5 References

[1] R. D. Brandt, V. Garg, R. Kumar, F. Lin, S. I. Marcus and W. M. Wonham, 1990. Formulas for calculating supremal controllable and normal sublanguages. *Systems & Control Letters, 15*, pp. 111-117.

[2] Y. Brave and M. Heymann, 1990. On stabilization of discrete event processes. *International Journal of Control, 51(5)*, pp. 1101-1117.

[3] Y. Brave and M. Heymann, 1993. On optimal attraction in discrete event processes. *Information Sciences, 67*, pp. 245-276.

[4] E. Chen and S. Lafortune, 1991. Dealing with blocking in supervisory control of discrete event systems. *IEEE Transactions on Automatic Control, 36(6)*, pp. 724-735.

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

[6] R. Cieslak, C. Desclaux, A. Fawaz and P. Varaiya, 1988. Supervisory control of discrete-event processes with partial observations. *IEEE Transactions on Automatic Control, 33(3)*, pp. 249-260.

[7] M. Heymann, 1990. Concurrency and discrete event control. *IEEE Control Systems Magazine, 10(4)*, pp. 103-112.

[8] M. Heymann, 1993. Some algorithmic questions in discrete-event control, to appear.

[9] M. Heymann and F. Lin, 1993. On-line control of partially observed discrete-event systems, Technion CIS report No 9310, Haifa, Israel.

[10] L. E. Holloway and B. H. Krogh, 1990. Synthesis of feedback control logic for a class of controlled Petri nets. *IEEE Transactions on Automatic Control, 35(5)*, pp. 514-523.

[11] R. Kumar, V. Garg and S. I. Marcus, 1992. Stability and stabilizability of behavior of discrete event dynamical systems. *SIAM Journal of Control and Optimization*, to appear.

[12] S. Lafortune and F. Lin, 1991. On tolerable and desirable behaviors in supervisory control of discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications, 1*, pp. 61-92.

[13] F. Lin, 1991. Control of large scale discrete event systems: task allocation and coordination. *Systems & Control Letters, 17*, pp. 169-175.

[14] F. Lin and W. M. Wonham, 1988. On observability of discrete event systems. *Information Sciences, 44(3)*, pp. 173-198.

[15] F. Lin and W. M. Wonham, 1988. Decentralized supervisory control of discrete-event systems. *Information Sciences, 44(3)*, pp. 199-224.

[16] F. Lin and W. M. Wonham, 1990. Decentralized control and coordination of discrete event systems with partial observation. *IEEE Transactions on Automatic Control, 35(12)*, pp. 1330-1337.

[17] R. J. Ramadge, 1989. Some tractable supervisory control problems for discrete-event systems modeled by Buchi automata. *IEEE Transactions on Automatic Control, 34(1)*, pp. 10-19.

[18] R. J. Ramadge and W. M. Wonham, 1987. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization, 25(1)*, pp. 206-230.

[19] K. Rudie and W. M. Wonham, 1992. Think globally, act locally: decentralized supervisory control. *IEEE Transactions on Automatic Control*, to appear.

[20] J. G. Thistle, 1991. Control of infinite behavior of discrete event systems. *Ph. D. Thesis*, Department of Electrical Engineering, University of Toronto.

[21] J. N. Tsitsiklis, 1989. On the control of discrete-event dynamical systems. *Mathematics of Control, Signals, and Systems, 2(1)*, pp. 95-107.

[22] Y. Willner and M. Heymann, 1992. Language convergence in controlled discrete-event systems. *CIS Report, 9205*, Technion.

[23] Y. Willner and M. Heymann, 1991. Supervisory control of concurrent discrete-event systems. *Int. Journal on Control, 54*, pp. 1143-1169.

[24] W. M. Wonham and P. J. Ramadge, 1987. On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization, 25(3)*, pp. 635-659.

