hat{\xi}$ observable events are used. Any additional computations are made in the selection of $\gamma_M(l\lambda)$. If the algorithm of Proposition 1.1 is used, then the overall complexity is $O(|\Sigma||X|)$.

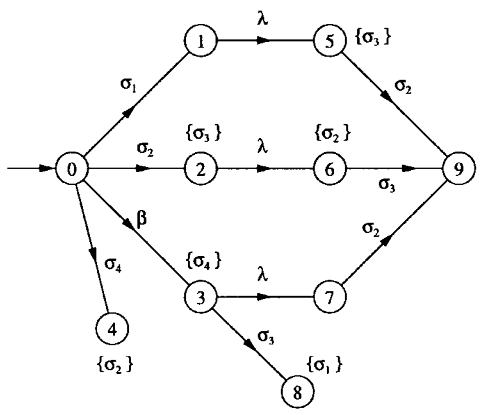

Fig. 1. The DFA used in the examples of Section IV.

IV. Examples

Fig. 1 shows a DFA $D = (X, \Sigma, \xi, x_0)$, where the state set of D is $X = \{0, 1, \cdots, 9\}$, the event set is $\Sigma = \{\beta, \lambda, \sigma_1, \cdots, \sigma_4\}$, and the initial state is $x_0 = 0$. The transition function ξ is defined in the figure by the labeled arrows between states, for example, $\xi(\lambda, 1) = 5$. The DFA D generates the language $K = \overline{\sigma_1\lambda\sigma_2 + \sigma_2\lambda\sigma_3 + \beta(\sigma_3 + \lambda\sigma_2) + \sigma_4}$.

Assume that all events are controllable except β, so $\Sigma_c = \{\lambda, \sigma_1, \cdots, \sigma_4\}$. Assume that the event λ is observable and all other events are unobservable, so $\Sigma_\epsilon = \{\beta, \sigma_1, \cdots, \sigma_4\}$. Thus, for $\sigma \in \Sigma$, $M(\sigma) = \lambda$ if $\sigma = \lambda$ and $M(\sigma) = \epsilon$ otherwise.

The events that are shown inside braces in Fig. 1 represent controllable events that extend strings of K into the plant language $L(G)$, but not into K. For example, the strings $\sigma_2\sigma_3$ and $\beta\sigma_4$ are assumed to be in $L(G)$ and not in K. The language K is a (possibly supremal) controllable sublanguage of a legal language $L \subseteq L(G)$. It can be shown that $\Omega(K) = \overline{\beta\lambda}$ is a controllable and observable sublanguage of K.

The events inside braces also define a (full-information) supervisor γ that satisfies $L(G, \gamma) = K$. For example, $\gamma(\sigma_2) = \{\sigma_3\}$ and $\gamma(\beta\lambda) = \emptyset$. The corresponding disable map γ_D gives disabled event sets as a function of the states in X, for example, $\gamma_D(6) = \{\sigma_2\}$ and $\gamma_D(7) = \emptyset$. By extending the domains of ξ and γ_D, we define the extended disable map $\overline{\gamma}_D$ and the extended transition function $\overline{\xi}$.

We use Proposition 1.1 to generate three different closed-loop (i.e., controllable and observable) sublanguages, all strictly containing $\Omega(K)$ and contained in K. To allow comparison among supervisors, we denote them γ_M^1, γ_M^2, and γ_M^3.

Example 1: Let $\Gamma'_l = \Sigma_c$ for all $l \in ML(G, \gamma_M^1)$.

