V. Conclusion

In this note, a class of new observation problem, called simultaneous observation problem, is introduced and studied. This is achieved by using the coprime factorization approach. Necessary and sufficient conditions for the existence of simultaneous observation are derived. It was shown that a general solution for the simultaneous observation problem reduces the problem of observing n plants using a common observer to one of observing $n-1$ plants using a common observer.

References

[1] M. Vidyasagar, *Control Systems Synthesis: A Factorization Approach*. Cambridge, MA: MIT Press, 1985.

[2] P. Saeks and J. Murray, "Fractional representation, algebraic geometry and the simultaneous stabilization problem," *IEEE Trans. Automat. Contr.*, vol. AC-27, pp. 895–903, 1982.

[3] M. Vidyasagar and N. Viswanadham, "Algebraic design techniques for reliable stabilization," *IEEE Trans. Automat. Contr.*, vol. AC-27, pp. 1085–1095, 1982.

[4] X. Ding, P. M. Frank, and L. Guo, "Robust observer design for dynamical systems under unknown disturbances," in *Proc. First IFAC Symp. Design Methods Contr. Syst.*, ETH Zurich, Switzerland, 1991, pp. 290–295.

[5] J. O'Reilly, *Observer for Linear Systems*. London: Academic, 1983.

[6] X. Ding and P. M. Frank, "Fault detection via factorization approach," *Syst. Cont. Lett.*, vol. 14, pp. 431–436, 1990.

An Algorithm for Computing the Mask Value of the Supremal Normal Sublanguage of a Legal Language

Michel Barbeau, Guy Custeau, and Richard St-Denis

***Abstract*—We consider the problem of finding the mask value of the supremal normal sublanguage L_R of some given language L. We describe a straightforward algorithmic solution that can be applied to existing off-line procedures for determining the supremal controllable and normal sublanguage of L and that does not require an explicit calculation of L_R. This problem is fundamental because it is related to the supervisory control problem under partial observation. Our algorithm applies only to closed languages.**

I. Introduction

One of the basic problems in supervisory control is to design a controller whose task is to enable and disable the controllable events of a discrete-event system (DES) such that the control requirements expressed as a legal language L are satisfied. A unifying theory has been developed by Ramadge and Wonham [11] to define and solve this problem. Lin and Wonham [10] and Cieslak *et al.* [6] have studied, in the framework of Ramadge and Wonham, the supervisory control problem under partial observation. Following the theory of Ramadge and Wonham, the uncontrolled DES is represented by a generator, which is a deterministic automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is a set of states, Σ a finite set of events, δ: $\Sigma \times Q \rightarrow Q$ a transition function, $q_0 \in Q$ an initial state, and $Q_m \subseteq Q$ a set of marked states. Let $\Gamma = \{0,1\}^{\Sigma}$ be the set of all binary assignments to the elements of Σ, and let $\gamma \in \Gamma$ and $\sigma \in \Sigma$. If $\gamma(\sigma) = 1$ then σ is enabled; otherwise, σ is disabled. Let Σ_c and Σ_u be fixed disjoint subsets of Σ denoting the sets of controllable and uncontrollable events respectively, such that $\Sigma = \Sigma_c \cup \Sigma_u$. A controller is a pair $C = (S, \phi)$, where $S = (Y, \Lambda, \zeta, y_0, Y_m)$ is a deterministic automaton and ϕ: $Y \rightarrow \Gamma$ is a feedback function satisfying: i) $\phi(y)(\sigma) = 1$, if $\sigma \in \Sigma_u$; and ii) $\phi(y)(\sigma) \in \{0,1\}$, otherwise.

Manuscript received December 20, 1993; revised August 1, 1994. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada and the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR).

The authors are with the Département de mathématiques et d'informatique, Université de Sherbrooke, Sherbrooke, Québec, Canada J1K 2R1.

IEEE Log Number 9408778.

To control a DES, Ramadge and Wonham introduced a transition function δ_c: $\Gamma \times \Sigma \times Q \rightarrow Q$ defined by

$$\delta_c(\gamma, \sigma, q) = \begin{cases} \delta(\sigma, q), & \text{if } \delta(\sigma, q) \text{ is defined and } \gamma(\sigma) = 1 \\ \text{undefined}, & \text{if } \delta(\sigma, q) \text{ is undefined or } \gamma(\sigma) = 0. \end{cases}$$

The controlled discrete-event system (CDES) is then the generator $G_c = (Q, \Gamma \times \Sigma, \delta_c, q_0, Q_m)$ obtained from G by specifying the sets Σ_c and Σ_u.