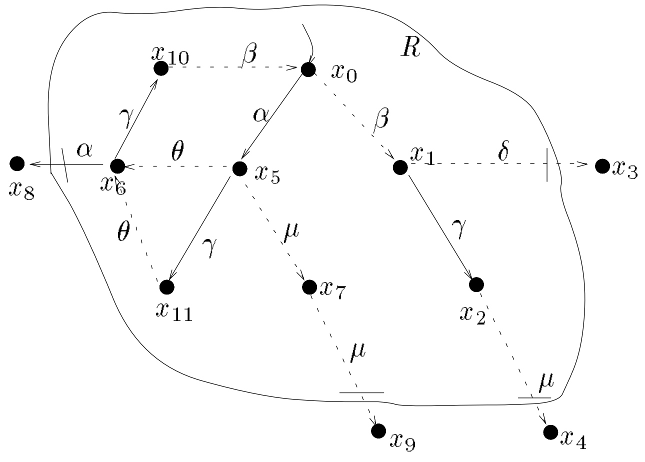

Figure 1:

Example 3

Consider the process depicted in Figure 1 where R consists of the transitions confined to the enclosed area while the outgoing transitions (to x_3, x_4, x_8, x_9) indicate transitions disabled by the application of the supervisor ψ, that is,

$$\psi(x_1) = \{\delta\}, \psi(x_2) = \psi(x_7) = \{\mu\}, \psi(x_6) = \{\alpha\}$$
$$\psi(x_i) = \emptyset \text{ otherwise.}$$

The observable and unobservable events are $\Sigma_o = \{\alpha, \gamma\}$ and $\Sigma_{uo} = \{\beta, \mu, \theta, \delta\}$. The partially observed closed loop system is obtained (on-line) by application of Algorithm 1 as depicted in Figure 2:

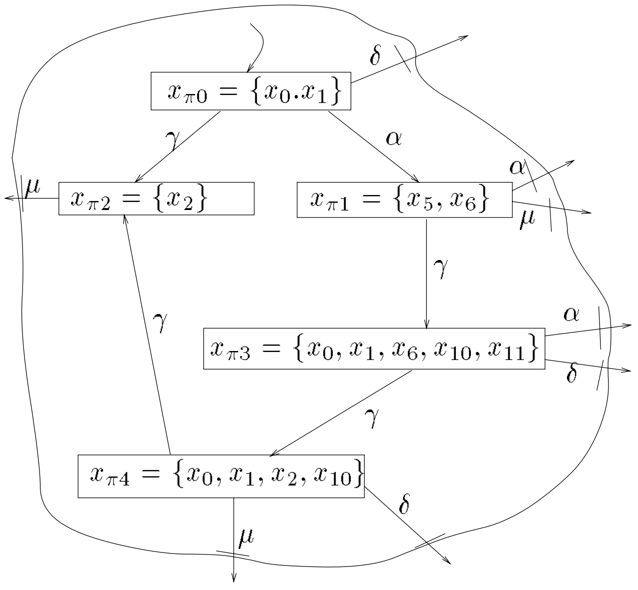
δ
$x_{\pi 0} = \{x_0.x_1\}$
γ
α
α
μ
$x_{\pi 2} = \{x_2\}$
$x_{\pi 1} = \{x_5, x_6\}$
μ
γ
γ
α
$x_{\pi 3} = \{x_0, x_1, x_6, x_{10}, x_{11}\}$
δ
γ
$x_{\pi 4} = \{x_0, x_1, x_2, x_{10}\}$
δ
μ

Figure 2:

Sample Calculations

- Initialization

$$\bar{x}_0 = \{x_0, x_1\}$$
$$\psi_\pi(x_0) = \{\delta\}$$
$$x_{\pi 0} = \{x_0, x_1\}$$

- $x_\pi = x_{\pi 3} = \{x_0, x_1, x_6, x_{10}, x_{11}\}, \gamma$ is observed

$$x_{\pi 3}(\gamma) = \{x_2, x_{10}\}$$
$$\bar{x}_{\pi 3}(\gamma) = \{x_0, x_1, x_2, x_{10}\}$$
$$\psi_\pi(x_{\pi 3}(\gamma)) = \{\mu, \delta\}$$
$$x_{\pi 4} = \xi_\pi(\gamma, x_{\pi 3}) = \{x_0, x_1, x_2, x_{10}\}$$

□

It is important to note that the on-line computation of the partial observation supervisor S_π is universal in that it is independent of the specific properties of the underlying full-observation supervisor S. This, in particular implies that if the supervisor S is implemented on-line, say by a limited lookahead policy ([5]), then the supervisor S_π can also be implemented on line. This last issue will be discussed in detail elsewhere.

