SUPERVISORY CONTROL OF DISCRETE EVENT PROCESSES WITH PARTIAL OBSERVATIONS

by

R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya

Memorandum No. UCB/ERL M86/63

7 August 1986

SUPERVISORY CONTROL OF DISCRETE EVENT PROCESSES WITH PARTIAL OBSERVATIONS

by

R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya

Memorandum No. UCB/ERL M86/63

7 August 1986

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

SUPERVISORY CONTROL OF DISCRETE EVENT PROCESSES WITH PARTIAL OBSERVATIONS

by

R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya

Memorandum No. UCB/ERL M86/63

7 August 1986

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Supervisory control of discrete event processes with partial observations[1]

R. Cieslak, C. Desclaux, A. Fawaz and P. Varaiya

Department of Electrical Engineering and Computer Sciences
and Electronics Research Laboratory
University of California, Berkeley CA 94720

ABSTRACT

The paper extends certain aspects of the work of Ramadge and Wonham on the control of a class of discrete event processes. The uncontrolled process is described by a language L whose strings specify the sequences of events $\sigma_1 \cdots \sigma_n$ that the process can execute. The controller makes *partial* observations of the process events. Based on these observations the controller must enable or disable certain process events so that the resulting language generated by the closed loop process is the specified sublanguage $K \subset L$. We also study the case of *decentralized* control where there are several controllers each of which makes partial observations and controls a subset of the process events. The results are illustrated by one example of communication protocols.

August 7, 1986

[1] Research supported by the MICRO program of the State of California, a grant from Bell Communications Research, and Joint Services Electronics Program Contract F49620-84-C-0057.

Supervisory control of discrete event processes with partial observations[1]

R. Cieslak, C. Desclaux, A. Fawaz and P. Varaiya

Department of Electrical Engineering and Computer Sciences
and Electronics Research Laboratory
University of California, Berkeley CA 94720

1. Controlled discrete event process

It is common to describe the behavior of a data communication network using sentences like "At time t_a user A sent a data packet to user B who received it at time t_b." The building blocks of such sentences are phrases – "packet sent" and "packet received" – that indicate the completion of discrete events. Similar sentences are used to specify a control policy or protocol that regulates network behavior, e.g., "If data packet is received, send acknowledgement" and "If acknowledgement is not received, retransmit packet." In this way the control policy prescribes actions that must be taken when certain discrete events occur.

Such descriptions are also employed to specify the sequence of operations involved in the manufacture of microelectronic chips. This typically takes 100-300 discrete operations performed on a variety of machines such as furnaces, steppers, etc. An operation in this sequence may be specified for example by a sentence like "Heat the wafer in a furnace at temperature T for a duration D." An operation may involve inspection of the wafer, and subsequent operations may be contingent on the outcome, e.g., "If the number of defective chips exceeds d, then stop further processing and alert the engineer." This sentence resembles the one describing the communication protocol.

Our purpose here is to introduce a class of mathematical models for formalizing both the description of such discrete event systems and the specification of control policies that regulate them. These models may help to realize potential benefits concerning two related but quite distinct aspects: analysis and implementation.

[1] Research supported by the MICRO program of the State of California, a grant from Bell Communications Research, and Joint Services Electronics Program Contract F49620-84-C-0057.

By *analysis* we mean the study of the system within the formal model in order to prove assertions about system behavior, effectiveness of particular control policies, etc. This is a purely formal study and the validity of any inference that one makes about the real system based on this study rests only on intuition because at present we lack a theory and practice that could link the formal model to the real system by means of experimental evidence. The rest of this paper is concerned exclusively with analysis and we devote a few remarks here to the implementation aspect mentioned previously.

A phrase like "packet sent" or "wafer inspected" is meant to denote the completion of an indivisible or atomic event in some real system. In actuality such a phrase typically refers to a sequence of operations in the real system. For example, suppose a communication protocol is realized by a microprocessor. The protocol command "Send packet to B" may then be realized by a series of microprocessor instructions: Read packet from specified memory location, calculate check bits, find address of B, prepare packet format, load transmitter buffer, etc. Similarly, in the microelectronics example, the command "Heat wafer at temperature T" may be executed by a human operator who must adjust several parameters on the furnace control panel. Thus *implementation* of a control policy specified within the formal model involves the translation of the policy from the model language into the language of the real system – microcode in the first example, natural language in the second. Clearly, the benefits of formal models will be greater if this translation can be done easily, accurately and automatically.

Several classes of models have been proposed for describing the behavior of controlled discrete event systems including, especially, Petri nets and finite state machines (FSM). Here we build on the framework of Ramadge and Wonham [1-4]. Although their framework is essentially based on FSM, it has one important feature that is absent in other approaches in that it allows us to evaluate and compare the effect of different control policies on the behavior of the "uncontroled" system. Other approaches require a separate model for each control policy and do not admit the notion of the uncontrolled system. For those familiar with feedback control, the Ramadge-Wonham approach treats the open loop plant and the controller separately, whereas other approaches only permit models of the closed loop system.

In this section, we extend this basic framework to include the case of a nondeterministic process with controllers making partial observations of the

process events.

1.1. Generator

A *generator* is a nondeterministic automaton

$$G = (Q, \Sigma, f, q_0, Q_m),$$

where Q is the set of states, Σ is the input alphabet of *events* or *transitions*, $f \subset \Sigma \times Q \times Q$ is the transition relation, $q_0 \in Q$ is the initial state, and $Q_m \subset Q$ is the set of *marker* states. Σ is a finite set, whereas Q_m and Q may be infinite. If $(\sigma, q, q') \in f$, this means that the transition σ from q to q' is possible. With an abuse of notation we identify the relation f with the point-to-set function $f(\sigma, q) = \{ q' \in Q \mid (\sigma, q, q') \in f \}$, and we say that $f(\sigma, q)$ is defined if it is nonempty. If $f(\sigma, q)$ contains at most one state for every (σ, q), the generator is said to be deterministic.

Σ^* denotes the set of all finite strings $s = \sigma_1 \cdots \sigma_n$ of elements of Σ, including the empty string ε. f can be extended over strings in Σ^* by: $f(\varepsilon, q) \equiv \{q\}$, and $f(s\sigma, q) = \bigcup_{q'} f(\sigma, q')$ where $q' \in f(s, q)$ whenever $f(s, q)$ is defined and $\exists\, q' \in f(s, q)$ such that $f(\sigma, q')$ is nonempty.

Any subset of Σ^* is a *language* over Σ. The language *generated* by G is

$$L(G) := \{ s \in \Sigma^* \mid f(s, q_0) \text{ is defined} \}.$$

The language *marked* by G is

$$L_m(G) := \{ s \in L(G) \mid f(s, q_0) \cap Q_m \neq \phi \}.$$

Strings in $L_m(G)$ are said to be marked by G. A language marked by a finite state generator is called *regular*.

We think of G as an uncontrolled device that starts in q_0 and executes or generates a sequence of events permitted by f. Events occur instantaneously and asynchronously -- two events cannot occur simultaneously and the time between consecutive events is not fixed. $L(G)$ is the set of all possible sequences of events executed by G. If we think of Q_m as the states reached when the device completes some task, then $L_m(G)$ is the sequences of events that lead to completed tasks. It is helpful to interpret events of G as atomic activities of the device such as "move an object" or "send a message". The "real" time taken by the device to complete an activity is not modeled by G. If $s = \sigma_1 \cdots \sigma_n \in L(G)$, this means only that the device is capable of carrying out

the corresponding activities in the order implied by $\sigma_1 \cdots \sigma_n$; in other words, only "logical" time is modeled.

It is sometimes convenient to eliminate states that are not accessible from q_0. Let

$$Q_a := \{q \mid \exists\, s \text{ s.t. } q \in f(s, q_0)\},\ Q_{a,m} := Q_a \cap Q_m,\ f_a := f\,|_{\Sigma \times Q_a}.$$

The accessible component of G is $A(G) := (Q_a, \Sigma, f_a, q_0, Q_{a,m})$. G is *accessible* if $G = A(G)$, it is *co-accessible* if every string in $L(G)$ can be completed to a string in $L_m(G)$, i.e,

$$s \in L(G) \implies \exists\, t \in \Sigma^*,\ st \in L_m(G).$$

G is *trim* if it is both accessible and co-accessible.

1.2. Controlled discrete event process (CDEP)

To the generator G we add a means of control. Let $\Sigma_c \subset \Sigma$ be a prespecified set of *controlled* events and call $\Sigma_u := \Sigma - \Sigma_c$ the set of *uncontrolled* events. Let Γ be the set of all functions $\gamma\colon \Sigma \to \{0, 1\}$ such that $\gamma(\sigma) = 1$ if $\sigma \in \Sigma_u$. Such a γ is called a *control pattern*; the event σ is said to be enabled or disabled by γ accordingly as $\gamma(\sigma) = 1$ or $\gamma(\sigma) = 0$. Define a new transition relation $f_c \subset \Gamma \times \Sigma \times Q \times Q$ by

$$f_c(\gamma, \sigma, q) := \begin{cases} f(\sigma, q), & \text{if } f(\sigma, q) \text{ is defined and } \gamma(\sigma) = 1 \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

The *controlled discrete event process* (CDEP) is the generator $G_c = (Q, \Gamma \times \Sigma, f_c, q_0, Q_m)$ obtained from G by specifying the set Σ_c of controlled events. An external control can now be applied to this CDEP by switching the control pattern through a sequence $\gamma, \gamma', \gamma'', \cdots$ in such a way that the CDEP behaves in a specified manner. Formally, a supervisor $\underline{S}$ for the CDEP G_c is a pair (S, Φ), where $S = (X, \Sigma, g, x_0, X_m)$ is a *deterministic* automaton and $\Phi\colon X \to \Gamma$ is a function that selects a control pattern $\Phi(x)$ for each state x. S is always assumed to be accessible. Φ is called the *state feedback* map.

