SIAM J. CONTROL AND OPTIMIZATION
Vol. 25, No. 1, January 1987

013

SUPERVISORY CONTROL OF A CLASS OF DISCRETE EVENT PROCESSES*

P. J. RAMADGE†‡ AND W. M. WONHAM†

Abstract. The paper studies the control of a class of discrete event processes, i.e., processes that are discrete, asynchronous and possibly nondeterministic. The controlled process is described as the generator of a formal language, while the controller, or *supervisor*, is constructed from a recognizer for a specified target language that incorporates the desired closed-loop system behavior. The existence problem for a supervisor is reduced to finding the largest *controllable* language contained in a given *legal* language. Two examples are provided.

Key words. discrete event systems, control, automata

AMS(MOS) subject classifications. 93C10, 93B50, 93C30

1. Introduction. In this paper we study the control of a class of systems broadly known as discrete event processes. The principal features of such processes are that they are discrete, asynchronous and (possibly) nondeterministic. Typical instances include computer networks, flexible manufacturing systems, and the start-up and shut-down procedures of industrial plants.

While numerous practical examples are described in the literature on simulation (see especially Fishman [1978] and Zeigler [1984]), there is at the present time apparently no unifying theory for the control of discrete event processes. Nor is it entirely clear what such a theory ought to encompass. Numerous approaches to the modeling of discrete event processes have appeared in the literature. A general sampling of these could include boolean models (Aveyard [1974]); Petri nets (Peterson [1981]); formal languages (Beauquier and Nivat [1980], Park [1981]); temporal logic (Pnueli [1979], Hailpern and Owicki [1983]); and port automata and flow networks (Milne and Milner [1979], Steenstrup, Arbib and Manes [1981]). All of this work is concerned, in one way or another, with the problem of how to achieve or verify the orderly flow of events; and to this end how to bring together ideas from logic, language and automaton theory. However, while control problems are implicit in much of the work just cited, control-theoretic ideas as such have found little application there. The variety of approaches reflects the diversity of areas in which discrete event processes play an important role. It also indicates that to date no dominant paradigm has emerged upon which a theory of control might be based.

In this article we investigate a simple abstract model of a controlled discrete event process, our main objective being to determine qualitative structural features of the relevant basic control problems. Specifically we take the controlled process to be the generator of a formal language, and study how the recognizer of a specified (target) language may be employed as a controller. In this regard we found suggestive the work of Shaw [1978] and Shields [1979] on flow expressions and path expressions

* Received by the editors August 17, 1984, and in final revised form March 21, 1985.

† Systems Control Group, Department of Electrical Engineering, University of Toronto, Toronto, Ontario, Canada M5S 1A4. This research was partially supported by the Natural Sciences and Engineering Research Council of Canada under grant A-7399.

‡ Present address, Department of Electrical Engineering and Computer Science, Princeton University, Princeton, New Jersey 08544.

respectively; while C. A. R. Hoare has recently brought to our attention certain points of similarity with his linguistic approach to concurrent processes in Hoare [1983, Chap. 2]. Nevertheless our definition of "controllable language," and our main results. (Theorems 7.1 and 10.1) on the existence and structure of controllers are believed to be quite new. Our approach is similar in spirit to some qualitative theories of multivariable control synthesis that have emerged over the last decade in the context of standard dynamic systems (for example, Wonham [1979], Nijmeijer [1983]). The present article is based on Ramadge [1983], and is summarized in Ramadge and Wonham [1984], while earlier versions appeared as Ramadge and Wonham [1982a, b].