Following the extensions proposed by Cieslak *et al.* [6], we consider the case in which C observes all the events of G_c through a mask or observation function M that maps each event in Σ into an observed event in $\Lambda \cup \{\varepsilon\}$. The events in $M^{-1}(\varepsilon)$ are those that cannot be seen by C. If C cannot distinguish between σ_1 and σ_2, then $M(\sigma_1) = M(\sigma_2)$. If M is the identity function, then $\Lambda = \Sigma$ and all the original events are observed by C. The special case in which M simply erases some of the events in Σ occurs frequently and is called a natural projection [12] or natural mask [14].

Finally, the CDES and controller are embodied in a closed-loop system to constitute a supervised discrete-event system (SDES) C/G_c, which is defined to be the generator $(Y \times Q, \Sigma, (\zeta \circ M) \times \delta_c, (y_0, q_0), Y_m \times Q_m)$, where the function

$$(\zeta \circ M) \times \delta_c : \Sigma \times Y \times Q \rightarrow Y \times Q$$

is defined as $((\zeta \circ M) \times \delta_c)(\sigma, y, q) = (\zeta(M(\sigma), y), \delta_c(\phi(y), \sigma, q))$, if $\delta(\sigma, q)$ and $\zeta(M(\sigma), y)$ are defined, and $\phi(y)(\sigma) = 1$, and is undefined, otherwise.

Let $K \subseteq L \subseteq L(G) \subseteq \Sigma^*$ and $\Sigma' \subseteq \Sigma$. We recall that a language K is closed if $K = \overline{K}$, the prefix closure of K. In this paper, we assume that all languages are closed and denote by $\bar{s}$ the prefix closure of a string $s \in \Sigma^*$. Language K is $(\Sigma', L(G))$-controllable if $(\forall s \in K)(\forall \sigma \in \Sigma')[s\sigma \in L(G) \Rightarrow s\sigma \in K]$. Language K is $(M, L(G))$-normal if $(\forall s \in K)(\forall s' \in L(G))[M(s) = M(s') \Rightarrow s' \in K]$. Finally, K is $(\Sigma', L(G))$-observable if $(\forall s, s' \in K)(\forall \sigma \in \Sigma')[(M(s) = M(s') \wedge s\sigma \in K \wedge s'\sigma \in L(G)) \Rightarrow s'\sigma \in K]$.

The supervisory control problem under partial observation is formally stated as follows. Given a CDES G_c, its legal behavior $L \subseteq L(G)$, and a mask function M, find a controller $C = (S, \phi)$ such that $L(C/G_c)$ is the largest sublanguage of L that is closed, $(\Sigma_u, L(G))$-controllable and $(M, L(G))$-normal. This problem is slightly different from the supervisory control and observation problem (SCOP) first formulated by Lin and Wonham [10], in which M is a natural mask and $L(C/G_c)$ is constrained to contain a minimal acceptable language.

Let us consider a simple example. Fig. 1 depicts generator G of the language of a plant and automaton R of its legal language L. The observation function is shown in Table I. Note that the illegal event c of the plant is observed as the legal event b. Note also that the words $adef$ and $bdef$ of the plant are observed as the legal words

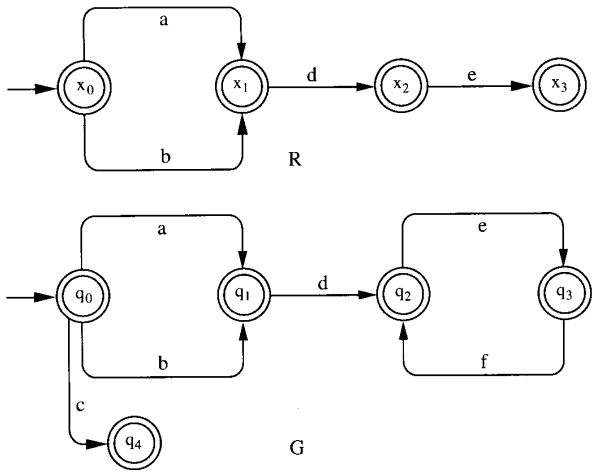

Fig. 1. Automata R and G.

ade and bde, respectively, but that $adef$ and $bdef$ are not admitted. The largest normal sublanguage of L clearly is: $L_R = \{a, ad\}$. To make R an automaton for L_R, the transition from x_0 to x_1 on event b must be removed as well as state x_3 with the incoming transition from x_2. In this way, indistinguishable behaviors of the plant are all permitted by the controller if they are all legal, or all rejected if at least one of them is illegal.

Cieslak *et al.* [6] proved, under the assumption that $M^{-1}(M(\Sigma_u) - \{\varepsilon\}) \subseteq \Sigma_u$, that K_R, the supremal controllable and normal language contained in L, can be computed as follows:

Step 1) Compute L_R, the largest $(M, L(G))$-normal sublanguage of L.