4 Complexity considerations

It has been well known for some time that synthesis of supervisors for operation under partial observation is problematic in that: 1. optimal supervisors (in the sense of supremal observable languages) do not exist [14], and hence the typical synthesis has been confined to supervisors that achieve only supremal normal sublanguages, 2. the synthesis, even for this non-optimal case, is an NP-complete problem [21].

The on-line approach proposed in the present paper circumvents the complexity problem in that it bypasses the need to design the full supervisor. This is achieved by relying on the predesigned full-observation supervisor whose design can be accomplished with linear complexity even when the specification language is not closed [8]. Specifically, if the size of the state set of the process G is n and that of the state set of the recognizer of the specification language K is m, then the design complexity of the full-observation supervisor is $O(\mid n \mid\mid m \mid)$. The adaptation of the full-observation supervisor to operation under partial observation is performed stepwise via Algorithm 1. Each step of the algorithm consists of at most two reachability tests in the state set of the automaton R whose dimension is at most $\mid n \mid\mid m \mid$. These reachability tests can be performed with complexity $O(\mid n \mid\mid m \mid)$ using standard algorithms.

Thus, the on-line approach provides supervisor computation with stepwise-linear complexity.

the set of initial states possibly reached upon application of the initial control $\psi_\pi(x_o)$ is not $x^o_{\pi o}$ but rather,

$$x_{\pi o} = \{x \in X \mid (\exists t \in (\Sigma_{uo}/\psi_\pi(x_o))^*)\xi(t, x_o) = x\}.$$

which is contained in $x^o_{\pi o}$. Similar containment relations hold after occurrences of observable events. This leads to a supervisor $S_\pi = (R_\pi, \psi_\pi)$ which is less restrictive (has fewer disablements) than S^o_π as follows.

$$R_\pi = trim(\Sigma_o, X_\pi, \xi_\pi, x_{\pi o})$$

where

$$\xi_\pi(\sigma, x_\pi) = \{x \in X \mid (\exists x' \in x_\pi(\sigma))$$
$$(\exists t \in (\Sigma_{uo}/\psi_\pi(x_\pi(\sigma)))^*)\xi(t, x') = x\}.$$

With the modified supervisor S_π, the closed loop language is given by $L(S_\pi/G)$, which is characterized by the following theorem.

Theorem 5 If $supC(E) \neq \emptyset$, then

$$supCN(E) \subseteq L(S^o_\pi/G) \subseteq L(S_\pi/G) \subseteq supC(E).$$

□

It is clear that the straightforward implementation of S_π is computationally inefficient because of the exponential blow-up in the size of the state set X_π of R_π relative to the size of the state set X of R. However, as we shall see below, the computation of S_π need not be performed explicitly and in advance for all states in X_π in order to achieve successful implementation. Indeed, having computed in advance the supervisor S for implementation under full observation, we can proceed with the implementation of S_π using an *on-line* approach.

By an on-line approach to supervisory control we mean that at each stage of an actual execution, the required control action is computed just for that stage. More specifically, suppose that the process is running and is currently at a state x_π. Suppose further that an observable event σ has just taken place. First, all the controllable events are immediately disabled. Next, we compute the control action required by the supervisor S_π, when at $x_\pi(\sigma)$. Upon completion of the computation of the required control action, the events that need not be disabled at that point are re-enabled. Under this control the closed loop process makes a (partially observed) transition to the state $\xi_\pi(\sigma, x_\pi)$. Our basic assumption is that, upon the occurrance of an observable event σ, the temporary disablement of all controllable events can be accomplished before any other controllable event takes place and that the computation time of the control at each state is sufficiently short so that the interim step of disablement of all controllable events is of negligible duration.

The precise computational steps required for on-line implementation of the supervisor S_π given above are as follows:

Algorithm 1 On-line implementation.

Initialization

The initialization consists of computation of the required initial control $\psi_\pi(x_o)$ and its application, and of the computation of the resultant state (set) $x_{\pi o}$.

- Compute $\overline{x}_o = \{x \in X \mid (\exists t \in \Sigma_{uo}{}^*)$ $\xi(t, x_o) = x\}$ by removing from the process R all transitions labeled by observable events and computing the set of states reachable from x_o in the resultant process.

- Compute the control $\psi_\pi(x_o) = \bigcup_{x \in \overline{x}_o} \psi(x)$ and apply it.