The paper is organized as follows. In § 2 we define the class of controlled processes and controllers (*supervisors*) of interest; and in § 3 we discuss various associated formal languages. Sections 4 and 5 develop criteria for the existence of a supervisor for which the corresponding closed-loop controlled system satisfies given linguistic requirements; the main new idea here is that of a *controllable language.* Section 6 introduces the notion of a supervisor that is *proper,* namely *nonblocking* and *nonrejecting.* In § 7 we pose two problems of supervisor synthesis: the Supervisory Marking Problem (SMP) and the Supervisory Control Problem (SCP). Each of these is then shown to be solvable in a *minimally restrictive,* or "optimal," fashion in the class of proper supervisors, the "optimality" depending on a semilattice property of the relevant classes of languages. Section 8 defines a *projection* (or simplification) of supervisors. The latter, combined with some notions of reduction of languages and recognizers in § 9, leads to our second main result in § 10, the *Quotient Structure Theorem.* According to this, every efficiently constructed supervisor is structurally equivalent to a quotient (i.e., high-level, or lumped, model) of a recognizer of the desired closed-loop generated language. We conclude in §§ 11 and 12 with two simple but practical illustrations.

2. Controlled discrete-event processes.

2.1. Generators. To establish notation we first recall various standard ideas from automaton and language theory (cf. Hopcroft and Ullman [1979]). We define a *generator* to be a 5-tuple

$$\mathcal{G} = (Q, \Sigma, \delta, q_0, Q_m)$$

where Q is the set of *states* q, Σ is the *alphabet* or set of output symbols σ, $\delta : \Sigma \times Q \to Q$ is the *transition function,* $q_0 \in Q$ is the *initial state* and $Q_m \subset Q$ is a subset of states to be called *marker states.*[1] We always assume that Σ, but not necessarily Q or Q_m, is finite. In general, δ is only a partial function (pfn), meaning that, for each fixed $q \in Q$, $\delta(\sigma, q)$ is defined only for some subset $\Sigma(q) \subset \Sigma$ that depends on q. Formally $\mathcal{G}$ is equivalent to a directed graph with node set Q and an edge $q \to q'$ labeled σ for each triple (σ, q, q') such that $q' = \delta(\sigma, q)$. Such an edge, or state transition, will be called an *event.*

We interpret $\mathcal{G}$ as a device that starts in q_0 and executes state transitions, i.e., generates a sequence of events, by following its graph. Events are considered to occur spontaneously (no auxiliary forcing mechanism is postulated), asynchronously (i.e., without reference to a clock) and instantaneously. An event is thought of as signaled (to an outside observer, say) by its *label* σ. $\mathcal{G}$ may be nondeterministic in the sense that more than one event may be available for selection at a given node of its graph; however, distinct events at a given node always carry distinct labels.

[1] The terms *generator* and *marker state* are nonstandard, but better suited to our interpretation than, for example, "automaton" and "final state." Our "generator" is a special case of Harrison's "transition system" (Harrison [1965]); it will play the role of "plant" in the sense of control theory.

Let Σ^* denote the set of all finite strings s of elements of Σ, including the empty string, 1.[2] In standard fashion we construct the extended transition function

$$\delta : \Sigma^* \times Q \to Q \quad \text{(pfn)}$$

according to

$$\delta(1, q) = q, \qquad q \in Q,$$

and

$$\delta(s\sigma, q) = \delta(\sigma, \delta(s, q))$$

whenever $q' = \delta(s, q)$ and $\delta(\sigma, q')$ are both defined. Any subset of Σ^* is a *language* over Σ. The strings of a language are often called *words.* The language *generated* by $\mathcal{G}$ is

$$L(\mathcal{G}) = \{w: w \in \Sigma^* \text{ and } \delta(w, q_0) \text{ is defined}\}.$$

The language *marked* by $\mathcal{G}$ is

$$L_m(\mathcal{G}) = \{w: w \in L(\mathcal{G}) \text{ and } \delta(w, q_0) \in Q_m\}.$$

We interpret $L(\mathcal{G})$ as the set of all possible finite sequences of events that can occur; while $L_m(\mathcal{G}) \subset L(\mathcal{G})$ is a distinguished subset of these sequences that may be "marked," or recorded, perhaps representing completed "tasks" (or sequences of tasks) carried out by the physical process that $\mathcal{G}$ is intended to model.[3]