Step 2) Compute $M(L_R)$, $M(L(G))$, and $\Delta_u = M(\Sigma_u) - \{\varepsilon\}$.

Step 3) Compute K, the largest $(\Delta_u, M(L(G)))$-controllable sublanguage of $M(L_R)$, by using the algorithm developed by Ramadge and Wonham [15].

Step 4) Compute $K_R = M^{-1}(K) \cap L(G)$.

This paper presents an alternate procedure for computing the supremal controllable and normal sublanguage K_R, in which Step 1) above is omitted. That is to say, our procedure does not require an intermediate representation of L_R. It includes a new algorithm for computing directly the mask value $M(L_R)$ of the supremal normal sublanguage L_R. Indeed, $L(G)$, $M(L_R)$, and $M(L(G))$ are the only languages required for completing the last two steps of the procedure. Compared with the approach of Cieslak *et al.* [6], our procedure provides a simpler and more straightforward algorithmic solution.

This paper is organized as follows. Section II presents a review of known related algorithms. Section III gives a description of our algorithm and a proof of its correctness. Section IV provides a comparison with existing algorithms and contains concluding remarks.

II. Related Work

The supervisory control problem under partial observation has received much attention, and several algorithms for computing the supremal controllable and normal sublanguage of a given language have been proposed in the literature [4]–[6], [14]. They can all be used for the off-line derivation of a controller, but the worst-case computational complexity of all these algorithms is theoretically exponential because there is no polynomial-time algorithm for the SCOP unless $P = NP$ [13]. The exponential-time complexity results from the construction of one or more deterministic automata from nondeterministic automata. But, it should be noted that in practice the worst-case occurs rarely [1]. An on-line approach was, however, suggested by Heymann and Lin [7] for improving this bound. They consider the required control action to be calculated at each step of the actual execution of the closed-loop system. The basic idea behind their algorithm is that a controller that has been designed for operation under full observation can be modified to operate under partial observation. The computational complexity at each step is polynomial in the product of the number of states in automaton R for the legal language L and the number of states in G. Furthermore, the closed-loop behavior thus achieved is a controllable and observable sublanguage larger than the supremal controllable and normal sublanguage. It should be noted, however, that the normality and observability properties are equivalent under the assumptions that the language is controllable and all controllable events are observable [9].

TABLE I
The Observation Function M

σ	a	b	c	d	e	f
$M(\sigma)$	a	b	b	ε	e	ε

Nevertheless, the off-line approach is useful when there is a need to design a full controller because the plant is highly time-critical or the closed-loop system must be verified prior to its execution. In addition to the off-line procedure of Cieslak *et al.* [6], Brandt *et al.* [4], [14] propose the algebraic formula $L - M^{-1}(M(L(G) - L))$ for the off-line computation of L_R. These operations can be interpreted as follows. Take the set of illegal words, that is, $L(G) - L$. Compute how the illegal words are observed, that is, $M(L(G) - L)$. Compute the set of words, legal or illegal, that are observed as illegal words, that is, $M^{-1}(M(L(G) - L))$. Reject from L the legal words for which there are illegal words observed the same way. This formula can be effectively computed with the TCT tool [14] under the hypothesis that M is a natural mask. The computation of L_R includes, however, the construction of six intermediate automata.

Cho and Marcus [5] give two algorithms for computing L_R. The first one is based on a graphical characterization of the notion of normal language. Indeed, they prove, under the assumption that R is a strict-subautomaton of G, that a language L is normal if and only if T_s is a subautomaton of T, where T and T_s are the deterministic automata for $M(L(G))$ and $M(L)$, respectively. The main step of this algorithm consists of the elimination, from T_s, of states and edges so that the resulting automaton $\hat{T}_s$ is the largest subautomaton of T. Language L_R equals $L \cap M^{-1}(L(\hat{T}_s))$. In comparison, our algorithm constructs a single intermediate nondeterministic finite automaton, that is, the one for T_s. Needless to say, elimination of states and edges is performed according to a different procedure.

The second algorithm provided by Cho and Marcus [5] constructs a nondeterministic automaton for the language L_R directly from G, R, and T, under the stronger assumption that G has a property called M-recognizable. If this assumption is satisfied, the cardinality of the state set of T is less than or equal to the cardinality of the state set of G. Unfortunately, G does not always exhibit this structural property, and the computational effort required for obtaining an M-recognizable nondeterministic automaton from G is equivalent to that of their first algorithm.