- Compute $x_{\pi o} = \{x \in X \mid (\exists t \in (\Sigma_{uo}/\psi_\pi(x_o))^*)$ $\xi(t, x_o) = x\}$ by removing from the process R all transitions labeled by observable events and all transitions labeled by events in $\psi_\pi(x_o)$, and computing the set of states reachable from x_o in the resultant process.

General step

Assume that the process is at an arbitrary (known) state $x_\pi \in X_\pi$, that $\psi_\pi(x_\pi)$ is known and applied. Assume further that an observable event $\sigma \in \Sigma_o$ has just taken place and that all controllable events have just been temporarily disabled.

The algorithm computes the target state (set) ${x'}_\pi = \xi_\pi(\sigma, x_\pi)$ and the control $\psi_\pi({x'}_\pi)$ and applies the computed control upon completion of the computation. (Note that $\overline{x}'_\pi = \overline{x}_\pi(\sigma)$ so that $\psi_\pi(x'_\pi) = \psi_\pi(x_\pi(\sigma))$.)

- Compute $x_\pi(\sigma) = \{x \in X \mid (\exists x' \in x_\pi)\xi(\sigma, x') = x\}$

- Compute the unobserved reach $\overline{x}_\pi(\sigma)$ of $x_\pi(\sigma)$,

$$\overline{x}_\pi(\sigma) = \{x \in X \mid (\exists x' \in x_\pi(\sigma))(\exists t \in \Sigma_{uo}{}^*)\xi(t, x') = x\}.$$

- Compute the control $\psi_\pi(x_\pi(\sigma)) = \bigcup_{x \in \overline{x}_\pi(\sigma)} \psi(x)$ and apply it.

- Compute the target state (set)

$$x'_\pi = \xi_\pi(\sigma, x_\pi) = \{x \in X \mid (\exists x' \in x_\pi(\sigma))$$
$$(\exists t \in (\Sigma_{uo}/\psi_\pi(x_\pi(\sigma)))^*)\xi(t, x') = x\}$$

by removing from the process R all transitions labeled by observable events and all transitions labeled by events in $\psi_\pi(x_\pi(\sigma))$, and computing the set of all states reachable from some state in $x_\pi(\sigma)$ in the resultant process. □

that is, γ_π disables after each string $s \in L(G)$, every event $\sigma \in \Sigma_c$ that is disabled by some element of s_π, the equivalence class of s.

It is readily seen that γ_π acts as a supervisor under partial observation, i.e., as a map

$$\gamma_\pi : \pi L(G) \to 2^{\Sigma_c}$$

because it disables exactly the same events after every $s \in s_\pi$.

We turn next to the examination of various properties of the supervisor γ_π. Denote $K = L(G, \gamma)$ and $K_\pi = L(G, \gamma_\pi)$. Then K_π is characterized as follows.

Theorem 1

$$\epsilon \in K_\pi$$

$$(\forall s \in K_\pi) s\sigma \in K_\pi \Leftrightarrow s\sigma \in L(G) \wedge (\forall s' \in K) \\ ((s' \in s_\pi \wedge s'\sigma \in L(G)) \Rightarrow s'\sigma \in K).$$

□

Proposition 2 The language $L(G, \gamma_\pi)$ is observable with respect to $L(G)$. □

Theorem 2 $L(G, \gamma_\pi) \subseteq L(G, \gamma)$. □

From Theorem 1, we can conclude that K_π is a closed, controllable and observable sublanguage of K. We show next that K_π contains every closed normal sublanguage of K and hence, in particular, its supremal closed normal sublanguage, denoted by $supN(K)$.

Theorem 3 Let $M \subseteq K$ be a closed normal sublanguage. Then $M \subseteq K_\pi$. □

Corollary 1 $supN(K) \subseteq K_\pi$. □

The above approach to modifying a supervisor is general in that it is independent of the particular way in which the original supervisor is designed. If the original supervisor γ is designed to solve the supervisory control problem [18] [24], in which $L(G, \gamma) = supC(E)$, the supremal controllable sublanguage of the maximal legal language E, then the modified supervisor γ_π generates a language that contains the supremal controllable and normal sublanguage of E, denoted by $supCN(E)$ [1] [14]. This fact is established in the following

Theorem 4 If $supC(E) \neq \emptyset$, then

$$supCN(E) \subseteq L(G, \gamma_\pi)$$

□

Next we give two examples that demonstrate some properties of the modified supervisors. The first example below shows that $L(G, \gamma_\pi)$ can contain $supCN(E)$ as a strict subset.