Since S executes a sequence of state transitions in response to an input string $s \in \Sigma^*$ we may couple $\underline{S}$ to G_c in a feedback loop by allowing the transitions of S to be forced by (events in) G_c and by constraining G_c by the successive control patterns determined by the states of S via the map Φ.

Define the relation $g \times f_c \subset \Sigma \times X \times Q \times X \times Q$ by

$$(g \times f_c)(\sigma, x, q) = \{g(\sigma, x)\} \times f_c(\Phi(x), \sigma, q).$$

Thus $(g \times f_c)(\sigma, x, q)$ is defined iff $f(\sigma, q)$ is defined, $\Phi(x)(\sigma) = 1$, and $g(\sigma, x)$ is defined. This yields the generator $(X \times Q, \Sigma, g \times f_c, (x_0, q_0), X_m \times Q_m)$, whose accessible component is the *supervised discrete event process* (SDEP)

$$\underline{S}/G_c := A(X \times Q, \Sigma, g \times f_c, (x_0, q_0), X_m \times Q_m).$$

In order to interpret $L(\underline{S}/G_c)$ as the set of sequences of events that can occur when $\underline{S}$ is coupled to G_c, one must ensure that transitions in S are defined whenever they occur in G and are enabled by Φ. Formally, $\underline{S}$ is *complete with respect to* G_c provided that for all $s \in \Sigma^*$, $\sigma \in \Sigma$ the three conditions

(a) $s \in L(\underline{S}/G_c)$

(b) $s\sigma \in L(G)$ (i.e., $f(s\sigma, q_0)$ is defined)

(c) $\Phi[g(s, x_0)](\sigma) = 1$ (i.e., σ is enabled at $g(s, x_0)$)

together imply

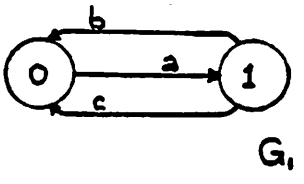
(d) $s\sigma \in L(\underline{S}/G_c)$ (i.e., $g(s\sigma, x_0)$ is defined).

(If $\underline{S}$ is not complete, an obvious modification of Φ makes it complete without changing $L(\underline{S}/G_c)$.)

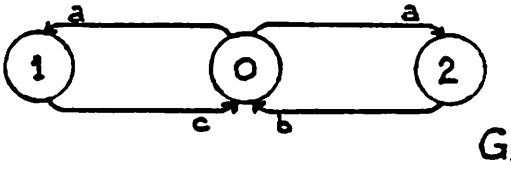
Let $L_m(\underline{S}/G_c)$ be the language marked by $\underline{S}/G_c$.

Example 1.1

Consider the deterministic automaton G_1 and the nondeterministic automaton G_2 sketched below.



G_1



G_2

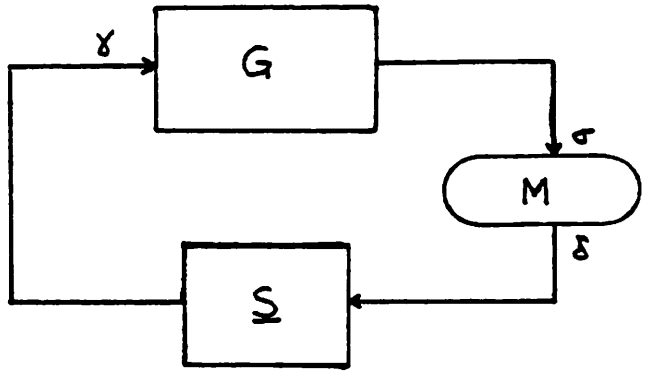
In both automata, state 0 is the initial and the final state, and $\Sigma_c = \{b, c\}$. $L(G_1) = L(G_2) = [a(b+c)]^*$. We can define a supervisor that will restrict the language generated by G_1 and G_2 to $(ab)^*$. This supervisor is a single state automaton with the state feedback map $\Phi(\sigma) = 0$ for $\sigma = c$. This supervisor works for both generators G_1 and G_2 although G_2 would be deadlocked if it changed from state 0 to state 1 upon event a.

This shows that the controller defined in this paper depends on the language generated by an automaton and not on the automaton itself, though different automata have different properties that we do not discuss, such as the possibility of deadlock.

This framework was developed by Ramadge and Wonham [1] for a deterministic generator. We can now state a version of their *supervisor synthesis problem* : Given a deterministic CDEP G_c and $L \subset L(G)$, find a complete supervisor $\underline{S}$ such that $L(\underline{S}/G_c) \subset L$ is as large as possible. We think of L as the set of desirable behaviors and we want to find a supervisor that constrains the behavior of G to L and within this constraint allows the maximum possible freedom. In [1], Ramadge and Wonham show that the synthesis problem always has a solution and in [2] they give a constructive procedure for finding a solution when both $L(G)$ and L are regular. Other related problems are also discussed in [3] and [4].

1.3. Partial observations

It was assumed above that when the supervisor $\underline{S}$ is coupled to G_c it observes all the transitions of G. In some real situations, however, $\underline{S}$ cannot observe all transitions, and in some other cases it cannot distinguish between certain transitions.



These situations can be represented by introducing an observation stage between G and $\underline{S}$ specified by a *mask* or observation function

$$M: \Sigma \to \Delta \cup \{\varepsilon\},$$

where $\delta = M(\sigma)$ is the symbol observed by $\underline{S}$ when the generator makes the transition σ. Thus events in $M^{-1}\{\varepsilon\}$ cannot be seen by $\underline{S}$; and if $M(\sigma) = M(\sigma')$ then $\underline{S}$ cannot distinguish between σ and σ'. M is extended to Σ^* by $M(\sigma_1 \cdots \sigma_n) = M(\sigma_1) \cdots M(\sigma_n)$.

The preceding framework goes through with some minor modifications to take into account the mask M. A supervisor $\underline{S} = (S, \Phi)$ where the automaton $S = (X, \Delta, g, x_0, X_m)$ now has input alphabet Δ; the state feedback map $\Phi: X \to \Gamma$ is as before. The SDEP $\underline{S}/G_c$ is defined by

$$\underline{S}/G_c = A(X \times Q, \Sigma, (g \circ M) \times f_c, (x_0, q_0), X_m \times Q_m)$$

Thus if the relation $h := (g \circ M) \times f_c$, then $h(\sigma, x, q) = \{g(M(\sigma), x)\} \times f_c(\Phi(x), \sigma, q)$ is defined iff $f(\sigma, q)$ is defined, $\Phi(x)(\sigma) = 1$ and $g(M(\sigma), x)$ is defined. (By convention, $g(\varepsilon, x) \equiv x$; however, ε is not a real event.) $\underline{S}$ is complete with respect to G_c provided that for each $s \in \Sigma^*$ and $\sigma \in \Sigma$ the conditions

(a) $s \in L(\underline{S}/G_c)$

(b) $s\sigma \in L(G)$ (i.e., $f(s\sigma, q_0)$ is defined)

(c) $\Phi[g(M(s), x_0)](\sigma) = 1$ (i.e., σ is enabled at $g(M(s), x_0)$)

together imply

(d) $s\sigma \in L(\underline{S}/G_c)$ (i.e., $g(M(s\sigma), x_0)$ is defined).

2. Languages of $\underline{S}/G_c$

Let $L \subset \Sigma^*$. The closure of L, $\overline{L}$, is the set of all prefixes of strings in L, i.e.,

$$\overline{L} := \{s \in \Sigma^* \mid \exists\, t \in \Sigma^*, st \in L\}.$$

L is *closed* if $L = \overline{L}$.

Let G be a generator, Σ_c its controlled events, and $M: \Sigma \to \Delta$ a mask. We call $L(G)$ the *uncontrolled process language*. $L(G)$ is closed, and if G is trim, $L(G) = \overline{L_m(G)}$. Let $\underline{S}$ be a supervisor of G_c, $L(\underline{S}/G_c)$ the language generated by $\underline{S}/G_c$ and $L_m(\underline{S}/G_c)$ the language marked by $\underline{S}/G_c$. Note that $s \in L(\underline{S}/G_c)$ is marked by $\underline{S}/G_c$ iff s is marked by G and $M(s)$ is marked by S. The *language controlled* by $\underline{S}$ is

$$L_c(\underline{S}/G_c) := L(\underline{S}/G_c) \cap L_m(G),$$

and can be interpreted as the set of all sequences generated by $\underline{S}/G_c$ that correspond to completed tasks. Evidently,

$$L_m(\underline{S}/G_c) \subset L_c(\underline{S}/G_c) \subset L_m(G),$$

and, if G is trim,

$$\overline{L_m(\underline{S}/G_c)} \subset \overline{L_c(\underline{S}/G_c)} \subset \overline{L(\underline{S}/G_c)} = L(\underline{S}/G_c) \subset \overline{L_m(G)}.$$

Controllable and recognizable languages

Let $K \subset L \subset \Sigma^*$ and $\Omega \subset \Sigma$. We say that K is

(a) (Ω, L)-*invariant* if

$\sigma \in \Omega, s \in K, s\sigma \in L \Rightarrow s\sigma \in K$, (i.e., $K\Omega \cap L \subset K$)

(b) (M, L)-*recognizable* if

$s \in K, s' \in L, M(s) = M(s') \Rightarrow s' \in K$, (i.e., $L \cap M^{-1}[M(K)] = K$)

(c) (M, Ω, L)-*controllable* if

s, s' in $K, \sigma \in \Omega, s\sigma \in K, s'\sigma \in L, M(s) = M(s') \Rightarrow s'\sigma \in K$.

The following elementary properties follow immediately from these definitions.

Lemma 2.1

Suppose $K_i \subset L$, $i = 1, 2$.