To conclude this subsection we remark that it is usually convenient to eliminate states of $\mathcal{G}$ that can never be reached (or "accessed") from q_0. Namely let

$$Q_{ac} = \{q: \exists w \in \Sigma^*, \delta(w, q_0) = q\},$$

$$Q_{ac,m} = Q_{ac} \cap Q_m,$$

$$\delta_{ac} = \delta \,|\, (\Sigma \times Q_{ac}).$$

The *accessible component* of $\mathcal{G}$, denoted by $Ac(\mathcal{G})$, is then defined to be

$$Ac(\mathcal{G}) = (Q_{ac}, \Sigma, \delta_{ac}, q_0, Q_{ac,m}).$$

A generator $\mathcal{G}$ is *accessible* if $\mathcal{G} = Ac(\mathcal{G})$.

We say that $\mathcal{G}$ is *co-accessible* if every string in $L(\mathcal{G})$ can be completed to a string in $L_m(\mathcal{G})$, i.e.,

$$(\forall w) w \in L(\mathcal{G}) \Rightarrow (\exists s) s \in \Sigma^* \quad \text{and} \quad ws \in L_m(\mathcal{G}).$$

If $\mathcal{G}$ is both accessible and co-accessible it is said to be *trim* (Eilenberg [1974]). It is well known (cf. Eilenberg [1974, § III.5]) that to every language (i.e., subset of Σ^*) there corresponds a trim generator that is essentially unique.

2.2. Controlled discrete event processes. To a generator $\mathcal{G} = (Q, \Sigma, \delta, q_0, Q_m)$ we now adjoin a means of control. That is, $\mathcal{G}$ will play the role of the "plant" (object to be controlled) of standard control theory. For this let $\Sigma_c \subset \Sigma$ be a distinguished subset of the alphabet; we say that an event (σ, q, q') is a *controlled event* if $\sigma \in \Sigma_c$. Let

$$\Gamma = \{0, 1\}^{\Sigma_c}$$

[2] 1 plays the role of identity of string concatenation, i.e., $1s = s1 = s$.

[3] Here there is no implication that generating action halts after the completion of some marked sequence; marked states of $\mathcal{G}$ need not be "final" states.

be the set of all binary assignments to the elements of Σ_c. Each assignment $\gamma \in \Gamma$, i.e., each function

$$\gamma : \Sigma_c \to \{0, 1\},$$

is a *control pattern.* An event (with label) σ is said to be *enabled by* γ if $\gamma(\sigma) = 1$, or *disabled by* γ if $\gamma(\sigma) = 0$. It is convenient to extend each $\gamma \in \Gamma$ to a map $\gamma : \Sigma \to \{0, 1\}$ by defining $\gamma(\sigma) = 1$ for each $\sigma \in \Sigma - \Sigma_c$. If $\delta : \Sigma \times Q \to Q$ is the transition function of $\mathcal{G}$, we define an augmented transition function

$$\delta_c : \Gamma \times \Sigma \times Q \to Q \quad \text{(pfn)}$$

according to

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

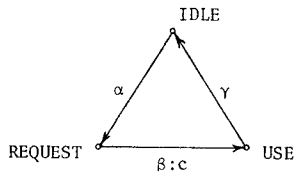
Formally, the object

$$\mathcal{G}_c = (Q, \Gamma \times \Sigma, \delta_c, q_0, Q_m)$$

is just another generator, constructed from $\mathcal{G}$ by a specification of Σ_c. However, we interpret $\mathcal{G}_c$ as a version of $\mathcal{G}$ that admits external control, as follows. For brevity call "an event labeled σ" simply "an event σ." For each fixed $\gamma \in \Gamma$ there is a generator $\mathcal{G}(\gamma)$ formed by deleting from the graph of $\mathcal{G}$ those events σ with $\gamma(\sigma) = 0$, i.e., those events that the control pattern γ disables. Then external control action would consist simply in switching the control pattern through a sequence of elements $\gamma, \gamma', \gamma'', \cdots$ in Γ, like switching the pattern of red and green lights in a traffic network. Observe that such control is "permissive" (cf. Peterson [1981]): while disabled events are certainly prevented from occurring, enabled events are not necessarily forced to occur.