By definition of closed-loop languages, $\epsilon \in L(G, \gamma_M^1)$. Let $\Gamma'_\epsilon = \Sigma_c$. We have that $\hat{\xi}^1(\epsilon) = \{0\}$. Using (6) compute $\Gamma_\epsilon = \Gamma'_\epsilon \cap \overline{\gamma}_D\overline{\xi}([\Sigma_\epsilon - \Gamma'_\epsilon]^*, \hat{\xi}^1(\epsilon)) = \Sigma_c \cap \overline{\gamma}_D\overline{\xi}(\{\beta\}^*, \{0\}) = \Sigma_c \cap \overline{\gamma}_D(\{0, 3\}) = \{\sigma_4\}$. Compute $\gamma_M^1(\epsilon) = \overline{\gamma}_D\overline{\xi}([\Sigma_\epsilon - \Gamma_\epsilon]^*, \hat{\xi}^1(\epsilon)) = \overline{\gamma}_D\overline{\xi}([\Sigma_\epsilon - \{\sigma_4\}]^*, \{0\}) = \overline{\gamma}_D(\{0, 1, 2, 3, 8\} = \{\sigma_1, \sigma_3, \sigma_4\}$. Given the value of $\gamma_M^1(\epsilon)$, we find that the sets of strings $\overline{\beta\lambda}$ and $\overline{\sigma_2\lambda}$ are in the closed-loop language. Hence $\lambda \in ML(G, \gamma_M^1)$. Consequently, $\hat{\xi}^1(\lambda) = \{6, 7\}$. Let $\Gamma'_\lambda = \Sigma_c$. By (6) we find that $\Gamma_\lambda = \{\sigma_2\}$.

Compute $\gamma_M^1(\lambda) = \overline{\gamma}_D\overline{\xi}([\Sigma_\epsilon - \Gamma_\lambda]^*, \hat{\xi}^1(\lambda)) = \overline{\gamma}_D\overline{\xi}([\Sigma_\epsilon - \{\sigma_2\}]^*, \{6, 7\}) = \overline{\gamma}_D(\{6, 7, 9\}) = \{\sigma_2\}$. Thus the string $\sigma_2\lambda\sigma_3$ is also in the closed-loop language. Thus $L(G, \gamma_M^1) = \overline{\beta\lambda + \sigma_2\lambda\sigma_3}$. By Proposition 1.2, $L(G, \gamma_M^1) = L(S_3/G)$ [11].

Example 2: Let $\Gamma'_l = \emptyset$ for all $l \in ML(G, \gamma_M^2)$.

Let $\Gamma'_\epsilon = \emptyset$. Thus by (6) $\Gamma_\epsilon = \emptyset$. Proceeding with the computation as in Example 1, we get $\gamma_M^2(\epsilon) = \overline{\gamma}_D(\{0, 1, 2, 3, 4, 8\}) = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$. Given $\gamma_M^2(\epsilon)$, we find that the set of strings $\overline{\beta\lambda}$ is in the closed-loop language. Hence $\lambda \in ML(G, \gamma_M^2)$. Consequently, $\hat{\xi}^2(\lambda) = \{7\}$.

Let $\Gamma'_\lambda = \emptyset$. Thus $\Gamma_\lambda = \emptyset$. Proceeding as in Example 1, we get $\gamma_M^2(\lambda) = \overline{\gamma}_D(\{7, 9\}) = \emptyset$. Thus the string $\beta\lambda\sigma_2$ is also in the closed-loop language. Thus $L(G, \gamma_M^2) = \overline{\beta\lambda\sigma_2}$. By Proposition 1.3, $L(G, \gamma_M^2) = L(S_M/G)$.

Example 3: Let $\Gamma'_l = \{\sigma_3, \sigma_4\}$ for all $l \in ML(G, \gamma_M^3)$.

Let $\Gamma'_\epsilon = \{\sigma_3, \sigma_4\}$. We find from (6) that $\Gamma_\epsilon = \{\sigma_3, \sigma_4\}$. Proceeding as above, we get $\gamma_M^3(\epsilon) = \overline{\gamma}_D(\{0, 1, 2, 3\}) = \{\sigma_3, \sigma_4\}$. Given $\gamma_M^3(\epsilon)$, we find that the set of strings $\overline{(\sigma_1 + \beta + \sigma_2)\lambda}$ is in the closed-loop language. Hence $\lambda \in ML(G, \gamma_M^3)$. Consequently, $\hat{\xi}^3(\lambda) = \{5, 6, 7\}$.