III. The Algorithm for Computing $M(L_R)$

In this section, an algorithm is presented for going from R, an automaton for the legal language L, to C_2, an automaton for the

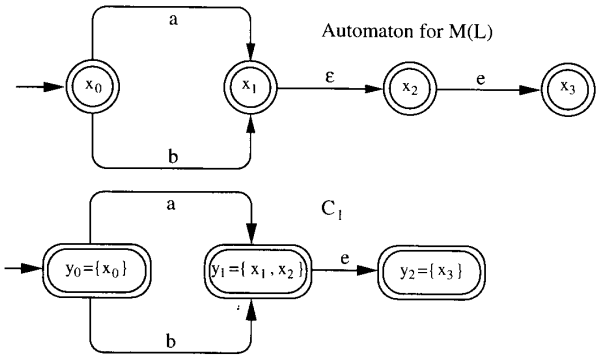

Fig. 2. Automaton for language $M(L)$, and automaton C_1.

projection $M(L_R)$, with respect to a plant G and an observation function M. The algorithm comprises two steps.

In the first step, a deterministic automaton C_1 is constructed by applying the observation function M to every transition label of R and translating the result into a deterministic automaton. The states of C_1 are subsets of the states of R, and $L(C_1) = M(L)$. The algorithm of Lewis and Papadimitriou [8] can be used for this purpose.

Fig. 2 shows the automaton for language $M(L)$ and automaton C_1, for the automaton R introduced in Section I and illustrated in Fig. 1.

Before describing the second step which computes $M(L_R)$ from G, R and M, let us state some easy-to-satisfy properties that G and R must have.

Let $T_1 = (X_1, \Sigma, f_1, x_{01}, X_1)$ and $T_2 = (X_2, \Sigma, f_2, x_{02}, X_2)$ be two deterministic automata of regular languages L_1 and L_2, respectively, with $L_1 \subseteq L_2$. We say that T_1 refines T_2 if

$$(\forall s, t \in L_1)[f_1(s, x_{01}) = f_1(t, x_{01}) \Rightarrow f_2(s, x_{02}) = f_2(t, x_{02})].$$

If T_1 refines T_2, then it can be shown [6] that there exists a unique correspondence function $h : X_1 \to X_2$ such that

$$(\forall s \in L_1)[h \circ f_1(s, x_{01}) = f_2(s, x_{02})].$$

The correspondence function h can be intuitively interpreted as follows. Given a state x of T_1 reached after the occurrence of string s, $h(x)$ yields the state in which T_2 would be after encountering the same string.

For example, in Fig. 1, R refines G, and the correspondence function from the states of R to the states of G is

$$h(x_i) = q_i (i = 0, 1, 2, 3).$$

In the sequel,[1] let $G = (Q, \Sigma, \delta, q_0, Q)$, $R = (X, \Sigma, \xi, x_0, X)$, and M: $\Sigma \to \Lambda \cup \{\varepsilon\}$. We assume, without loss of generality, that R refines G. Moreover, let h: $X \to Q$ denote the correspondence function, and $C_1 = (Y_1, \Lambda, \zeta_1, y_0, Y_1)$ with $Y_1 \subseteq 2^X$.

A. The Concept of Normal Transition

A transition of C_1, from a state y to a state y', labeled with an event $\lambda \in (\Lambda \cup \{\varepsilon\})$, is normal if there is no state $x_i \in y$ and event $\tau \in \Sigma$ such that:

- in state $h(x_i)$ of the plant, event τ is active, that is, $\delta(\tau, h(x_i))!$[2];
- event τ is observed as λ, that is, $M(\tau) = \lambda$; and
- event τ is not admitted from x_i, that is, $\xi(\tau, x_i)$ is undefined.

[1] In this paper, every state is marked and every language is closed.

[2] $\delta(\tau, h(x_i))!$ means that $\delta(\tau, h(x_i))$ is defined.

Intuitively, C_1 is an intermediate structure that mirrors the legal language as observed through the mask function. A transition of C_1 labeled λ, from state y to state y', is on the path leading to the acceptance of a word of the form $u\lambda v$, where $u, v \in \Lambda^*$. Such a word corresponds to a word in the legal language of the form $s\tau t$ with $s, t \in \Sigma^*, \tau \in \Sigma, M(s) = u, M(\tau) = \lambda$, and $M(t) = v$. If there exists another word $s'\tau't'$ in $L(G)$ but not in L, with $M(s') = u, M(\tau') = \lambda$, and $M(t') = v$, this word is illegal and observed the same way as the legal word $s\tau t$. A normal transition is one for which such a word $s'\tau't'$ does not exist. A normal transition labeled λ is called a normal λ-transition.

In the second step of the algorithm, generator C_2 is obtained from C_1 by pruning states and transitions. Pruned states are those for which nonnormal ε-transitions are active, and pruned transitions are those that do not have the normal property. The language of C_2 is $M(L_R)$.