A structure $\mathcal{G}_c$ as described above will be called a *controlled discrete event process* (CDEP).

2.3. Example—a primitive CDEP. A user of a resource may be modeled as a deterministic CDEP with three states I (IDLE), R (REQUEST) and U (USE), and with transitions as shown. Here we take (with some change of notation)



$\Sigma = \{\alpha, \beta, \gamma\}$ and $\Sigma_c = \{\beta\}$. The (two) control patterns correspond to evaluations $c = 0$ or $c = 1$ of the control variable c. A transition $R \to U$ may occur only when $c = 1$.

More interesting examples arise with the concurrent control of several CDEPs; these may then be combined into a single nondeterministic CDEP. At this stage the reader may skip ahead to §§ 11 and 12 for a glance at examples of this type.

2.4. Supervisors. Our objective will be to design a controller that switches control patterns in such a way that a given CDEP, $\mathcal{G}_c$, as described in § 2.2, behaves in obedience to various constraints. Such a controller will be called a *supervisor.* Formally a supervisor $\mathcal{S}$ is a pair

$$\mathcal{S} = (S, \phi).$$

Here

$$S = (X, \Sigma, \xi, x_0, X_m)$$

is a deterministic automaton with (possibly infinite) state set X, input alphabet Σ, transition (partial) function $\xi: \Sigma \times X \to X$, initial state x_0 and marker subset $X_m \subset X$; while

$$\phi : X \to \Gamma$$

is a (total) function that maps supervisor states x into control patterns γ. Thus for each $x \in X$,

$$\gamma := \phi(x) \in \{0, 1\}^{\Sigma_c}.$$

(As before we extend $\phi(x)$ to a map $\phi(x): \Sigma \to \{0, 1\}$ with $\phi(x)(\sigma) = 1$ for each $\sigma \in \Sigma - \Sigma_c$.) S will always be assumed to be accessible. We call ϕ the *state feedback map.*

In many applications it will be the case that $X_m = X$, i.e., the supervisor plays no auxiliary "marking" role; but the extra generality with $X_m \neq X$ is obtained with little effort.

We interpret S conventionally, as a device that executes a sequence of state transitions (according to ξ) in response to an appropriate input string $w \in \Sigma^*$. Thus we may couple $\mathcal{S}$ to $\mathcal{G}_c$ in a feedback loop by allowing the state transitions of S to be forced by $\mathcal{G}_c$, and requiring $\mathcal{G}_c$ to be constrained by the successive control patterns determined by the states of S. Formally define the partial function

$$\xi \times \delta_c : \Sigma \times X \times Q \to X \times Q \quad \text{(pfn)}$$

according to

$$(\sigma, x, q) \mapsto (\xi(\sigma, x), \delta_c(\phi(x), \sigma, q)).$$

Thus $(\xi \times \delta_c)(\sigma, x, q)$ is defined iff $\delta(\sigma, q)$ is defined, $\phi(x)(\sigma) = 1$, and $\xi(\sigma, x)$ is defined. This yields the generator

$$(X \times Q, \Sigma, \xi \times \delta_c, (x_0, q_0), X_m \times Q_m).$$

We define the *supervised discrete event process* (SDEP), denoted by $\mathcal{S}/\mathcal{G}_c$, to be the accessible generator[4]

$$\mathcal{S}/\mathcal{G}_c = Ac(X \times Q, \Sigma, \xi \times \delta_c, (x_0, q_0), X_m \times Q_m). \tag{2.1}$$

From now on we shall assume that $\xi \times \delta_c$ has been extended to strings of Σ^* in the way described in § 2.1 for δ. Of course, so far there is nothing to guarantee that $(X \times Q)_{ac}$ is anything more than the singleton $\{(x_0, q_0)\}$, or that $L(\mathcal{S}/\mathcal{G}_c)$ is any larger than the singleton $\{1\}$ consisting of the empty string alone.