Example 1 Let $\Sigma = \Sigma_c = \{\alpha, \beta, \lambda\}$, $\Sigma_o = \{\alpha\}$, $L(G) = \overline{\alpha\beta\lambda + \alpha\lambda\beta}$, and $E = \overline{\alpha\beta + \alpha\lambda}$. Then $supCN(E) = \{\epsilon\}$. But $L(G, \gamma_\pi) = \{\epsilon, \alpha\}$. □

The following example shows that the language $L(G, \gamma_\pi)$ is not necessarily a maximal observable sublanguage of $supC(L(G))$.

Example 2 Let $\Sigma = \Sigma_c = \{\alpha, \beta, \lambda, \mu\}$, $\Sigma_o = \emptyset$, $L(G) = \overline{\alpha(\beta + \lambda) + \beta(\lambda + \mu) + \mu}$ and $E = \overline{\alpha + \beta + \mu}$. Then $L(G, \gamma_\pi) = \{\epsilon, \alpha\}$. But note that the language $M = \{\epsilon, \alpha, \mu\}$ is also observable and $L(G, \gamma_\pi) \subset M$ with strict inclusion. □

3 Implementation

In the previous section we have shown how a supervisor γ can be modified to a supervisor γ_π suitable for operation under partial observation. We have shown the properties and advantages of γ_π as compared with a supervisor that is synthesized directly for operation under the condition of partial observation. In the present section we shall discuss the issue of algorithmic implementation.

In many respects, the method for implemention of γ_π depends on how γ itself is implemented. Suppose, for example, that γ is implemented by a recognizer $R = (\Sigma, X, \xi, x_o)$ and a feedback map $\psi : X \to 2^{\Sigma_c}$, such that for each state $x \in X$, $\psi(x)$ is the (smallest) set of controllable events that must be disabled in G when R is in x. Thus, the supervisor is implemented as the pair $S = (R, \psi)$ where each string $s \in L(S/G)$ is represented by a unique state $x \in X$. Furthermore assume that, without loss of generality, the language generated by S is equal to the language generated by R, i.e., $L(S/G) = L(R)$ and ψ disables events only when it is necessary to do so. Then a direct implementation of the modified supervisor in the previous section is $S_\pi^o = (R_\pi^o, \psi_\pi)$ defined as follows. First, for any subset $x_\pi \subseteq X$, define its *unobserved reach* $\overline{x}_\pi$ as

$$\overline{x}_\pi = \{x \in X \mid (\exists x' \in x_\pi)(\exists t \in \Sigma_{uo}{}^*)\xi(t, x') = x\}.$$

Then, the generator R_π^o is given by

$$R_\pi^o = trim(\Sigma_o, X_\pi, \xi_\pi^o, x_{\pi o}^o)$$

where the state set $X_\pi = 2^X$, $x_{\pi o}^o = \overline{x}_o = \overline{\{x_o\}}$, and $\xi_\pi^o(\sigma, x_\pi) = \overline{x}_\pi(\sigma)$ with $x_\pi(\sigma) = \{x \in X \mid (\exists x' \in x_\pi)\xi(\sigma, x') = x\}$.

The control feedback map ψ_π is defined for each state x_π as

$$\psi_\pi(x_\pi) = \bigcup_{x \in \overline{x}_\pi} \psi(x).$$

Notice that the control ψ_π restricts the transition behavior of the system and hence the possible states that the supervised system may reach. For example, since the events in $\psi_\pi(x_o) = \psi_\pi(\{x_o\})$ are disabled,

Σ_{uc}; $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$. It is also partitioned into the observable event set Σ_o and the unobservable event set Σ_{uo}; $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$.

A supervisor is used to restrict the behavior of the closed loop system by disabling some controllable events. Under the condition of full observation, a supervisor is characterized by a map

$$\gamma : L(G) \to 2^{\Sigma_c},$$

where for each $s \in L(G)$, $\gamma(s)$ is the set of events disabled by the supervisor γ after the string s. The language $L(G,\gamma)$ generated by G under supervision by γ is given recursively by

$$\begin{array}{l} \epsilon \in L(G,\gamma) \\ (\forall s \in L(G,\gamma)) s\sigma \in L(G,\gamma) \Leftrightarrow s\sigma \in L(G) \wedge \sigma \notin \gamma(s). \end{array}$$

It is well known [17] that given a sublanguage $K \subseteq L(G)$, there exists a supervisor γ such that $L(G,\gamma) = K$ if and only if K is closed and controllable.