The concept of normal transition is stated in an operational fashion as follows (let Σ_λ denote the set of events in Σ observed as λ, and $y \in Y_1 \subseteq 2^X$)

function normal (y, λ)
{precondition: $\zeta_1(\lambda, y)!$}
forall $x_i \in y$ **do**
  forall $\tau \in \Sigma_\lambda$ **do**
    if $\delta(\tau, h(x_i))!$ **and not** $\xi(\tau, x_i)!$**then**
      return false
return true

B. The Computation of C_2

In this section, we present an algorithm for deriving a deterministic automaton C_2 from C_1 such that a word t in the language of C_2: i) is the projection of some word s in L; and ii) any word s' in $L(G)$ for which the projection is also t, is in L as well. More formally, the following two results will be proved (Lemmata 1, 2, 3, and Theorem 1)

$$1)\ L(C_2) = \{M(s) : s \in L \wedge (\forall s' \in \bar{s})(\forall s'' \in L(G)) [M(s'') = M(s') \Rightarrow s'' \in L]\}$$

and

$$2)\ L(C_2) = M(L_R).$$

The algorithm consists of two loops. The first one inspects every state $y \in Y_1$ of C_1. By convention, the transition $\zeta_1(\varepsilon, y)$ is always defined. If this ε-transition is normal, then state y is copied into set Y_2. Otherwise, the ε-transition is disabled by rejecting state y, because it is impossible to act on nonobservable transitions.

The second loop inspects every event $\lambda \in \Lambda$ that is active from a state $y \in Y_2$. If the transition on that event is defined in C_1, is normal, and leads to a state in Y_2, then it is also defined in C_2.

If the initial state y_0 of C_1 is in Y_2, then the algorithm returns an automaton $C_2 = (Y_2, \Lambda, \zeta_2, y_0, Y_2)$. Otherwise, it returns Φ, the empty automaton for the empty language, which has as its set of states the empty set. The computation of C_2 is performed by the following procedure

function step2 (M, G, R, h, C_1)
{Selection of states with normal ε-transition}
$Y_2 \leftarrow \{\}$
forall $y \in Y_1$ **do**
  if normal(y, ε) **then** $Y_2 \leftarrow Y_2 \cup \{y\}$
{Selection of normal λ-transitions, with $\lambda \in \Lambda$, from states in Y_2}
if $y_0 \in Y_2$ **then**

TABLE II
COMPARISON OF THE ALGORITHMS

Alg.	Authors	Restrictions	Number of intermediate automata for producing L_R	Number of intermediate automata for producing $M(L_R)$	Ease of understanding
1	Cieslak et al.		3	4	Low
2	Brandt et al.	M is a natural projection	6	7	High
3	Cho and Marcus (1)		4	3	Intermediate
4	Cho and Marcus (2)		3	4	Intermediate
5	Our algorithm		3	2	Intermediate

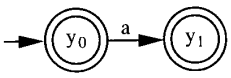

Fig. 3. Automaton C_2.

forall $(\lambda, y) \in (\Lambda \times Y_2)$ **do**
  if $\zeta_1(\lambda, y)!$ **and** normal(y, λ) **and** $\zeta_1(\lambda, y) \in Y_2$ **then**
    $\zeta_2(\lambda, y)$ is equal to $\zeta_1(\lambda, y)$
  else
    $\zeta_2(\lambda, y)$ is undefined
  return $(Y_2, \Lambda, \zeta_2, y_0, Y_2)$
else
  return Φ
end

We first discuss an application of our algorithm, using the example presented in Figs. 1 and 2, then formally demonstrate its correctness.

Example: Referring to Figs. 1 and 2, it is easy to see that normal(y_i, ε) returns true for $i = 0, 1$. This is not the case for $i = 2$, however, since we have $M(f) = \varepsilon$, $\delta(f, h(x_3))$ defined (that is, $\delta(f, q_3)!$) and $\xi(f, x_3)$ undefined. Hence, state y_2 must be eliminated as well as the incoming transition on event e.

Similarly, normal(y_0, a) is true. The result of normal(y_0, b) is false, however, because we have $M(c) = M(b)$, $\delta(c, h(x_0))$ defined (that is, $\delta(c, q_0)!$) and $\xi(c, x_0)$ undefined. Thus, the transition from y_0 to y_1 on event b must be removed. The resulting automaton C_2 is shown in Fig. 3. Another application of our algorithm can be found in Barbeau *et al.* [3].