(1) If K_1 and K_2 are (Ω, L)-invariant, then $K_1 \cap K_2$ and $K_1 \cup K_2$ are (Ω, L)-invariant.

(2) If K_1 and K_2 are (M, L)-recognizable, then $K_1 \cap K_2$ and $K_1 \cup K_2$ are (M, L)-recognizable.

(3) If K_1 and K_2 are (M, Ω, L)-controllable, then $K_1 \cap K_2$ is (M, Ω, L)-controllable.

(4) If K_1 is (M, L)-recognizable, then K_1 is (M, Ω, L)-controllable.

Note that if M is one-to-one then K is always (M, L)-recognizable. Example 2.1 below shows that $K_1 \cup K_2$ need not be (M, Ω, L)-controllable.

For the remainder of this section fix a CDEP G_c and a mask $M: \Sigma \to \Delta \cup \{\varepsilon\}$.

Lemma 2.2

Let $K \subset L_m(G)$. There is a complete supervisor $\underline{S}$ such that

(1) $L(\underline{S}/G_c) = L(G)$, and

(2) $L_m(\underline{S}/G_c) = K$

iff K is $(M, L_m(G))$-recognizable. Moreover, if K is regular, then $\underline{S}$ can be selected to be finite state.

Proof

Let $\underline{S} = (S, \Phi)$ satisfy (1) and (2). Let $s \in K$ so that $M(s)$ is marked by S. Let $s' \in L_m(G)$. Then s' is generated by $\underline{S}/G_c$ and marked by G. If $M(s') = M(s)$, then $M(s')$ is also marked by S hence $s' \in L_m(\underline{S}/G_c) = K$ and so K is $(M, L_m(G))$-recognizable.

Conversely, suppose K is $(M, L_m(G))$-recognizable. Let $S = (X, \Delta, g, x_0, X_m)$ be a *recognizer* for $M(K)$, i.e., an automaton such that $g(x_0, d)$ is defined for all $d \in \Delta^*$ and $g(x_0, d) \in X_m$ iff $d \in M(K)$. (If K is regular, then $M(K)$ is regular and X can be selected to be finite.) Let $\underline{S} = (S, \Phi)$ with $\Phi(x) \equiv 1$. Since $\underline{S}$ does not disable any transition in G, $L(\underline{S}/G_c) = L(G)$. A string s will be marked by $\underline{S}/G_c$ iff s is marked by G and $M(s)$ is marked by S iff $s \in L_m(G) \cap M^{-1}[M(K)] = K$.

∎

Corollary 2.1

Let $L \subset L(G)$, and suppose there is a complete supervisor $\underline{S}$ with $L(\underline{S}/G_c) = L$. If K is $(M, L_c(\underline{S}/G_c))$-recognizable, there exists a supervisor $\underline{S}_K$ such that

(1) $L(\underline{S}_K/G_c) = L$, and

(2) $L_m(\underline{S}_K/G_c) = K$.

Moreover, if K is regular, then $\underline{S}_K$ can be selected to be finite state.

Proof

Let $T = (Y, \Delta, h, y_0, Y_m)$ be a recognizer for $M(K)$. If K is regular, T can be selected to be finite state. Let $\underline{S} = (S, \Phi)$ be a complete supervisor for which $L(\underline{S}/G_c) = L$ with $S = (X, \Delta, g, x_0, X)$. Define the supervisor,

$$\underline{S}_K = (S', \Phi'), \quad S' = (X', \Delta, g', x'_0, X'_m)$$

by

$$X' = X \times Y, \quad g'(\delta, x, y) = (g(\delta, x), h(\delta, y)), \quad x'_0 = (x_0, y_0),$$

$$X'_m = X \times Y_m, \quad \Phi'(x, y) = \Phi(x).$$

Since the control action of $\underline{S}_K$ is the same as that of $\underline{S}$, $L(\underline{S}_K/G_c) = L$. Furthermore, $g'(\delta, x, y)$ is defined iff $g(\delta, x)$ is defined, so $\underline{S}_K$ is complete with respect to G_c. Finally, the language marked by $\underline{S}_K/G_c$ is equal to $L \cap L_m(G) \cap M^{-1}[M(K)] = L_c(\underline{S}/G_c) \cap M^{-1}[M(K)] = K$ because K is $(M, L_c(\underline{S}/G_c))$-recognizable. ∎

Lemma 2.3

Let $K \subset L(G)$. There exists a complete supervisor $\underline{S}$ such that $L(\underline{S}/G_c) = K$ iff

(1) K is closed,

(2) K is $(\Sigma_u, L(G))$-invariant, and

(3) K is $(M, \Sigma_c, L(G))$-controllable.

Moreover, if K is regular, then $\underline{S}$ can be selected to be finite state.

Lemma 2.3 is proved in Propositions 2.1-2.3 below. Let

$$G = (Q, \Sigma, f, q_0, Q_m).$$

Proposition 2.1

If $\underline{S}$ is a complete supervisor such that $L(\underline{S}/G_c) = K$, then K satisfies (1)-(3) of Lemma 2.3.

Proof

Let $\underline{S} = (S, \Phi)$ with $S = (X, \Delta, g, x_0, X_m)$. Since $L(\underline{S}/G_c)$ is closed, (1) holds.

Suppose $s \in K$ and $\sigma \in \Sigma_u$. Then $g(M(s), x_0)$ is defined and $\Phi[g(M(s), x_0)](\sigma) = 1$. Suppose $s\sigma \in L(G)$, so $f(s\sigma, q_0)$ is defined. Since $\underline{S}$ is complete, $s\sigma \in K$, so (2) holds. Finally suppose s, s' are in K, $M(s) = M(s')$, $\sigma \in \Sigma_c$, and $s\sigma \in K$. Then $g(M(s\sigma), x_0) = g(M(s'\sigma), x_0)$ is defined, $\Phi[g(M(s), x_0)](\sigma) = \Phi[g(M(s'), x_0)](\sigma) = 1$. Suppose $s'\sigma \in L(G)$ so that $f(s'\sigma, q_0)$ is defined. Then $s'\sigma \in K$ (because $\underline{S}$ is complete) so (3) holds.

For the remainder of the proof fix $K \subset L(G)$ such that (1)-(3) hold. For $d \in \Delta^*$, let

$$\Sigma(d) := \{\sigma \in \Sigma_c \mid \exists\, s \in K,\ M(s) = d,\ s\sigma \in K\}. \tag{2.1}$$

Proposition 2.2

Let $\underline{S} = (S, \Phi)$ be a complete supervisor such that

(1) $g(d, x_0)$ is defined iff $d \in M(K)$, and

(2) $\Phi[g(d, x_0)](\sigma) = 1$ iff $\sigma \in \Sigma_u \cup \Sigma(d)$.

Then $L(\underline{S}/G_c) = K$.

Proof

Suppose $s = \sigma_1 \cdots \sigma_m \in K \subset L(G)$. Then $f(s, q_0)$ is defined. It is easily checked from properties (1) and (2) that for each $n < m$, $g(M(\sigma_1 \cdots \sigma_n), x_0)$ is defined and $\Phi[g(M(\sigma_1 \cdots \sigma_n), x_0)](\sigma_{n+1}) = 1$. It follows that $s \in L(\underline{S}/G_c)$, hence $K \subset L(\underline{S}/G_c)$.

We prove $L(\underline{S}/G_c) \subset K$ by induction on the length $|s|$ of $s \in L(\underline{S}/G_c)$. Suppose $|s| = 1$, $s = \sigma \in L(\underline{S}/G_c)$. Write $\sigma = \varepsilon\sigma$. Since K is closed, $\varepsilon \in K$. By (2), $\sigma \in \Sigma(\varepsilon) \cup \Sigma_u$. If $\sigma \in \Sigma_u$, then $\sigma \in K$ because K is $(\Sigma_u, L(G))$-invariant. If $\sigma \in \Sigma(\varepsilon)$, then by (2.1) there exists $s'\sigma \in K$, $M(s') = \varepsilon$. Since K is $(M, \Sigma_c, L(G))$-controllable, it follows that $\sigma \in K$. Now suppose the assertion is true for strings of length l. Let $|s| = l$, $s \in K$, and $s\sigma \in L(\underline{S}/G_c)$. Let $d = M(s)$, then $\sigma \in \Sigma(d) \cup \Sigma_u$. If $\sigma \in \Sigma_u$, then $s\sigma \in K$ because K is $(\Sigma_u, L(G))$-invariant. If $\sigma \in \Sigma(d)$, then by (2.1) there exists $s' \in K$ such that $s'\sigma \in K$ and $M(s') = d$, and then $s\sigma \in K$ since K is $(M, \Sigma_c, L(G))$-controllable.

The proof of Proposition 2.2 yields the following interesting corollary.

Corollary 2.2

Let K be any closed sublanguage of $L(G)$. Let $\underline{S}$ be any complete supervisor satisfying (1) and (2) of Proposition 2.2. Then, $L(\underline{S}/G_c)$ is the smallest closed,

$(\Sigma_u, L(G))$-invariant, and $(M, \Sigma_c, L(G))$-controllable language containing K.

There is a trivial supervisor $\underline{S} = (S, \Phi)$ that satisfies (1), (2) of Proposition 2.2. Take $S = (X, \Delta, g, x_0, X)$ with $X = M(K)$, $g(\delta, d) = d\delta$ if $d\delta \in M(K)$, and $g(\delta, d)$ is undefined if $d\delta \notin M(K)$, $x_0 = \varepsilon$. Now take $\Phi(d)(\sigma) = 1$ iff $\sigma \in \Sigma(d) \cup \Sigma_u$. The states of this supervisor "memorize" the entire sequence of observations made by the supervisor. If $M(K)$ is infinite, this supervisor has infinite states, even if $M(K)$ is regular. The supervisor constructed below overcomes this defect.