Suppose now that $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$ and let $\pi : \Sigma^* \to {\Sigma_o}^*$ be the projection map that erases from every string the unobservable events. That is, π is defined inductively as

$$\begin{array}{l} \pi(\epsilon) = \epsilon \\ (\forall s \in \Sigma^*)\ \pi(s\sigma) = \left\{ \begin{array}{ll} \pi(s)\sigma & \text{if } \sigma \in \Sigma_o \\ \pi(s) & \text{if } \sigma \in \Sigma_{uo} \end{array} \right. . \end{array}$$

Under partial observation, a supervisor is characterized by

$$\tilde{\gamma} : \pi L(G) \to 2^{\Sigma_c},$$

that is, $\tilde{\gamma}$ is a map defined on the set of projected (observed) strings, and $\tilde{\gamma} \circ \pi$ is a map from $L(G)$ to 2^{Σ_c}. The language $L(G,\tilde{\gamma})$ generated by G under supervision by $\tilde{\gamma}$ is given inductively by

$$\begin{array}{l} \epsilon \in L(G,\tilde{\gamma}) \\ (\forall s \in L(G,\tilde{\gamma})) s\sigma \in L(G,\tilde{\gamma}) \Leftrightarrow s\sigma \in L(G) \wedge \sigma \notin \tilde{\gamma}(\pi s). \end{array}$$

The goal of supervisor synthesis is to design a supervisor $\tilde{\gamma}$ for a given language $K \subseteq L(G)$ such that $L(G,\tilde{\gamma}) = K$. It can be proved [14] that such a $\tilde{\gamma}$ exists if and only if K is closed, controllable and observable. The definitions of controllability and observability, as given below, were introduced in [18] [14].

A sublanguage $K \subseteq L(G)$ is *controllable* (with respect to $L(G)$) if

$$(\forall s \in \overline{K})(\forall \sigma \in \Sigma_{uc}) s\sigma \in L(G) \Rightarrow s\sigma \in \overline{K}.$$

Let $\Delta \subseteq \Sigma$ be any subset. A sublanguage $K \subseteq L(G)$ is Δ-*observable* (with respect to $L(G)$) if

$$\begin{array}{c} (\forall s, s' \in \overline{K} \mid \pi(s) = \pi(s'))(\forall \sigma \in \Delta) \\ (s\sigma \in \overline{K} \wedge s'\sigma \in L(G)) \Rightarrow s'\sigma \in \overline{K}. \end{array}$$

K is called *observable* if it is Σ-observable.

In [14], a stronger version of observability, called normality is also defined. A sublanguage $K \subseteq L(G)$ is *normal* (with respect to $L(G)$) if

$$(\forall s \in L(G)) \pi(s) \in \pi(\overline{K}) \Rightarrow s \in \overline{K}.$$

It is readily shown that $L(G,\tilde{\gamma})$ is observable with respect to $L(G)$. We state this fact below for completeness.

Proposition 1 $kL(G,\tilde{\gamma})$ is observable with respect to $L(G)$. $\square$

The design of full-observation supervisors can be accomplished with linear computational complexity (see e.g [8]). We shall show below how to modify a full-observation supervisor so as to apply under the condition of partial-observation. Our approach is universal in that it applies to every full-observation supervisor regardless of the method by which it has been designed. A further major advantage is that given any full-observation supervisor, our modification algorithm for operation under partial observation can be implemented on-line with tep-wise linear computational complexity.

2 Modified Supervisors

For a language L over Σ, the projection map π induces a natural equivalence relation E over L such that for every two strings $s, s' \in L$

$$sEs' \Leftrightarrow \pi(s) = \pi(s').$$

This equivalence relation partitions L into equivalence classes such that each $s \in L$ belongs to a unique equivalence class s_π

$$\begin{array}{ll} s_\pi & = \{s' \in L \mid \pi(s') = \pi(s)\} \\ & = L \cap \pi^{-1}\pi(s). \end{array}$$

In the quotient language $\pi(L) \subseteq {\Sigma_o}^*$, each equivalence class s_π is represented by a single string $\pi(s)$ ($s \in s_\pi$). It is not difficult to see that the main property of supervisors under partial observation is that they act exactly the same way after all strings in $L(G)$ that belong to the same equivalence class. This fact gives us an immediate clue as to how to modify a given supervisor to one that is suitable for operation under partial observation.