C. Demonstration of Correctness

We now formally demonstrate the correctness of our algorithm. We first introduce the following two facts:

Fact 1) By construction of C_1, given $y \in Y_1, x_i \in X$, and $s \in \Sigma^*$

$$\zeta_1(M(s), y_0) = y \wedge x_i \in y \Rightarrow (\exists t \in \Sigma^*)[\xi(t, x_0) = x_i \wedge \delta(t, q_0) = h(x_i) \wedge M(t) = M(s)].$$

Fact 2) Given $s \in L$ (that is, $\xi(s, x_0)$ is defined) such that $(\forall s' \in L(G))[M(s') = M(s) \Rightarrow s' \in L]$:
if $\delta(s', q_0)$ is defined and $M(s') = M(s)$, then $\xi(s', x_0)$ is defined.

Lemma 1: Given $s \in L$,

$$(\forall s' \in L(G))[M(s') = M(s) \Rightarrow s' \in L] \Rightarrow \text{normal}(\zeta_1(M(s), y_0), \varepsilon).$$

Proof: Since C_1 is a deterministic automaton for $M(L)$, then, for an s in L, $\zeta_1(M(s), y_0)$ is defined (it is denoted by y in the sequel).

Let $x_i \in y$ and $\tau \in \Sigma$ such that $\delta(\tau, h(x_i))$ is defined and $M(\tau) = \varepsilon$. From Fact 1), $(\exists t \in \Sigma^*)$ $[\xi(t, x_0) = x_i \wedge \delta(t, q_0) = h(x_i) \wedge M(t) = M(s)]$. Thus, if we let $s' = t\tau$, $\delta(s', q_0) = \delta(t\tau, q_0) = \delta(\tau, \delta(t, q_0)) = \delta(\tau, h(x_i))$ is defined. Furthermore, $M(s') = M(t\tau) = M(t)M(\tau) = M(t) = M(s)$.

It follows from the hypothesis of Lemma 1 and Fact 2) that $\xi(s', x_0)$ is defined. Therefore, $\xi(\tau, x_i) = \xi(\tau, \xi(t, x_0)) = \xi(t\tau, x_0) = \xi(s', x_0)$ is defined and, by definition, normal$(\zeta_1(M(s), y_0), \varepsilon)$ is true. □

Lemma 2: Given $s \in L$

$$(\forall s' \in \bar{s})(\forall s'' \in L(G))[M(s'') = M(s') \Rightarrow s'' \in L] \Leftrightarrow M(s) \in L(C_2).$$

The proof is by induction on the length $|s|$ of s.

Proof of $\Rightarrow$ *(Basis step)* Let s be such that $|s| = 0$. Then, $s = \varepsilon$. By Lemma 1, normal$(\zeta_1(M(s), y_0), \varepsilon)$ is true. Since $M(\varepsilon) = \varepsilon$ and $\zeta_1(\varepsilon, y_0) = y_0$, then normal$(y_0, \varepsilon)$ is true. By construction of C_2, $y_0 \in Y_2$. Therefore, $M(\varepsilon) \in L(C_2)$.

(Induction step) Let s be such that $|s| = n + 1$, with $n \geq 0$. We may write s as $u\sigma$ for some $u \in \Sigma^*$ and some $\sigma \in \Sigma$, with $|u| = n$.

By hypothesis, s is such that $(\forall s' \in \bar{s})(\forall s'' \in L(G))$ $[M(s'') = M(s') \Rightarrow s'' \in L]$. In particular, u is such that $(\forall u' \in \bar{u})(\forall u'' \in L(G))[M(u'') = M(u') \Rightarrow u'' \in L]$. By the induction hypothesis, $M(u) \in L(C_2)$. Therefore, $\zeta_2(M(u), y_0)$ is defined. For $\zeta_2(M(u\sigma), y_0)$ to be defined, we have to show:

1) that normal$(\zeta_1(M(u\sigma), y_0), \varepsilon)$ is true; and
2) that the transition from state $\zeta_1(M(u), y_0)$ to state $\zeta_1(M(u\sigma), y_0)$ labeled $M(\sigma)$ is normal.

The first assertion follows from Lemma 1. The second assertion is verified by using an argument similar to the one developed in Lemma 1. Let $x_i \in \zeta_1(M(u), y_0)$ and $\tau \in \Sigma$ such that $\delta(\tau, h(x_i))$ is defined and $M(\tau) = M(\sigma)$. From Fact 1), $(\exists t \in \Sigma^*)[\xi(t, x_0) = x_i \wedge \delta(t, q_0) = h(x_i) \wedge M(t) = M(u)]$. Then, if we let $s' = t\tau$, $\delta(s', q_0) = \delta(t\tau, q_0) = \delta(\tau, \delta(t, q_0)) = \delta(\tau, h(x_i))$ is defined. Furthermore, $M(s') = M(t\tau) = M(t)M(\tau) = M(u)M(\sigma) = M(u\sigma) = M(s)$. It follows from the hypothesis and Fact 2) that $\xi(s', x_0)$ is defined. Therefore, $\xi(\tau, x_i) = \xi(\tau, \xi(t, x_0)) = \xi(t\tau, x_0) = \xi(s', x_0)$ is defined and, by definition, normal$(\zeta_1(M(u), y_0), M(\sigma))$ is true. By construction of C_2, $\zeta_1(M(u\sigma), y_0) \in Y_2$ and $\zeta_2(M(\sigma), \zeta_2(M(u), y_0))$ is defined. Therefore $M(s) \in L(C_2)$.