In analogy to the case of $\mathcal{G}$ itself, we wish to interpret the language $L(\mathcal{S}/\mathcal{G}_c)$ generated by $\mathcal{S}/\mathcal{G}_c$ as the set of all possible finite sequences of events that can occur when $\mathcal{S}$ is coupled to $\mathcal{G}_c$ as just described. For this it is necessary to ensure that transitions of S are actually defined whenever they can occur in $\mathcal{G}$ and are enabled by ϕ. To formalize this relationship we shall say that $\mathcal{S}$ is *complete with respect to* $\mathcal{G}_c$ provided the following is true: for all $s \in \Sigma^*$, $\sigma \in \Sigma$ the three conditions

(i) $s \in L(\mathcal{S}/\mathcal{G}_c)$,
(ii) $s\sigma \in L(\mathcal{G})$ (i.e., $\delta(s\sigma, q_0)$ is defined),
(iii) $[\phi \circ \xi(s, x_0)](\sigma) = 1$ (i.e., σ is enabled at $\xi(s, x_0)$), together imply that
(iv) $s\sigma \in L(\mathcal{S}/\mathcal{G}_c)$ (i.e., $\xi(s\sigma, x_0)$ is defined).

While the definition (2.1) is logically acceptable as it stands, it will be of real value only when it is physically interesting, namely when $\mathcal{S}$ is complete with respect to $\mathcal{G}_c$.

[4] The notation is intended to suggest simply that "$\mathcal{G}_c$ is under supervision by $\mathcal{S}$;" no quotient structure is implied!

Before continuing with the general development, the reader might wish to glance at the opening paragraphs of §§ 11 and 12, where two concrete examples of supervisory control problems are provided.

3. Languages of $\mathcal{S}/\mathcal{G}_c$.

3.1. Definitions. Let $L \subset \Sigma^*$. The *closure* of L, denoted by $\bar{L}$, is the set of all strings that are prefixes of words of L, i.e.,

$$\bar{L} = \{s: s \in \Sigma^* \text{ and } (\exists t) t \in \Sigma^*, st \in L\}.$$

For instance, if $L = \emptyset$ then $\bar{L} = \emptyset$ and if $L \neq \emptyset$ then $1 \in \bar{L}$. A language L is *closed* if $L = \bar{L}$. If $\mathcal{G}$ is any generator then $L(\mathcal{G})$ is closed; if in addition $\mathcal{G}$ is trim then

$$L(\mathcal{G}) = \bar{L}_m(\mathcal{G}).$$

Let $\mathcal{G}_c$ be a CDEP constructed from a generator $\mathcal{G}$. For simplicity we shall denote $\mathcal{G}_c$ simply by its underlying generator $\mathcal{G}$. The notation $L(\mathcal{G})$ will henceforth denote the language generated by $\mathcal{G}$ if disabling control action were absent, i.e., all events $\sigma \in \Sigma_c$ were permanently enabled. Similarly we refer to $L_m(\mathcal{G})$ as the *uncontrolled (discrete-event) process language.* Let $\mathcal{S}$ be a supervisor for $\mathcal{G}$, $L(\mathcal{S}/\mathcal{G})$ the language generated by $\mathcal{S}/\mathcal{G}$ and $L_m(\mathcal{S}/\mathcal{G})$ the language marked by $\mathcal{S}/\mathcal{G}$. Define the *language controlled by $\mathcal{S}$ in $\mathcal{G}$* to be

$$L_c(\mathcal{S}/\mathcal{G}) := L(\mathcal{S}/\mathcal{G}) \cap L_m(\mathcal{G}). \tag{3.1}$$

In other words, $L_c(\mathcal{S}/\mathcal{G})$ consists of those (marked) strings of the uncontrolled process language that "survive" in the presence of supervision.