To this end we proceed as follows. Let γ be a supervisor designed to solve a control problem under full observation. Without loss of generality, we assume that γ disables events only when it is necessary to do so. In other words,

$$\gamma(s) = \left\{ \begin{array}{ll} \{\sigma \mid s\sigma \in L(G) - L(G,\gamma)\} & \text{if } s \in L(G,\gamma) \\ \emptyset & \text{otherwise.} \end{array} \right.$$

Let E be the equivalence relation (as explained above) over the language $L(G)$. The modified supervisor for partial observation γ_π is then given as

$$\gamma_\pi(s) = \bigcup_{s' \in s_\pi} \gamma(s'),$$

ON-LINE CONTROL OF PARTIALLY OBSERVED DISCRETE EVENT SYSTEMS

Michael Heymann[*]
Department of Computer Science
Technion, Israel Institute of Technology
Haifa 32000, Israel
E-mail: heymann@cs.technion.ac.il

Feng Lin[†]
Dept. of Electrical and Computer Engineering
Wayne State University
Deteroit, MI 48202
E-mail: flin@ece.eng.wayne.edu

March 9, 1993

Abstract

It is well known that the design of supervisors for partially observed discrete-event systems is an NP-complete problem and hence computationally impractical. Furthermore, optimal supervisors for partially observed systems do not generally exist. Hence, the best supervisors that can be designed directly for operation under partial observation are the ones that generate the supremal normal (and controllable) sublanguage. In the present paper we show that a standard procedure exists by which any supervisor that has been designed for operation under full observation, can be modified to operate under partial observation. When the procedure is used to modify the optimal full-observation supervisor (i.e., the one that generates the supremal controllable language), the resultant modified supervisor is at least as efficient as the best one that can be designed directly (that generates the supremal normal sublanguage). The supervisor modification algorithm can be carried out on-line with linear computational complexity and hence makes the control under partial observation a computationally feasible procedure.

Key words: discrete event systems, supervisory control, partial observation, on-line control.

1 Introduction

Supervisors have been used to solve different problems in the discrete event systems literature. Examples are the supervisory control problem [18], the supervisory control and observation problem [14], the decentralized control problem [6] [15] [19] [23], the coordination problem [13] [16], the optimal attraction problem [2] [3], the language convergence problem [11] [22], the supervisory control problem with infinite behavior [17] [20], the supervisory control problem with blocking [4], the supervisory control problem under tolerance [12], the supervisory control using Petri nets [10] and others [7]. In many of these problems, the supervisors are obtained under the assumption that all the events are observable. However, this assumption is often violated in practice, because observing all events may be impossible or inefficient. In such cases, observability becomes an issue. In general, control problems under partial observation are much more complicated mainly due to the following two facts. (1) Observable languages do not have the nice properties that controllable languages have. In particular, the supremal observable sublanguage of a given language may not exist. (2) The computation of supervisors for control under partial observation is an NP-complete problem and therefore impractical in most applications. To overcome these two difficulties, we introduce in the present paper a new method for construction of supervisors for partially observed systems. We first construct a supervisor under the assumption of full observation. This can be accomplished by standard methods as described in the above mentioned references (see also [8]). We then modify the supervisor to incorporate the constraint of partial observation. We shall see that our approach has significant computational advantages in addition to yielding supervisors that are at least as efficient as the ones obtained by direct synthesis. A complete version of the paper including all the proofs is given in [9].

As usual, let G be the discrete event system to be controlled and $L(G)$ the language generated by G. Σ^* is the set of all strings over the event set Σ, including the empty string ϵ. We say that a language L is *closed* if all the prefixes of L also belong to L. We will only discuss closed language in this paper. The event set is partitioned into the controllable event set Σ_c and the uncontrollable event set

[*]Completed in part while the author was on leave as a NRC Senior Research Associate at NASA-Ames Research Center, Moffett Field, California 94035. Also supported in part by the Technion fund for promotion of research.

[†]Supported in part by the National Science Foundation.