Let $\hat{S} = (\hat{X}, \Sigma, \hat{g}, \hat{x}_0, \hat{X})$ be a generator for K, i.e., $\hat{g}(s, \hat{x}_0)$ is defined iff $s \in K$. (If K is regular, we can select $\hat{X}$ to be finite.) Let X be the set of all nonempty subsets of $\hat{X}$ and define the automaton

$$S := (X, \Delta, g, x_0, X) \tag{2.2}$$

as follows:

$$g(\delta, x) := \begin{cases} \{\hat{g}(s, \hat{x}) \mid \hat{x} \in x, M(s) = \delta\}, & \text{if this is nonempty} \\ \text{undefined, otherwise;} \end{cases} \tag{2.3}$$

$$x_0 := \{\hat{g}(s, \hat{x}_0) \mid M(s) = \varepsilon\}. \tag{2.4}$$

Note that X is finite if $\hat{X}$ is finite.

Remark

The following comment may provide some insight into the relationship between $\hat{S}$, S, and the closed loop system. Suppose we have found a supervisor $\underline{S}$ such that $L(\underline{S}/G_c) = K$. Then the automaton $\hat{S}$ is an accurate representation of the closed loop system $\underline{S}/G_c$. However, since the supervisor can observe the transitions of $\hat{S}$ (or $\underline{S}/G_c$) only through M, it can only partially reconstruct the internal state $\hat{x}$ of $\hat{S}$. Proposition 2.3 says that the states of S provide this partial reconstruction.

Proposition 2.3

Let $d \in \Delta^*$. Then $g(d, x_0) = \{\hat{g}(s, \hat{x}_0) \mid M(s) = d\}$ if this is nonempty, and $g(d, x_0)$ is undefined otherwise.

Proof

We use induction on the length of d. If $d = \varepsilon$,

$$g(\varepsilon, x_0) := x_0 = \{\hat{g}(s, \hat{x}_0) \mid M(s) = \varepsilon\} \text{ by (2.4)},$$

so the assertion is true for $|d| = 0$. Suppose it is true for d, $g(d, x_0) = x$ is

defined, and consider the string $d\delta$. Then

$$x = \{\hat{g}(s, \hat{x}_0) \mid M(s) = d\} \neq \phi,$$

and so

$$\begin{aligned} g(d\delta, x_0) &= g(\delta, x) \\ &= \{\hat{g}(s', \hat{x}) \mid \hat{x} \in x, M(s') = \delta\} \text{ by (2.3)} \\ &= \{\hat{g}(ss', \hat{x}_0) \mid M(s) = d, M(s') = \delta\} \text{ by induction hypothesis} \\ &= \{\hat{g}(t, \hat{x}_0) \mid M(t) = d\delta\} \end{aligned}$$

as required. ∎

Since $\hat{S}$ generates K, Proposition 2.3 implies that $g(d, x_0)$ is defined iff $d \in M(K)$. For $x \in X$, let

$$\Sigma(x) := \{\sigma \in \Sigma_c \mid \exists\, \hat{x} \in x, \hat{g}(\sigma, \hat{x}) \text{ is defined}\}.$$

Recall the definition (2.1) of $\Sigma(d)$.

Proposition 2.4
If $g(d, x_0) = x$, then $\Sigma(x) = \Sigma(d)$.

Proof
Suppose $\hat{x} \in x$ and $\hat{g}(\sigma, \hat{x})$ is defined, so that $\sigma \in \Sigma(x)$. By Proposition 2.3, there exists s with $M(s) = d$ and $\hat{x} = \hat{g}(s, \hat{x}_0)$. Hence $\hat{g}(s\sigma, \hat{x}_0) = \hat{g}(\sigma, \hat{x})$ is defined and so $s\sigma \in K$. This implies that $\sigma \in \Sigma(d)$. Conversely, let $\sigma \in \Sigma(d)$, so there exists $s\sigma \in K$ with $M(s) = d$. But then $\hat{g}(s, \hat{x}_0) = \hat{x} \in x$ and $\hat{g}(\sigma, \hat{x})$ is defined, so that $\sigma \in \Sigma(x)$. ∎

Proof of Lemma 2.3
The necessity follows from Proposition 2.1. To prove sufficiency consider the supervisor $\underline{S} = (S, \Phi)$ where S is given by (2.2) and

$$\Phi(x)(\sigma) = 1 \text{ iff } \sigma \in \Sigma(x) \cup \Sigma_u.$$

It is easily seen that $\underline{S}$ is complete and by Propositions 2.2 and 2.4, $L(\underline{S}/G_c) = K$. ∎

Lemma 2.4
If there is a supervisor $\underline{S}$ with $L_m(\underline{S}/G_c) = K$, then K is $(M, L_c(\underline{S}/G_c))$-recognizable.

Proof

The proof follows from the definitions of $L_c(\underline{S}/G_c)$ and (M, L)-recognizability. ∎

Theorem 2.1

Let $K_1 \subset L_m(G)$, $K_2 \subset L_m(G)$, and $K_3 \subset L(G)$ with $K_3 \neq \phi$. Then there exists a complete decentralized supervisor $\underline{S}$ such that

(1) $L_m(\underline{S}/G_c) = K_1$,

(2) $L_c(\underline{S}/G_c) = K_2$, and

(3) $L(\underline{S}/G_c) = K_3$

iff

(a) K_1 is (M, K_2)-recognizable,

(b) $K_2 = K_3 \cap L_m(G)$,

(c) K_3 is closed, $(\Sigma_u, L(G))$-invariant, and $(M, \Sigma_c, L(G))$-controllable.

Proof

Suppose (1)-(3) hold. Then (a) follows from Lemma 2.4, (b) from the definition of $L_c(\underline{S}/G_c)$, and (c) from Lemma 2.3. Conversely, suppose (a)-(c) hold. By Corollary 2.1, it is enough to construct a complete supervisor $\underline{S}$ such that $L(\underline{S}/G_c) = K_3$. Lemma 2.3 states that this is possible. ∎

Example 2.1

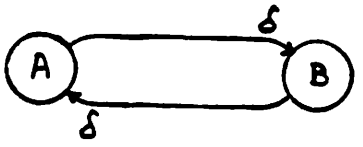
Let G be a generator with a single state, $X = X_m = \{0\}$, and two transitions, $\Sigma = \Sigma_c = \{a, b\}$. Then $L_m(G) = L(G) = \Sigma^*$. Let $M(a) = M(b) = \delta$, so a controller cannot distinguish between events a and b. Let

$$K := \text{closure of } \{(ab)^n \mid n \geq 0\}.$$

Then $M(K) = \{\delta\}^*$, so $M^{-1}[M(K)] = \Sigma^*$. Hence K is not $(M, L_m(G))$-recognizable. By Lemma 2.2 there is *no* supervisor $\underline{S}$ such that $L(\underline{S}/G_c) = \Sigma^*$ and $L_m(\underline{S}/G_c) = K$.

However, K is $(M, \Sigma_c, L(G))$-controllable because if s, s' are in K with $M(s) = M(s')$, then $s = s'$. Furthermore, since $\Sigma_u = \phi$, K is $(\Sigma_u, L(G))$-invariant. By Lemma 2.3 there exists a supervisor $\underline{S} = (S, \Phi)$ such that $L(\underline{S}/G_c) = K$. Such a supervisor can be built using an automaton with two states, A and B,

initial state A, and the state feedback map Φ given by $\Phi(A)(\sigma) = 1$ iff $\sigma = a$, and $\Phi(B)(\sigma) = 1$ iff $\sigma = b$.



Continuing with this example we see that

$$K := \text{closure of } \{(ba)^n \mid n \geq 0\}$$

is also $(M, \Sigma_c, L(G))$-controllable. However,

$$\tilde{K} := K \cup K = \text{closure of } \{(ab)^n, (ba)^n \mid n \geq 0\}$$

is not $(M, \Sigma_c, L(G))$-controllable because the strings $s = a$, ab are in K, b is in K, $M(a) = M(b)$, but $bb \notin \tilde{K}$. Hence $\tilde{K}$ is not $(M, \Sigma_c, L(G))$-controllable and there is *no* supervisor S such that $L(S/G_c) = \tilde{K}$.

Remark

This example shows that the set of $(M, \Sigma_c, L(G))$-controllable sublanguages of $L(G)$ is not closed under union. This has implications for the supervisor synthesis problem posed in § 1.2. Recall that the problem is to find a supervisor S such that $L(S/G_c)$ is as large a subset of a specified $L \subset L(G)$ as possible. Since the family of languages $L(S/G_c)$ may not be closed under union, there may not exist such a largest subset. Indeed in the example above take $L = \tilde{K}$. It is possible to retain the closure property in the partial information case if we allow "nondeterministic" supervisors; however, this seems unreasonable in practical applications.

Wonham [5] considers the case of partial observations where the mask M is a projection, i.e., there is $\Sigma_M \subset \Sigma$ such that $M(\sigma) = \sigma$ if $\sigma \in \Sigma_M$ and $M(\sigma) = \varepsilon$ if $\sigma \notin \Sigma_M$. His results include Lemma 2.3 for the case where K is $(M, L(G))$-recognizable (in our notation); such K is automatically $(M, \Sigma_c, L(G))$-controllable by Lemma 2.1.

3. A supervisor synthesis problem for a special class of languages

Recall the supervisor synthesis problem stated in section 1.2: Given a CDEP G_c and $L \subset L(G)$, find a complete supervisor S such that $L(S/G_c) \subset L$ is as large as possible. Example 2.1 above demonstrates that the set of $(M, \Sigma_c, L(G))$–controllable sublanguages of $L(G)$ may not be closed under union

and therefore the desired $L(\underline{S}/\,G_c) \subset L$ may not exist. In this section, we consider a subset that is closed under union: the set of languages that are $(M,L(G))$–recognizable. If we restrict ourselves to languages in this subset, a solution to the supervisor synthesis problem does exist. Furthermore, if L and $L(G)$ are regular, the solution is regular and the procedure for determining it is effectively computable.