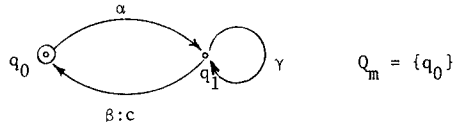
It is clear from the definitions that

$$L_m(\mathcal{S}/\mathcal{G}) \subset L_c(\mathcal{S}/\mathcal{G}) \subset L_m(\mathcal{G}) \tag{3.2}$$

and, if $\mathcal{G}$ is trim,

$$\bar{L}_m(\mathcal{S}/\mathcal{G}) \subset \bar{L}_c(\mathcal{S}/\mathcal{G}) \subset \bar{L}(\mathcal{S}/\mathcal{G})[= L(\mathcal{S}/\mathcal{G})] \subset L(\mathcal{G}) = \bar{L}_m(\mathcal{G}). \tag{3.3}$$

3.2. Examples. Let $\mathcal{G}$ be the generator over $\Sigma = \{\alpha, \beta, \gamma\}$ displayed below.

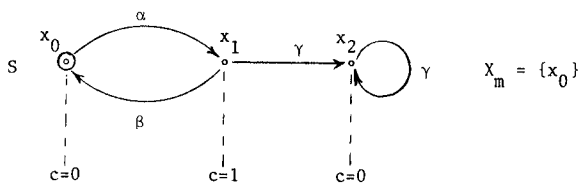


Then[5]

$$L_m(\mathcal{G}) = (\alpha\gamma^*\beta)^*.$$

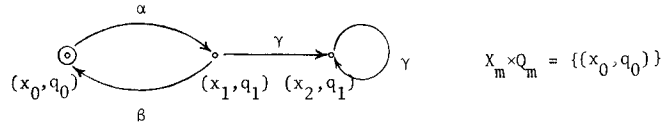
We shall consider two different supervisors, each specified by its transition graph, as follows.

(i)



[5] For the notation of regular expressions used here and below see, for example, Hopcroft and Ullman [1979].

This gives for $\mathcal{S}/\mathcal{G}$ the transition graph



It is seen that

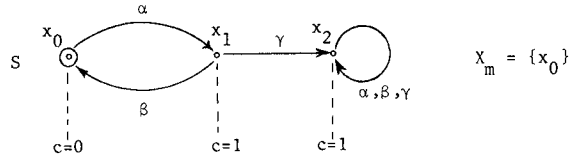
$$L(\mathcal{S}/\mathcal{G}) = (\alpha\beta)^*(1+\alpha\gamma^*),$$

$$L_c(\mathcal{S}/\mathcal{G}) = L(\mathcal{S}/\mathcal{G}) \cap L_m(\mathcal{G}) = (\alpha\beta)^* = L_m(\mathcal{S}/\mathcal{G}).$$

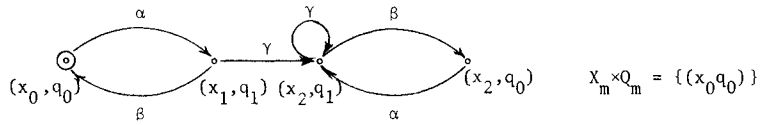
For future reference (§ 6) we note, however, that

$$\bar{L}_c(\mathcal{S}/\mathcal{G}) = (\alpha\beta)^*(1+\alpha) \subsetneq L(\mathcal{S}/\mathcal{G}).$$

(ii)



This gives for $\mathcal{S}/\mathcal{G}$ the transition graph



It is seen that

$$L_c(\mathcal{S}/\mathcal{G}) = L_m(\mathcal{G}) = (\alpha\gamma^*\beta)^*.$$

Again for future reference (§ 6), we note that

$$L_m(\mathcal{S}/\mathcal{G}) = (\alpha\beta)^* \subsetneq L_c(\mathcal{S}/\mathcal{G})$$

and

$$\bar{L}_m(\mathcal{S}/\mathcal{G}) \subsetneq \bar{L}_c(\mathcal{S}/\mathcal{G}).$$

4. Marking and control. A supervisor $\mathcal{S}$ performs two essentially independent tasks: marking (as described by $L_m(\mathcal{S}/\mathcal{G})$) and control (as described by $L_c(\mathcal{S}/\mathcal{G})$, $L(\mathcal{S}/\mathcal{G})$). If a given controlled behavior is achievable, then any marking task is simultaneously achievable that is consistent with the controlled behavior.