Let $\Gamma'_\lambda = \{\sigma_3, \sigma_4\}$. By (6) we find that $\Gamma_\lambda = \{\sigma_3\}$. We get $\gamma_M^3(\lambda) = \overline{\gamma}_D(\{5, 6, 7, 9\}) = \{\sigma_2, \sigma_3\}$. No more strings are in the closed-loop language. Thus $L(G, \gamma_M^3) = \overline{\beta\lambda + \sigma_2\lambda + \sigma_1\lambda}$. □

We have shown (by Examples 1 and 2) that the languages [11] and $L(S_M/G)$ [9] can be generated, as well as other languages (Example 3). Other languages (viz., elements of the class) are generated by choosing (for each l) different values of Γ'_l in Proposition 1.1. Since $L(S_3/G)$ [11] and $L(S_M/G)$ [9] are generated by *particular* choices of Γ'_l, the proposed supervisor synthesis algorithm is a generalization of the two algorithms presented in [9] and [11]. Each of the three generated languages contains strings not in the others. Consequently, the generated languages are incomparable to each other in terms of set inclusion. All three languages properly include $\Omega(K)$ and are contained in K.

Similar classes of languages have been found in [8] and [13] using controllable-event ordering. In [8], a class of languages that contain $\Omega(K)$ [and hence also contain $\sup CCN(K)$] was found. Unfortunately, no online procedure has yet been developed for computing supervisors for those languages. In [13] an online technique was used to synthesize supervisors for a class of maximal sublanguages. The synthesis algorithm is called in [13] the Variable Lookahead Policy under Partial Observation (VLP-PO) algorithm. In addition, [13] provides a scheme of event ordering that ensures that the generated maximal language includes $\sup CCN(K)$ and observes that another event-ordering scheme exists to guarantee that the generated maximal languages contain $\Omega(K)$. In this sense the languages $\sup CCN(K)$ and $\Omega(K)$ can be "maximized." On the other hand, it also shows that there *does not exist* an event-ordering scheme which ensures that the generated maximal language contains $L(S_M/G)$. Our class of languages contains elements [such as $L(S_M/G)$] not generated (nor "maximized") by the VLP-PO algorithm. Consequently, the class of languages generated here is incomparable (in terms of language inclusion) to the class generated by the VLP-PO algorithm.

V. Conclusion

A partial information supervisor that generates a class of closed controllable and observable sublanguages of a closed and regular legal language is presented. The supervisor can be implemented *online* using a technique introduced by Heymann and Lin [9] where computations of the disabled event set can be made after each event observation. The generated languages contain a language proposed independently by Heymann and Lin [9], Kumar [10], and Fa *et al.* [8]; this language contains the supremal closed controllable and normal sublanguage of the legal language. The generated class of languages is incomparable (in terms of set inclusion) to both the class of maximal languages generated by Hadj-Alouane *et al.* [13] and the class of languages generated in [8]. The (online) computational complexity of our algorithms is $O(mn)$, where m is the number of plant events and n is the number of states in the generator for the supremal closed controllable sublanguage of the legal language.

In some applications (e.g., plants with a large number of events that can occur very frequently) it is possible that the algorithms presented here are still too computationally intensive to be implemented online. This topic requires future study.

Appendix A
Proof of Theorem 1

We assume that γ_M is a partial information supervisor. The corresponding functions $\hat{\xi}$ and $\hat{\xi}_+$ are defined in the statement of Theorem 1. All other assumptions are stated in the beginning of Section III. The first lemma shows that $L(G, \gamma_M)$ is indeed a sublanguage of K.

Lemma A.1: Assume that $(\forall l \in ML(G, \gamma_M))\overline{\gamma}_D\hat{\xi}_+(l) \subseteq \gamma_M(l)$. Then $s \in L(G, \gamma_M)$ implies that $s \in K$ and $\xi(s, x_0) \in \xi_+(Ms)$.

Proof: We use induction on $|s|$.

1) Basis: $|s| = 0$, so $s = \epsilon$. $L(G, \gamma_M)$ is a closed-loop language, so $\epsilon \in L(G, \gamma_M)$. By assumption, since K is nonempty and closed, $\epsilon \in K$. We also have that $\xi(\epsilon, x_0) = x_0 \in \{x_0\} = \overline{\xi}(\{\epsilon\}, \{x_0\})$ and therefore $\xi(\epsilon, x_0) \in \overline{\xi}([\Sigma_\epsilon - \gamma_M(\epsilon)]^*, \{x_0\}) = \hat{\xi}_+(\epsilon)$.