Lemma 2.1 states that if $K \subset L(G)$ is $(M,L(G))$–recognizable, then K is also $(M, \Sigma_c, L(G))$–controllable. For a closed language $L \subset L(G)$, define the class of languages

$$\underline{\mathcal{D}}(L) = \{K \subset L \mid K \text{ is closed,}$$
$$(\Sigma_u, L(G))\text{–invariant and } (M,L(G))\text{–recognizable}\}.$$

By Lemma 2.1, $\underline{\mathcal{D}}(L)$ is closed under union. Therefore, $\sup \underline{\mathcal{D}}(L)$ exists and is equal to

$$\cup \{K \subset L \mid K \text{ is closed,}$$
$$(\Sigma_u, L(G))\text{–invariant and } (M,L(G))\text{–recognizable}\}.$$

Suppose we have a procedure P that accepts as input regular languages and produces a regular language as output. We say that P is effectively computable if given the finite state automata that generate the input languages, we can construct in a finite number of steps a finite state automaton that generates the output language.

We now describe a method for computing a sublanguage K_R of L. This procedure consists of three major steps. Afterwards we prove that $K_R = \sup \underline{\mathcal{D}}(L)$.

Summary of procedure

Given: L and $L(G)$ both regular and closed, and $L \subset L(G)$.

Step 1 Compute $M(L(G))$, $M(L)$, and let $\Delta_u = M(\Sigma_u)$.

Step 2 Using the method developed by Ramadge and Wonham in [2], compute K, the largest sublanguage of $M(L)$ that is closed and $(\Delta_u, M(L(G)))$–invariant.

Step 3 Compute $K_R = M^{-1}(K) \cap L(G)$.

A detailed description of this procedure follows.

Step 1

Let $\hat{S} = (\hat{X}, \Sigma, \hat{f}, \hat{x}_0, \hat{X})$ and $\hat{T} = (\hat{Y}, \Sigma, \hat{g}, \hat{y}_0, \hat{Y})$ be two finite state automata

that generate $L(G)$ and L respectively. Let X and Y be the sets of all nonempty subsets of $\hat{X}$ and $\hat{Y}$ respectively. Define the two deterministic automata

$$S := (X, \Delta, f, x_0, X) \text{ and } T := (Y, \Delta, g, y_0, Y)$$

as follows:

$$f(\delta, x) := \begin{cases} \{\hat{f}(s, \hat{x}) \mid \hat{x} \in x, M(s) = \delta\}, & \text{if this is nonempty} \\ \text{undefined, otherwise;} \end{cases}$$

$$x_0 := \{\hat{f}(s, \hat{x}_0) \mid M(s) = \varepsilon\}.$$

and

$$g(\delta, y) := \begin{cases} \{\hat{g}(s, \hat{y}) \mid \hat{y} \in y, M(s) = \delta\}, & \text{if this is nonempty} \\ \text{undefined, otherwise;} \end{cases}$$

$$y_0 := \{\hat{g}(s, \hat{y}_0) \mid M(s) = \varepsilon\}.$$

S and T are finite state automata that generate $M(L(G))$ and $M(L)$ respectively. Since $|X| \leq 2^{|\hat{X}|}$ and $|Y| \leq 2^{|\hat{Y}|}$ (where $|\cdot|$ denotes cardinality), this procedure is effectively computable. Proposition 3.1 follows immediately from the above construction.

Proposition 3.1

$M(L(G))$ and $M(L)$ are regular, closed, and the procedure to find them is effectively computable.

Step 2

In [2], Ramadge and Wonham give a procedure that effectively computes the largest closed and $(\Sigma_u, L(G))$–invariant sublanguage of a given language L. We use this procedure to compute K, the largest closed and $(\Delta_u, M(L(G)))$–invariant sublanguage of $M(L)$. Proposition 3.2 is a direct consequence of the procedure described in [2].

Proposition 3.2

K is regular, closed and can be effectively computed.

Step 3

Let $V = (Z, \Delta, h, z_0, Z)$ be a deterministic finite state automaton that generates K. We construct a deterministic finite state automaton $\hat{V} := (Z, \Sigma, \hat{h}, z_0, Z)$ that generates $M^{-1}(K)$. $\hat{V}$ is defined as follows:

$$\hat{h}(\sigma, z) := \begin{cases} h(M(\sigma), z), & \text{if this is defined} \\ \text{undefined, otherwise.} \end{cases}$$

By convention, $h(\varepsilon, z) = z$.

We now construct the deterministic automaton

$$W := (\hat{X} \times Z, \Sigma, p, (\hat{x}_0, z_0), \hat{X} \times Z)$$

where

$$p(\sigma, (\hat{x}, z)) := \begin{cases} (\hat{f}(\sigma, \hat{x}), \hat{h}(\sigma, z)), & \text{if both are defined} \\ \text{undefined, otherwise.} \end{cases}$$

W generates $K_R = M^{-1}(K) \cap L(G)$. Proposition 3.3 follows from the above construction.

Proposition 3.3

K_R is regular, closed, and can be effectively computed.

Proposition 3.4

K_R is $(\Sigma_u, L(G))$–invariant and $(M, L(G))$–recognizable.

Proof

Suppose $s \in K_R$, $\sigma \in \Sigma_u$, and $s\sigma \in L(G)$. Let $M(s)M(\sigma) = d\delta$, $\delta \in \Delta_u$. Since $s \in K_R$, $M(s) = d \in K$; since $s\sigma \in L(G)$, $d\delta \in M(L(G))$. Since K is $(\Delta_u, M(L(G)))$–invariant, $d\delta \in K$, so $s\sigma \in M^{-1}(K)$. Hence $s\sigma \in K_R$ and K_R is $(\Sigma_u, L(G))$–invariant.

Suppose s, s' in $L(G)$ with $M(s) = M(s') = d$ and $s \in K_R$. Then $d \in K \subset M(L(G))$, so $s' \in M^{-1}(K)$, hence $s' \in K_R$ and K_R is $(M, L(G))$–recognizable.

∎

Proposition 3.5

K_R is the largest sublanguage of L that is closed, $(\Sigma_u, L(G))$–invariant, and $(M, L(G))$–recognizable.

Proof

Let K'_R be a sublanguage of L that is closed, $(\Sigma_u, L(G))$–invariant, and $(M, L(G))$–recognizable. We show that $K'_R \subset K_R$.

For all $s \in K'_R$, if $\sigma \in \Sigma_u$ and $s\sigma \in L(G)$, then $s\sigma \in K'_R$. Also $s \in L \subset L(G)$, since $K'_R \subset L \subset L(G)$. Let $d = M(s)$, then $d \in M(K'_R) \subset M(L) \subset M(L(G))$. For δ in Δ_u such that $d\delta \in M(L(G))$, $s\sigma \in L(G)$ where $\sigma \in M^{-1}(\delta) \cap \Sigma_u$. Since K'_R is $(\Sigma_u, L(G))$–invariant, $s\sigma \in K'_R$ and therefore $M(s\sigma) = d\delta \in M(K'_R)$. Thus $M(K'_R)$ is $(\Delta_u, M(L(G)))$–invariant. But K is the largest sublanguage of $M(L)$ that is $(\Delta_u, M(L(G)))$–invariant, therefore $M(K'_R) \subset K$, and $d \in K$. This implies that $s \in M^{-1}(d) \subset M^{-1}(K)$, and since $s \in L(G)$, $s \in K_R$.

The theorem below follows from propositions 3.1 through 3.5.

Theorem 3.1

(1) $K_R = \sup \underline{D}(L)$

(2) K_R is regular and can be effectively computed.

4. Decentralized control

In some situations the uncontrolled generator G represents a collection of coupled subsystems that are in different locations. Physical considerations may then lead us to control G by a collection of supervisors, one for each subsystem. Each supervisor makes (possibly different) observations of events in G and controls different transitions of G. We propose a model for such situations.

4.1. Decentralized supervisor

Let $G_c = (Q, \Gamma \times \Sigma, f_c, q_0, Q_m)$ be a CDEP with controlled events Σ_c. We are also given subsets $\Sigma_{1,c}, \cdots, \Sigma_{n,c}$ (not necessarily disjoint) with $\Sigma_c = \bigcup_i \Sigma_{i,c}$ and n masks $M_i: \Sigma \to \Delta_i \cup \{\varepsilon\}$, $i = 1, \cdots, n$. A ***decentralized supervisor*** is a collection

$$\{\underline{S}_i\} = \{\underline{S}_i = (S_i, \Phi_i), i = 1, \cdots, n\},$$

where S_i is a deterministic automaton

$$S_i = (X_i, \Delta_i, g_i, x_{i,0}, X_{i,m}),$$

and Φ_i is a state feedback map $\Phi_i: X_i \to \Gamma$ such that

$$\Phi_i(x_i)(\sigma) = 1 \text{ if } \sigma \notin \Sigma_{i,c}.$$

From the decentralized supervisor $\{\underline{S}_i\}$ we construct a (standard) supervisor $\underline{S} = (S, \Phi)$ as follows: $S = (S_1 \times \cdots \times S_n)$ is the product automaton,

$$\begin{aligned} S &= (X, \Delta, g, x_0, X_m) \\ &= (X_1 \times \cdots \times X_n, \Delta_1 \times \cdots \times \Delta_n, g_1 \times \cdots \times g_n, \\ &\qquad (x_{1,0}, \cdots, x_{n,0}), X_{1,m} \times \cdots \times X_{n,m}), \end{aligned}$$

and $\Phi: X \to \Gamma$ is given by

$$\Phi(x_1, \cdots, x_n)(\sigma) = 1 \text{ iff } \forall i, \Phi_i(x_i)(\sigma) = 1.$$

Thus $\underline{S}_i$ can only disable events in $\Sigma_{i,c}$. Note that $\underline{S}$ observes transitions in G

through the mask $M: \Sigma \to \Delta \cup \{\varepsilon\}$, where $M(\sigma) = (M_1(\sigma), \cdots, M_n(\sigma))$. From G_c and $\underline{S}$ we construct the SDEP

$$\underline{S}/G_c := A(X \times Q, \Sigma, (g \circ M) \times f_c, (x_0, q_0), X_m \times Q_m).$$

Decentralization of the supervisor places a restriction on the languages generated by the SDEP because each component of the supervisor controls a subset of the controllable events. Thus, the class of languages $L(\underline{S}/G_c)$ in the decentralized case is smaller than the class of languages $L(\underline{S}/G_c)$ in the centralized case with the same mask.