Without essential loss of generality we assume that the generator $\mathcal{G}$ is trim, namely

$$L(\mathcal{G}) = \bar{L}_m(\mathcal{G}).$$

PROPOSITION 4.1. (i) *For each sublanguage $K \subset L_m(\mathcal{G})$ there exists a complete supervisor $\mathcal{S}$ such that, for $\mathcal{S}/\mathcal{G}$, we have*

$$L(\mathcal{S}/\mathcal{G}) = L(\mathcal{G}), \qquad L_m(\mathcal{S}/\mathcal{G}) = K.$$

(ii) *Let L be a closed sublanguage of $L(\mathcal{G})$. If there exists a complete supervisor $\mathcal{S}$ for which $L(\mathcal{S}/\mathcal{G}) = L$ then for every sublanguage K of $L \cap L_m(\mathcal{G})$ there exists a complete supervisor $\mathcal{S}_K$ such that, correspondingly,*

$$L(\mathcal{S}_K/\mathcal{G}) = L, \qquad L_m(\mathcal{S}_K/\mathcal{G}) = K.$$

Before proving Proposition 4.1 we make the (more or less standard) definition: if $L \subset \Sigma^*$, a *recognizer* $\mathcal{M}$ for L is an accessible generator $\mathcal{G}$ such that $L_m(\mathcal{G}) = L$. While

a recognizer and an accessible generator are formally no different, we interpret a recognizer $\mathcal{M} = (Q, \Sigma, \delta, q_0, Q_m)$ as a device which, like a supervisor, is forced externally by strings in Σ^*; its action is thus to "recognize" precisely the words of L, regarded as input strings to $\mathcal{M}$.

Proof of Proposition 4.1. (i) Let $S = (X, \Sigma, \xi, x_0, X_m)$ be a recognizer for K. By adjoining a "dump" state to X, if necessary, we can arrange that $\xi(\sigma, x)$ is defined for all $(\sigma, x) \in \Sigma \times X$. Define

$$\phi : X \to \{0, 1\}^{\Sigma}$$

according to

$$\phi(x)(\sigma) = 1, \qquad x \in X, \quad \sigma \in \Sigma.$$

It is clear that $\mathcal{S} = (S, \phi)$ has the required properties.

(ii) Let

$$T = (Y, \Sigma, \eta, y_0, Y_m)$$

be a recognizer for K. By adjoining a "dump" state to Y, if necessary, it can be arranged that $\eta(\sigma, y)$ is defined for all $(\sigma, y) \in \Sigma \times Y$. Let

$$\mathcal{S} = (S, \phi), \qquad S = (X, \Sigma, \xi, x_0, X)$$

be a complete supervisor for which

$$L(\mathcal{S}/\mathcal{G}) = L.$$

Define the supervisor

$$\mathcal{S}_K = (S', \phi'), \qquad S' = (X', \Sigma, \xi', x_0', X_m')$$

according to

$$X' = X \times Y, \qquad \xi'(\sigma, x, y) = (\xi(\sigma, x), \eta(\sigma, y)),$$
$$x_0' = (x_0, y_0), \qquad X_m' = X \times Y_m, \qquad \phi'(x, y) = \phi(x).$$

Since the control action of $\mathcal{S}_K$ is the same as that of $\mathcal{S}$, it is clear that $L(\mathcal{S}_K/\mathcal{G}) = L$, and obviously $L_m(\mathcal{S}_K/\mathcal{G}) = K$. Also, $\xi'(\sigma, x, y)$ is defined just when $\xi(\sigma, x)$ is defined, so that S_k is complete with respect to $\mathcal{G}_c$. □

In this proof our construction merely installs a recognizer that acts as a marking device, either alone in part (i), or "in parallel" with the original supervisor $\mathcal{S}$ in part (ii). This does nothing to change the control action, but might be thought of as a means of recording when words in K have been completed.

5. Controllability. In this section we introduce a definition of controllability that will play a key role in characterizing those languages that can be generated by closed-loop structures $\mathcal{S}/\mathcal{G}$ with a given CDEP $\mathcal{G}$ and a suitable choice of complete supervisor $\mathcal{S}$.