Proof of $\Leftarrow$ *(Basis step)* Let s be such that $|s| = 0$. Then $s = \varepsilon$, $\bar{s} = \{s\}$, and $M(s) = \varepsilon$. Let us suppose that there exists an $s'' \in L(G)$ such that $M(s'') = M(s)$. We may write $s'' = \tau_1\tau_2 \cdots \tau_m$ and $M(\tau_1) = M(\tau_2) = \cdots = M(\tau_m) = \varepsilon$.

Since $M(s) \in L(C_2)$, then normal(y_0, ε) is true, by construction of C_2. Furthermore, since $\delta(\tau_1, q_0)$ is defined and $M(\tau_1) = \varepsilon$, then $\xi(\tau_1, x_0)$ is defined and $\xi(\tau_1, x_0) = h^{-1}(\delta(\tau_1, q_0)) \in y_0$. In the

same way

$$\xi(\tau_2, \xi(\tau_1, x_0)) \in y_0, \ldots, \xi(\tau_m, \xi(\ldots, \xi(\tau_1, x_0)\cdots)) \in y_0.$$

Therefore, $\xi(s'', x_0)$ is defined and $s'' \in L$.

(Induction step) Let $s \in L$, with $|s| = n + 1$ $(n \geq 0)$, and $M(s) \in L(C_2)$. Since L and $L(C_2)$ are closed, for all proper prefixes s' of s, $s' \in L$, $M(s') \in L(C_2)$, and the conclusion is immediate by the induction hypothesis. It remains to be shown that the conclusion holds for $s' = s$.

We may write s as $t\sigma$ for some $t \in \Sigma^*$ and some $\sigma \in \Sigma$, with $|t| = n$. Assume that there is an $s'' \in L(G)$, with $M(s'') = M(s)$. There are two cases: $M(\sigma) = \varepsilon$ and $M(\sigma) \neq \varepsilon$.

If $M(\sigma) = \varepsilon$, then $t\sigma$ is observed as t which is a proper prefix of s and the conclusion is immediate.

Consider the case where $M(\sigma) \neq \varepsilon$. We may write $s'' = t''\tau u''$, where $t'', u'' \in \Sigma^*$, $\tau \in \Sigma$, $M(t'') = M(t)$, $M(\tau) = M(\sigma)$, and $M(u'') = \varepsilon$. Since $L(G)$ is closed, $t'' \in L(G)$ and by the induction hypothesis $t'' \in L$. Let $y = \zeta_2(M(t), y_0)$. By construction of C_2, $\xi(t'', x_0) \in y$. Since $\zeta_2(M(t\sigma), y_0)$ is defined, then normal$(y, M(\sigma))$ is true, which implies that $\xi(t''\tau, x_0)$ is defined, and normal$(\zeta_2(M(\sigma), y), \varepsilon)$ is true, which implies that $\xi(u'', \xi(t''\tau, x_0))$ is defined. Consequently, $s'' = t''\tau u'' \in L$. □

Lemma 3:

$$(\forall t \in \Sigma^*)[t \in L(C_2) \Rightarrow (\exists s \in L)[M(s) = t]].$$

Proof: Trivially true since C_2 is derived from C_1, a deterministic automaton for $M(L)$, solely by pruning states and transitions. □

Theorem 1: If L_R is the largest $(M, L(G))$-normal sublanguage of L, then $M(L_R) = L(C_2)$.

Proof: We must show that:

1) if $s \in L_R$, then $M(s) \in L(C_2)$, that is, $M(L_R) \subseteq L(C_2)$; and conversely,
2) if $t \in L(C_2)$, then there exists an $s \in L_R$ such that $M(s) = t$, that is, $L(C_2) \subseteq M(L_R)$.

Proof of 1: Let $s \in L_R$. Since L_R is closed and $(M, L(G))$-normal, we have that $s \in L$ and $(\forall s' \in \bar{s})(\forall s'' \in L(G))[M(s'') = M(s') \Rightarrow s'' \in L]$. From Lemma 2, we may conclude that $M(s) \in L(C_2)$.