With an obvious abuse of notation we sometimes identify $\underline{S}$ and the decentralized supervisor $\{\underline{S}_i\}$. Say that $\{\underline{S}_i\}$ is complete with respect to G_c provided that for $s \in \Sigma^*$ and $\sigma \in \Sigma$ the conditions

(a) $s \in L(\underline{S}/G_c)$

(b) $s\sigma \in L(G)$

(c) $\Phi[g(M(s), x_0)](\sigma) = 1$ (i.e., $\forall\, i,\ \Phi_i[g_i(M_i(s), x_{i,0})](\sigma) = 1$)

together imply

(d) $s\sigma \in L(\underline{S}/G_c)$ (i.e., $\forall\, i,\ g_i(M_i(s\sigma), x_{i,0})$ is defined).

As before, $L_m(\underline{S}/G_c)$ is the language marked by $\underline{S}/G_c$. Note that since S is the product automaton, $s \in L(\underline{S}/G_c)$ is marked iff s is marked by G and for every i, $M_i(s)$ is marked by S_i. Lemma 4.1 is the analog of Lemma 2.2 and it is proved in the same way.

Lemma 4.1

Let $K \subset L(G)$. There is a complete decentralized supervisor $\underline{S} = \{\underline{S}_i\}$ such that

(1) $L(\underline{S}/G_c) = L(G)$ and

(2) $L_m(\underline{S}/G_c) = K$

iff K is $(M, L_m(G))$-recognizable. Moreover, if K is regular, then $\underline{S}$ can be selected to be finite state.

Corollary 4.1

Let $L \subset L(G)$ and suppose there is a complete decentralized supervisor $\underline{S}$ with $L(\underline{S}/G_c) = L$. If K is $(M, L_c(\underline{S}/G_c))$-recognizable, there exists a complete decentralized supervisor $\underline{S}_K$ such that

(1) $L(\underline{S}_K/G_c) = L$, and

(2) $L_m(\underline{S}_K/G_c) = K$.

Proof

Let $T_i = (Y_i, \Delta_i, h_i, y_{i,0}, Y_{i,m})$ be a recognizer for $M_i(K)$, and let $\underline{S} = \{S_i, \Phi_i\}$ be a complete supervisor for which $L(\underline{S}/G_c) = L$. Define the supervisor

$\underline{S}_K = \{S'_i, \Phi'_i\}$ where S'_i is defined in terms of (S_i, Φ_i) and T_i in the same way as S' is defined in terms of (S, Φ) and T in Corollary 2.1. The rest of the argument is also similar.

Associate with each $\sigma \in \Sigma_c$ a set $I_c(\sigma) \subset \{1, 2, \cdots, n\}$ where $I_c(\sigma) = \{i \mid \sigma \in \Sigma_{i,c}\}$. We will say that $K \subset L(G)$ is $(\{M_i\}, \{\Sigma_{i,c}\}, L(G))$-*controllable* if for $\sigma \in \Sigma_c$, a sequence $\{s_i\}$ of (not necessarily distinct) strings in K indexed by $i \in I_c(\sigma)$, and $s' \in K$,
(1) $s_i\sigma \in K$ for all $i \in I_c(\sigma)$,
(2) $s'\sigma \in L(G)$,
(3) $M_i(s_i) = M_i(s')$ for all $i \in I_c(\sigma)$
together imply

$$s'\sigma \in K.$$

In the case where the $\Sigma_{i,c}$ are disjoint, K is $(\{M_i\}, \{\Sigma_{i,c}\}, L(G))$-controllable iff K is $(M_i, \Sigma_{i,c}, L(G))$-controllable for each i.

The next result is the analog of Lemma 2.3.

Lemma 4.2
There exists a complete decentralized supervisor $\underline{S} = \{\underline{S}_i\}$ such that $L(\underline{S}/G_c) = K$ iff
(1) K is closed,
(2) K is $(\Sigma_u, L(G))$-invariant, and
(3) K is $(\{M_i\}, \{\Sigma_{i,c}\}, L(G))$-controllable.

The proof follows the same lines as the proof of Lemma 2.3. We give the steps and indicate the key differences.

Proposition 4.1
If $\underline{S} = \{S_i, \Phi_i\}$ is a complete decentralized supervisor such that $L(\underline{S}/G_c) = K$, then K satisfies (1)-(3) of Lemma 4.2.

This is proved in the same way as Proposition 2.1 taking into account the requirement that $\Phi_i(x_i)(\sigma) = 1$ if $\sigma \notin \Sigma_{i,c}$.

Now fix $K \subset L(G)$ such that (1)-(3) hold. For $d_i \in \Delta_i^*$ let

$$\Sigma_i(d_i) := \{\sigma \in \Sigma_{i,c} \mid \exists\, s \in K,\ M_i(s) = d_i,\ s\sigma \in K\}.$$

Proposition 4.2
Let $\underline{S} = \{(S_i, \Phi_i)\}$ be a complete decentralized supervisor such that
(1) $g_i(d_i, x_{i,0})$ is defined iff $d_i \in M_i(K)$, and

(2) $\Phi_i[g_i(d_i, x_{i,0})](\sigma) = 1$ iff $\sigma \in (\Sigma - \Sigma_{i,c}) \cup \Sigma_i(d_i)$.
Then $L(\underline{S}/\, G_c) = K$.

This is proved in virtually the same way as Proposition 2.2.

Let $\hat{S} = (\hat{X}, \Sigma, \hat{g}, \hat{x}_0, \hat{X})$ be a generator for K, and let X_i be the set of nonempty subsets of $\hat{X}$. Define

$$S_i := (X_i, \Delta_i, g_i, x_{i,0}, X_i)$$

as follows:

$$g_i(\delta_i, x_i) := \begin{cases} \{\hat{g}(s, \hat{x}) \mid \hat{x} \in x_i,\ M_i(s) = \delta_i\}, & \text{if this is nonempty} \\ \text{undefined,} & \text{otherwise;} \end{cases}$$

$$x_{i,0} := \{\hat{g}(s, \hat{x}_0) \mid M_i(s) = \varepsilon\}.$$

As in Proposition 2.3 one can show that for all $d_i \in \Delta_i^*$

$$g_i(d_i, x_{i,0}) := \begin{cases} \{\hat{g}(s, \hat{x}_0) \mid M_i(s) = d_i\}, & \text{if this is nonempty} \\ \text{undefined,} & \text{otherwise.} \end{cases}$$

Since S generates K, $g_i(d_i, x_{i,0})$ is defined iff $d_i \in M_i(K)$. Let

$$\Sigma_i(x_i) := \{\sigma \in \Sigma_{i,c} \mid \exists\ \hat{x} \in x_i,\ \hat{g}(\sigma, \hat{x}) \text{ is defined}\}.$$

As in Proposition 2.4 one can show that

$$g(d_i, x_{i,0}) = x_i \ \Rightarrow\ \Sigma_i(x_i) = \Sigma_i(d_i).$$

Finally define $\Phi_i: X_i \to \Gamma$ by

$$\Phi_i(x_i) = 1 \text{ iff } \sigma \in (\Sigma - \Sigma_{i,c}) \cup \Sigma_i(x_i).$$

Then $\underline{S} = \{(S_i, \Phi_i)\}$ is a complete supervisor such that $L(\underline{S}/\, G_c) = K$, completing the proof of Lemma 4.2.

The next result follows immediately from the definitions of $L_c(\underline{S}/\, G_c)$ and (M, L)-recognizability.

Lemma 4.3
If there is a complete decentralized supervisor $\underline{S}$ with $L_m(\underline{S}/\, G_c) = K$, then K is $(M, L_c(\underline{S}/\, G_c))$-recognizable.

The following result is proved in the same way as Theorem 2.1.

Theorem 4.1 Let $K_1 \subset L_m(G)$, $K_2 \subset L_m(G)$ and $K_3 \subset L(G)$ with $K_3 \neq \phi$. Then there exists a complete decentralized supervisor $\underline{S}$ such that
(1) $L_m(\underline{S}/\, G_c) = K_1$,
(2) $L_c(\underline{S}/\, G_c) = K_2$, and

(3) $L(\underline{S}/\, G_c) = K_3$,

iff

(a) K_1 is (M, K_2)-recognizable,

(b) $K_2 = K_3 \cap L_m(G)$,

(c) K_3 is closed, $(\Sigma_u, L(G))$-invariant and $(\{M_i\}, \{\Sigma_{i,c}\}, L(G))$-controllable.

5. Generator connections

Interconnected subsystems can be represented by modeling each subsystem as a generator and describing the connections between them in such a way that the "global" generator can be obtained in a mechanical way. We describe one such connection used by Hoare [6] and Merlin and Bochmann [7].