2) Induction: Assume that the lemma holds for all $|s| \leq k, k > 0$. Let $|s| = k$. We shall first show that $s\sigma \in L(G, \gamma_M) \Rightarrow s\sigma \in K$. Let $s\sigma \in L(G, \gamma_M)$. Then $s\sigma \in L(G)$ and $\sigma \notin \gamma_M(Ms)$. Let $l = Ms$. By assumption, $\sigma \notin \overline{\gamma}_D\hat{\xi}_+(l)$. By the inductive hypothesis $\sigma \notin \gamma_D(\xi(s, x_0))$, hence $\sigma \notin \gamma(s)$. Since $s\sigma \in L(G)$, $s \in K$, $\sigma \notin \gamma(s)$, and $L(G, \gamma) = K$, we conclude that $s\sigma \in K$. To show that $\xi(s\sigma, x_0) \in \hat{\xi}_+(M(s\sigma))$, two cases are considered, when $M\sigma = \epsilon$ and when $M\sigma \neq \epsilon$. For both, we start using the inductive hypothesis to get $\xi(s\sigma, x_0) \in \overline{\xi}(\{\sigma\}, \{\xi(s, x_0)\}) \subseteq \overline{\xi}(\{\sigma\}, \hat{\xi}_+(l))$. It is then straightforward to show that in each case, $\overline{\xi}(\{\sigma\}, \hat{\xi}_+(l)) \subset \hat{\xi}_+(M(s\sigma))$. This completes the proof of the lemma. $\square$

Proof of Theorem 1: Assume that (5) holds. Due to Lemma A.1, all that remains to be shown is that $\Omega(K) \subseteq L(G, \gamma_M)$. For this we use induction on $|s|$. The basis step $s = \epsilon$ is trivial, since both are closed-loop languages.

Induction: Assume that for all s with $|s| \leq k, k > 0$, $s \in \Omega(K) \Rightarrow s \in L(G, \gamma_M)$. Let $|s| = k$. We want to show that $s\sigma \in \Omega(K) \Rightarrow s\sigma \in L(G, \gamma_M)$. Let $s\sigma \in \Omega(K)$. Then $s \in \Omega(K) \subseteq K$, $s \in L(G, \gamma_M)$, $s\sigma \in L(G)$, and $\sigma \notin \gamma_M^\Omega(Ms)$. Let $l = Ms$. From the definition of γ_M^Ω, we have that $\gamma_M^\Omega(l) = \overline{\gamma}_D\overline{\xi}(M_K^{-1}l, \{x_0\})$. It is straightforward to show that $(\forall s \in L(G, \gamma_M))\overline{\xi}(\Sigma_\epsilon^*, \hat{\xi}(Ms)) \subseteq \overline{\xi}(M_K^{-1}Ms, \{x_0\})$. Consequently, $\sigma \notin \overline{\xi}(\Sigma_\epsilon^*, \hat{\xi}(l))$ and therefore $\sigma \notin \gamma_M(l)$. Since $s \in L(G, \gamma_M)$ and $s\sigma \in L(G)$, we conclude that $s\sigma \in L(G, \gamma_M)$.

References

[1] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete-event processes," *SIAM J. Contr. Opt.,* vol. 25, pp. 206–230, Jan. 1987.

[2] F. Lin and W. M. Wonham, "On observability of discrete-event systems," *Info. Sci.,* vol. 44, pp. 173–198, 1988.

[3] P. J. Ramadge and W. M. Wonham, "The control of discrete-event systems," *Proc. IEEE,* vol. 77, pp. 81–97, Jan. 1989.

[4] R. D. Brandt, V. Garg, F. Lin, S. I. Marcus, and W. M. Wonham, "Formulas for calculating supremal controllable and normal sublanguages," *Syst. Contr. Lett.,* vol. 15, pp. 111–117, 1990.