Proof of 2: Let $t \in L(C_2)$. From Lemma 3, there exists an $s \in L$ such that $M(s) = t$, and, from Lemma 2, $(\forall s' \in \bar{s})(\forall s'' \in L(G))[M(s'') = M(s') \Rightarrow s'' \in L]$ (that is, $M^{-1}M(s') \cap L(G) \subseteq L$). From Proposition 3.3 of Cieslak *et al.* [6], we may conclude that $s \in L_R$. □

IV. Conclusion

Table II gives the results of an analysis, conducted in [2], comparing the characteristics of our algorithm with those of the four other off-line algorithms mentioned in Section II.

As mentioned in the Introduction, the computation of $(M, L(G))$-normal languages is significant because it is related to the problem of supervisory control under partial observation. The computation of L_R is an intermediate step in the solution of this problem.

Any of the algorithms listed in Table II for computing L_R can be used in combination with either of the two main algorithms for computing a controller C, namely, that of Wonham and Ramadge [15] and Cho and Marcus [5].

The algorithm in [15] is used in [6] for computing the largest $(\Delta_u, M(L(G)))$-controllable sublanguage of $M(L_R)$ and works under the assumption that $M^{-1}(M(\Sigma_u) - \{\varepsilon\}) \subseteq \Sigma_u$. That is, no controllable event can be observed as an uncontrollable event. Furthermore, it takes as input the projection of L_R, that is, $M(L_R)$ rather than L_R. Therefore, the most suitable algorithm in that context is the one which produces $M(L_R)$ with less effort. This is Algorithm 5 because it computes an automaton for $M(L_R)$ through the construction of only two automata.

The algorithm in Cho and Marcus [5] is more general than that in Wonham and Ramadge [15]. Indeed, the assumption that $M^{-1}(M(\Sigma_u) - \{\varepsilon\}) \subseteq \Sigma_u$ is not necessary. The former is based on the application in alternation of the algorithm of Wonham and Ramadge and an algorithm for computing an automaton that generates L_R. This process is therefore iterative, and generality is obtained at the price of a longer run-time. Any of the algorithms listed in Table II can be used for computing L_R. The most suitable one is Algorithm 4, however, because it does not require reconstruction of any intermediate automaton after the first iteration [5].

Acknowledgment

The authors wish to thank M. Wonham and K. Rudie for their guidance and encouragement. They would also like to thank the referees for their comments and suggestions.

References

[1] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques and Tools.* Reading, MA: Addison-Wesley, 1986.
[2] M. Barbeau, G. Custeau, and R. St-Denis, "On the computation of normal languages," in *Proc. Thirty-first Annual Allerton Conf. Communication, Contr. Computing*, Univ. of Illinois at Urbana-Champaign, Sept. 1993.
[3] ——, "Requirements engineering and synthesis of a control system," *Automat. Contr. Production Syst.*, vol. 28, no. 1, pp. 37–52, 1994.
[4] R. D. Brandt, V. Garg, R. Kumar, F. Lin, S. I. Marcus, and W. M. Wonham, "Formulas for calculating supremal controllable and normal sublanguages," *Syst. Contr. Lett.*, vol. 15, no. 2, pp. 111–117, 1990.
[5] H. Cho and S. I. Marcus, "On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation," *Mathematics Contr., Signals Syst.*, vol. 2, pp. 47–69, 1989.
[6] 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, no. 3, pp. 249–260, 1988.
[7] M. Heymann and F. Lin, "On-line control of partially observed discrete event processes with partial observation," Dept. Computer Science, Technion-Israel Institute of Technology, Haifa, Israel, Tech. Rep. CIS-9310, 1993.
[8] H. R. Lewis and C. H. Papadimitriou, *Elements of the Theory of Computation.* Englewood Cliffs, NJ: Prentice-Hall, 1981.
[9] F. Lin and H. Mortazavian, "A normality theorem for decentralized control of discrete event systems," *IEEE Trans. Automat. Contr.*, vol. 39, no. 5, pp. 1089–1093, 1994.
[10] F. Lin and W. M. Wonham, "On observability of discrete-event systems," *Inform. Sciences*, vol. 44, pp. 173–198, 1988.
[11] P. J. 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, 1987.
[12] ——, "The control of discrete event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, 1989.
[13] J. N. Tsitsiklis, "On the control of discrete-event dynamical systems," *Mathematics of Contr., Signals Syst.*, vol. 2, no. 1, pp. 95–107, 1989.
[14] W. M. Wonham, "Notes on control of discrete-event systems," Dept. Electrical Engineering, University of Toronto, June 1993.
[15] W. M. Wonham and P. J. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM J. Contr. Optim.*, vol. 25, no. 3, pp. 637–659, 1987.