Let $G_i = (Q_i, \Sigma_i, f_i, q_{i,0}, Q_{i,m})$, $i = 1, \cdots, n$ be generators. Let $\Sigma = \bigcup_i \Sigma_i$, and for each σ let $I(\sigma) = \{i \mid \sigma \in \Sigma_i\}$. The generator $G := G_1 // \cdots // G_n$ is defined as $G = (Q, \Sigma, f, q_0, Q_m)$, where $Q = Q_1 \times \cdots \times Q_n$, $q_0 = (q_{1,0}, \cdots, q_{n,0})$, $Q_m = Q_{1,m} \times \cdots \times Q_{n,m}$, and $f = f_1 // \cdots // f_n$ is given by

$f(\sigma, q)$ is undefined if $f_i(\sigma, q_i)$ is undefined for some $i \in I(\sigma)$,

$q' \in f(\sigma, q)$ with $q'_i \in f_i(\sigma, q_i)$ for all $i \in I(\sigma)$, and $q'_i = q_i$ for $i \notin I(\sigma)$.

Thus, in the connected machine G, events that are common to more than one G_i must occur simultaneously. In particular, if the Σ_i are all disjoint G is the *shuffle* of the G_i, see [1]. Lemma 5.1 gives a simple characterization of the language generated by G. Let M_i denote the projection of Σ on Σ_i, i.e., $M_i(\sigma) = \sigma$ if $\sigma \in \Sigma_i$ and $M_i(\sigma) = \varepsilon$ if $\sigma \notin \Sigma_i$.

Lemma 5.1

Let $G = G_1 // \cdots // G_n$. Then

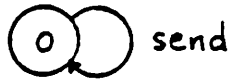
(1) $L(G) = \{s \in \Sigma^* \mid \forall\, i,\ M_i(s) \in L(G_i)\}$,

(2) $L_m(G) = \{s \in \Sigma^* \mid \forall\, i,\ M_i(s) \in L_m(G_i)\}$.

6. The alternating bit protocol

We consider a simple example of a protocol for a communication system represented as an interconnection of six machines: sender S, transmitter buffer TB, transmitter T, channel C, receiver buffer RB, and receiver R.

Sender



S has only one state, 0, and one transition *send*. Transition *send* means that a new message is written in the buffer, and by doing so, the sender requests its transmission from the transmitter.

Transmitter buffer

TB is a single buffer whose state d_1 takes values 0 or 1, where 0 corresponds to the buffer being empty and 1 corresponds to the buffer being full. Messages are written by S and are read by the transmitter T. The initial state is 0. TB has one transition, *store*. The transition *store* means that a new message is stored in the buffer. If the buffer is originally full, its content is overwritten and it remains in the full state.

Transmitter

T has only one state, 0, and two transitions t_0 and t_1. The transitions are to be interpreted as follows. Transition t_i means that the message in the transmitter buffer has been read, assigned a sequence number (s.n.) i, and transmitted. (There are only two s.n.'s, 0 and 1, the s.n. assigned to new messages alternates between 0 and 1, hence the protocol name.) Repeated transitions $t_0t_0t_0\cdots$ or $t_1t_1t_1\cdots$ means that the *same* message is retransmitted; otherwise it is a new message. Thus, for example, the sequence $t_0t_0t_1t_0t_0t_0$ means that the first message is transmitted twice, the second message once, and the third message three times.

Channel

C is a buffer of size n in which messages sent by T are stored and later forwarded to RB in the same order, i.e., C is a FIFO queue. The state of C is

represented by an n-tuple

$$b = (b_1, \cdots, b_n) \in \{0, 1, e\}^n,$$

where $b_i = 0$, 1 or e, accordingly as buffer i contains a message with s.n. 0, 1 or is empty. Messages are forwarded to RB from buffer b_1 and messages from T are stored in the empty buffer with least number. Denote this least number by

$$l(b) := \begin{cases} \min \{i \mid i \le n,\ b_i = e\} \\ n+1, \text{ if } \forall\, i,\ b_i \neq e. \end{cases}$$

Thus the accessible states of C form the subset $B \subset \{0, 1, e\}^n$ consisting of n-tuples of the form

$$b = (b_1, \cdots, b_{l(b)-1}, e, \cdots, e).$$

The initial state is $(e, \cdots, e)$. C has six transitions: $t_0, t_1, r_0, r_1, l_0, l_1$; t_i represents messages input to C, whereas r_i, l_i represent messages output by C.

$$t_i(b) := \begin{cases} b \text{ if } l(b) = n+1 \text{ (reject message if all buffers full)} \\ (b_1, \cdots, b_{l(b)-1}, i, e \cdots, e) \text{ (store message with s.n. } i) \end{cases}$$

$$r_i(b_1, \cdots, b_n) := \begin{cases} (b_2, \cdots, b_n, e), \text{ if } b_1 = i \text{ (output message with s.n. } i) \\ \text{undefined, if } b_1 \neq i \end{cases}$$

$$l_i(b) := r_i(b), \ \forall\ b.$$

The difference between r_i and l_i will be seen in the receiver buffer model RB; r_i represents channel outputs that are successfully received by RB, whereas l_i are channel outputs that are "lost" and do not reach RB.

Receiver buffer

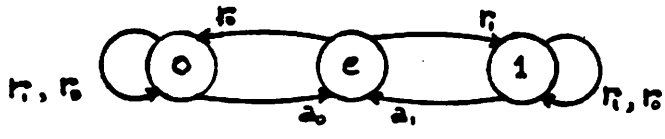
RB consists of a single buffer whose state d_2 takes values 0, 1 or e. As we will see, $d_2 = 0$ or 1 accordingly as the most recent message received had s.n. 0 or 1; whereas $d_2 = e$ if the most recent message received was acknowledged. Thus RB has four transitions, r_0, r_1, a_0, a_1.

$$r_0(d_2) := \begin{cases} 0, \text{ if } d_2 = e \text{ (receive message with s.n. 0)} \\ 1, \text{ if } d_2 = 1 \\ \text{undefined, otherwise} \end{cases}$$

$$r_1(d_2) := \begin{cases} 1, \text{ if } d_2 = e \text{ (receive message with s.n. 0)} \\ 0, \text{ if } d_2 = 0 \\ \text{undefined, otherwise} \end{cases}$$

$$a_0(d_2) := \begin{cases} e, \text{ if } d_2 = 0 \text{ (acknowledge message with s.n. 0, clear buffer)} \\ \text{undefined, otherwise} \end{cases}$$

$$a_1(d_2) := \begin{cases} e, & \text{if } d_2 = 1 \text{ (acknowledge message with s.n. 1, clear buffer)} \\ \text{undefined}, & \text{otherwise} \end{cases}$$



Observe that *RB* can receive several messages (i.e., several r_i transitions can occur) before the receiver buffer sends an acknowledgement. According to our model, only the last message will be acknowledged.

Receiver

R has only one state, 0, and one transition *read*. This transition means that the content of the receiver buffer is read.

The generator

The overall communication system is constructed by interconnecting the six subsystem generators *S*, *TB*, *T*, *C*, *RB* and *R* in the way described in the previous section.

The *send* transition in *S* and the *store* transition in *TB* are coupled in the sense that they must occur simultaneously. Thus both the *send* and *store* transitions will be labelled *s*.

This defines the generator $G = S//TB//T//C//RB//R$. Let

$$G = (Q, \Sigma, f, q_0, Q),$$

where

$$Q = \{(0, d_1, 0, b, d_2, 0) \mid d_1 \in \{0,1\}, b \in B, d_2 \in \{0, 1, e\}\},$$

$$\Sigma = \{s, t_i, r_i, l_i, a_i, read \mid i = 0, 1\},$$

and

$$q_0 = (0, 0, 0, (e, \cdots, e), e, 0).$$

The transition function f is given implicitly by the individual transition functions of *S*, *TB*, *T*, *C*, *RB* and *R* and by the defined connection.

Suppose the controlled transitions are $\Sigma_c = \{s, t_0, t_1, a_0, a_1, read\}$. Also assume that the a_i's are observed at the transmitter end instantaneously. This defines a CDEP G_c.

Notice that our assumption about the a_i's could be removed if we add to our model a reverse channel, similar in behavior to the forward channel, for the a_i's. But this assumption reduces the complexity of the overall generator G without any loss of generality.

The protocol

Informally, an AB protocol is a controller $\underline{S} = (S, \Phi)$ such that $\underline{S} / G_c$ has the following two-state behavior.

State 0: T sends a new message with s.n. 0 and repeats it until this message is acknowledged, i.e., a_0 is received. It then switches to state 1.

State 1: T sends a new message with s.n. 1 and repeats it until this message is acknowledged, i.e., a_1 is received. It then switches to state 0.

Also, duplicate messages should not be read by the receiver.

Formally, for $s = \sigma_1 \cdots \sigma_n \in L(G)$ define

$$\lambda(0) := 0$$

$$\mu(i) := \min \{m > \lambda(i-1) \mid \sigma_m = a_0\}$$

$$\lambda(i) := \min \{m > \mu(i) \mid \sigma_m = a_1\}$$

We also define $M_{SR}(s)$ to be the projection of s over $\{s, read\}$.

Say that $s \in L(G)$ is *valid* if

$$\{\lambda(i-1) < m < \mu(i)\} \wedge \{\sigma_m = t_0 \text{ or } t_1\} \Rightarrow \sigma_m = t_0, \tag{6.1a}$$

$$\{\mu(i) < m < \lambda(i)\} \wedge \{\sigma_m = t_0 \text{ or } t_1\} \Rightarrow \sigma_m = t_1. \tag{6.1b}$$

$$M_{SR}(s) \in \{(s.read)^*, s.(read.s)^*\}. \tag{6.1c}$$

Let K be the set of valid strings.