[5] R. Cieslak, C. Desclaux, A. S. Fawaz, and P. Varaiya, "Supervisory control of discrete-event processes with partial observations," *IEEE Trans. Automat. Contr.,* vol. 33, pp. 249–260, Mar. 1988.

[6] R. Kumar, V. Garg, and S. I. Marcus, "On controllability and normality of discrete-event dynamical systems," *Syst. Contr. Lett.,* vol. 17, pp. 157–168, 1991.

[7] K. G. Rudie, "Software for the control of discrete-event systems: A complexity study," Elect. Eng. Dept., Univ. Toronto, Canada, Systems Control Group Rep. 8806, Sept. 1988.

[8] J. Fa, X. Yang, and Y. Zheng, "Formulas for a class of controllable and observable sublanguages larger than the supermal controllable and normal sublanguage," *Syst. Contr. Lett.,* vol. 20, pp. 11–18, 1993.

[9] M. Heymann and F. Lin, "On-line control of partially-observed discrete-event systems," *DEDS,* vol. 4, no. 3, pp. 221–236, 1994.

[10] R. Kumar, "Observability formulas for discrete-event dynamical systems," in *Proc. Conf. Info. Sci. Syst.,* Johns Hopkins University, Baltimore, MD, 1993, pp. 581–596.

[11] J. H. Prosser and M. Kam, "Supervisor synthesis for approximating maximally-allowable behaviors," in *Proc. Conf. Info. Sci. Syst.,* Princeton Univ., Princeton, NJ, Feb. 1994.

[12] M. Heymann and F. Lin, "On-line control of partially-observed discrete-event systems," Comp. Sci. Dept., Technion-Israel Inst. Technol., Haifa, Israel, Tech. Rep. CIS-9310R, 1993.

[13] N. B. Hadj-Alouane, S. Lafortune, and F. Lin, "On-line synthesis of maximal control policies for partially observed discrete event systems," College of Eng., Univ. Michigan, Ann Arbor, MI, Control Group Rep. CGR-94-04, Oct. 1994.

[14] W. M. Wonham and P. J. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM J. Contr. Opt.,* vol. 25, pp. 637–659, May 1987.

[15] F. Lin and W. M. Wonham, "Decentralized supervisory control of discrete-event systems," *Info. Sci.,* vol. 44, pp. 199–224, 1988.

On Damped Algebraic Riccati Equations

C.-Y. He, J. J. Hench, and V. Mehrmann

***Abstract*—In a recent paper, an algorithm was proposed which produces dampening controllers based on damped algebraic Riccati equations (DARE's) derived from a periodic Hamiltonian system. The solution to one of these DARE's is symmetric and the other, skew-symmetric; both of these solutions lead to a dampening feedback, i.e., a stable closed-loop system for which the real parts of the eigenvalues are larger in modulus than the imaginary parts.**

In this paper, the authors extend these results to include a broader class of damped algebraic Riccati equations which have Hermitian and skew-Hermitian solutions and show that every convex combination of these solutions produces a dampening feedback. This property can be used to vary the feedback with two parameters and thus obtain more flexibility in the controller design process.

***Index Terms*—Damped algebraic Riccati equations, dampening feedback, linear quadratic control, periodic Hamiltonian systems, periodic Schur decomposition.**

Manuscript received September 5, 1996. This work was supported in part by the Engineering and Physical Sciences Research Council of the United Kingdom under Grant GR/K 83618, by the Academy of Sciences of the Czech Republic, by the Grant Agency of the Czech Republic under Contract 102/94/0294, and by the Deutsche Forschungsgemeinschaft Projekt Singuläre Steuerungsprobleme under Contract ME-790/7-1.

C.-Y. He is with the Department of Mathematics, University of Kansas, Lawrence, KS 66045 USA.

J. J. Hench was with the Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, 128 08 Praha 8, Czech Republic. He is now with the Guidance and Control Analysis Group, Jet Propulsion Laboratory, Pasadena, CA 91109 USA (e-mail: John.Hench@jpl.nasa.gov).

V. Mehrmann is with the Technische Universität Chemnitz-Zwickau, Fakultät für Mathematik, D-09107 Chemnitz, Germany.

Publisher Item Identifier S 0018-9286(98)07543-6.