To implement the AB protocol, we are going to use two supervisors, one at the sender end and a second one at the receiver end. Each supervisor will control the transitions in its physical location. Therefore we subdivide Σ_c into $\Sigma_{1,c}$ and $\Sigma_{2,c}$, where $\Sigma_{1,c} = \{s, t_0, t_1\}$ and $\Sigma_{2,c} = \{a_0, a_1, read\}$. For the task of controlling $\Sigma_{1,c}$, the first supervisor $\underline{S}_1$ needs to observe $\{a_0, a_1, s\}$ while to control $\Sigma_{2,c}$, $\underline{S}_2$ needs to observe $\{r_0, r_1, read\}$. Hence we define the two masks M_1, M_2 such that $M_1 : \Sigma \to \{a_0, a_1, s, \varepsilon\}$ where $M_1(a_i) = a_i$, $M_1(s) = s$, and $M_1(\sigma) = \varepsilon$ otherwise; and $M_2 : \Sigma \to \{r_0, r_1, read, \varepsilon\}$ where $M_2(r_i) = r_i$, $M_2(read) = read$,

and $M_2(\sigma) = \varepsilon$ otherwise. From (6.1) it is easy to verify the next result.

Proposition 6.1

K is closed, $(\Sigma_u, L(G))$-invariant, and $(\{M_i\}, \{\Sigma_{i,c}\}, L(G))$-controllable.

It is easy to see that $L(\underline{S}/G_c)$, where $\underline{S} = \{\underline{S}_i, i = 1, 2\}$ is defined below. Supervisor $\underline{S}_1$ observes G through M_1 and controls $\Sigma_{1,c}$. It has a five state automaton S_1 shown in the sketch below and the feedback rule

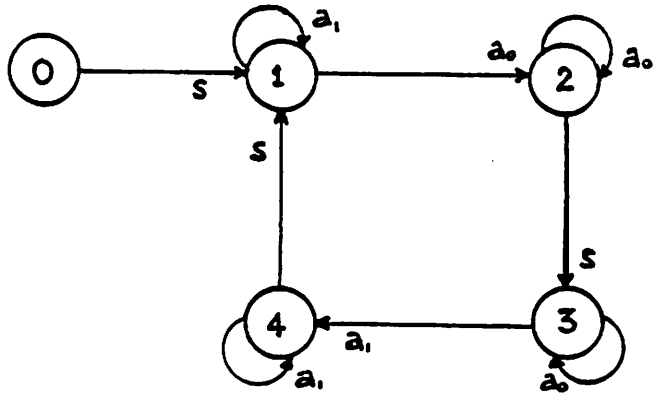
$$\Phi_1(0)(\sigma) = 1 \text{ iff } \sigma \in \{s\} \cup (\Sigma - \Sigma_{1,c})$$

$$\Phi_1(1)(\sigma) = 1 \text{ iff } \sigma \in \{t_0\} \cup (\Sigma - \Sigma_{1,c})$$

$$\Phi_1(2)(\sigma) = 1 \text{ iff } \sigma \in \{s\} \cup (\Sigma - \Sigma_{1,c})$$

$$\Phi_1(3)(\sigma) = 1 \text{ iff } \sigma \in \{t_1\} \cup (\Sigma - \Sigma_{1,c})$$

$$\Phi_1(4)(\sigma) = 1 \text{ iff } \sigma \in \{s\} \cup (\Sigma - \Sigma_{1,c}).$$



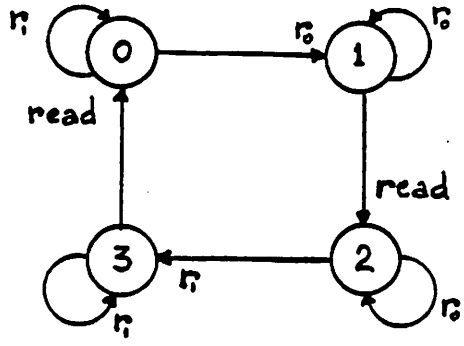
Supervisor $\underline{S}_2$ observes G through M_2 and controls $\Sigma_{2,c}$. It has the four state automaton S_2 shown in the sketch below and the feedback rule

$$\Phi_2(0)(\sigma) = 1 \text{ iff } \sigma \in \{a_1\} \cup (\Sigma - \Sigma_{2,c})$$

$$\Phi_2(1)(\sigma) = 1 \text{ iff } \sigma \in \{read\} \cup (\Sigma - \Sigma_{2,c})$$

$$\Phi_2(2)(\sigma) = 1 \text{ iff } \sigma \in \{a_0\} \cup (\Sigma - \Sigma_{2,c})$$

$$\Phi_2(3)(\sigma) = 1 \text{ iff } \sigma \in \{read\} \cup (\Sigma - \Sigma_{2,c}).$$



If $\underline{S}_1$ can also observe the transmission events t_0 and t_1, even without distinguishing between them, then it can enforce a minimum separation between retransmissions of the same message equal to a "timeout" of some counter. In this case the mask M_1 would map $\Sigma \rightarrow \{ a_0, a_1, t, s, \varepsilon\}$ where $M_1(t_i) = t$, and $M_1(\sigma)$ unchanged otherwise. The corresponding machine and the counter are shown in the sketch below. The counter has two states 0 and 1. In state 0, it is idle, while in state 1, it is counting. A transition that takes the counter from state 1 to state 0 resets the counter to its initial count value. The counter is connected to $\underline{S}_1$ in the way defined previously, so that the a_i, t and *Tout* transitions occur simultaneously in both machines.

The feedback rule is given by

$$\Phi_1(0)(\sigma) = 1 \text{ iff } \sigma \in \{s\} \cup (\Sigma - \Sigma_{1,c})$$

$$\Phi_1(1)(\sigma) = 1 \text{ iff } \sigma \in \{t_0\} \cup (\Sigma - \Sigma_{1,c})$$

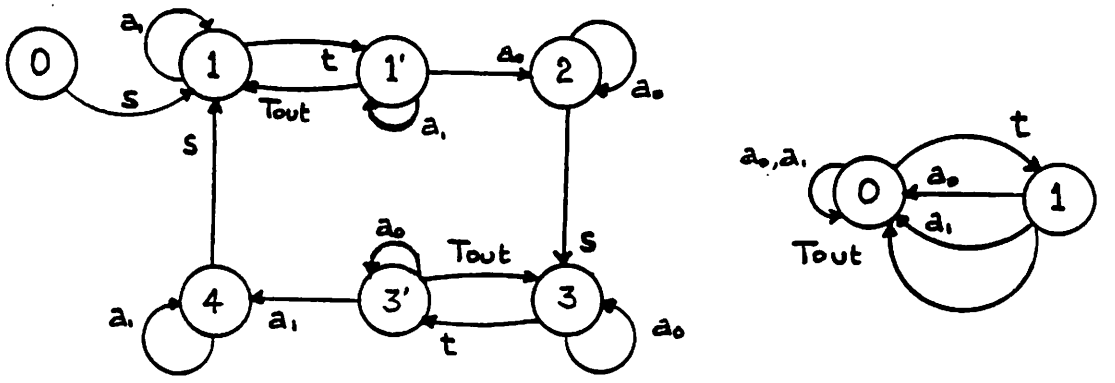
$$\Phi_1(1')(\sigma) = 1 \text{ iff } \sigma \in (\Sigma - \Sigma_{1,c})$$

$$\Phi_1(2)(\sigma) = 1 \text{ iff } \sigma \in \{s\} \cup (\Sigma - \Sigma_{1,c})$$

$$\Phi_1(3)(\sigma) = 1 \text{ iff } \sigma \in \{t_1\} \cup (\Sigma - \Sigma_{1,c})$$

$$\Phi_1(3')(\sigma) = 1 \text{ iff } \sigma \in (\Sigma - \Sigma_{1,c})$$

$$\Phi_1(4)(\sigma) = 1 \text{ iff } \sigma \in \{s\} \cup (\Sigma - \Sigma_{1,c}).$$



Thus transmissions t_0 and t_1 are enabled only in states 1 and 4 respectively, so that a timeout (*Tout*) or a valid acknowledgement must occur between successive transmissions.

7. Extensions and problems

While the framework presented here has the advantage relative to other approaches of posing explicitly the problem of control, these other approaches have been developed more extensively in terms of analysis and also, especially, in terms of simulation software. It would be quite valuable then to extend those approaches along the directions presented here in order to formulate the control problem.

Second, there is a need to relate the Ramadge-Wonham model to *performance analysis* that gives a numerical performance measure. One can readily imagine doing this by introducing a Markovian structure that assigns transition probabilities to events. However, a more important and difficult issue concerns the treatment of real time. This is necessary since most performance measures involve time in the form of "througput" or "delay".

Finally, there is virtually no work that explores the *implementation* aspect of the Ramadge-Wonham model in the sense mentioned in the introduction, both theoretically and empirically.

References

[1] Ramadge, P.J. and Wonham, W.M., "Supervisory control of a class of discrete event processes," Systems Control Group Reports #8311, October 1983 and #8515, November 1985, University of Toronto. To appear in *SIAM J. Contr. Optim.*, 1986.

[2] Ramadge, P.J. and Wonham, W.M., "On the supremal controllable sublanguage of a given language," Systems Control Group Reports #8312, November 1983 (revised November 1984), University of Toronto. To appear in *SIAM J. Contr. Optim.*, 1986.

[3] Ramadge, P.J. and Wonham, W.M., "Modular feedback logic for discrete event systems," Technical Report 48, Information Sciences and Systems Laboratory, Princeton University, July 1985.

[4] Ramadge, P.J. and Wonham, W.M., "Modular supervisory control of discrete event systems," INRIA Conference, June 1986.

[5] Wonham, W.M., "On control of discrete event systems," Systems Control Group Report #8508, July 1983, University of Toronto.

[6] Hoare, C.A.R., *Communicating Sequential Processes*, Prentice-Hall International, U.K., Ltd., 1985.

[7] Merlin, P. and Bochmann, G.V., "On the construction of submodule specifications and communication protocols," *ACM Trans. Prog. Lang. and Syst.* Vol 5(1), Jan. 1983, 1-25.