Let $\mathcal{G} = (Q, \Sigma, \delta, q_0, Q_m)$ be a fixed CDEP. We assume that $\mathcal{G}$ is trim, i.e., $L(\mathcal{G}) = \bar{L}_m(\mathcal{G})$. Write $\Sigma_u = \Sigma - \Sigma_c$, i.e., Σ_u is the set of (labels of) events that cannot be disabled. Let $K \subset \Sigma^*$, $L \subset \Sigma^*$ be arbitrary languages. We say that K is

(i) *L-closed* if $K = \bar{K} \cap L$,

(ii) *(Σ_u, L)-invariant* if $\bar{K}\Sigma_u \cap L \subset \bar{K}$,

(iii) *controllable* if $K \subset L(\mathcal{G})$ and K is $(\Sigma_u, L(\mathcal{G}))$-invariant i.e., $\bar{K}\Sigma_u \cap L(\mathcal{G}) \subset \bar{K}$.

Recall that $\bar{K}$ is the language consisting of K together with all the prefixes (including the empty word) of words in K. Thus a sublanguage K of L is L-closed iff any prefix of K that is a word of L is also a word of K.

The language $\bar{K}\Sigma_u \cap L$ consists of all strings $s' = s\sigma$ where $s' \in L$, $s \in \bar{K}$ and $\sigma \in \Sigma_u$. If we think of L as representing "physically possible behavior," and $\bar{K}$ as "legally admissible behavior," then the string $s\sigma$ is a legally admissible string s followed by an uncontrolled symbol σ such that $s\sigma$ is physically possible. K is (Σ_u, L)-invariant precisely when all such strings are legally admissible, i.e., certain instances of uncontrolled behavior are nonetheless legal.

Finally, thinking of $L(\mathcal{G})$ as the uncontrolled process language, i.e., the physically possible uncontrolled behavior of our CDEP, we have that K is controllable if every prefix $s \in \bar{K}$ is physically possible, and every physically possible string $s\sigma$, with $s \in \bar{K}$ and σ uncontrolled, is again in $\bar{K}$.

The following technical proposition will support our main results (Theorems 6.1, 7.1) on existence of supervisors.

PROPOSITION 5.1. *Let $K_1 \subset L_m(\mathcal{G})$, $K_2 \subset L_m(\mathcal{G})$ and $K_3 \subset L(\mathcal{G})$ with $K_3 \neq \emptyset$. There exists a complete supervisor $\mathcal{S}$ such that for the closed-loop system $\mathcal{S}/\mathcal{G}$,*

$$L_m(\mathcal{S}/\mathcal{G}) = K_1, \qquad L_c(\mathcal{S}/\mathcal{G}) = K_2, \qquad L(\mathcal{S}/\mathcal{G}) = K_3, \tag{5.1}$$

iff

(i) $K_1 \subset K_2$,

(ii) $K_2 = K_3 \cap L_m(\mathcal{G})$,

(iii) K_3 *is closed and controllable.*

Proof. (*Only if*). Let the complete supervisor $\mathcal{S}$ satisfy (5.1). Condition (i) follows by (3.2) and (ii) by the definition (3.1) of $L_c(\mathcal{S}/\mathcal{G})$. We have already noted that $L(\mathcal{S}/\mathcal{G})(= K_3)$ is closed. To show that K_3 is controllable, suppose that $s\sigma \in L(\mathcal{G})$, with $s \in \bar{K}_3$ $(= K_3 = L(\mathcal{S}/\mathcal{G}))$ and $\sigma \in \Sigma_u$. If $\mathcal{S} = (S, \phi)$ with $S = (X, \Sigma, \xi, x_0, X_m)$ then, in the notation of § 2.4,

$$(x, q) := (\xi \times \delta_c)(s, x_0, q_0)$$

is defined. Since $\sigma \in \Sigma_u$ we have $\phi(x)(\sigma) = 1$, and as $s\sigma \in L(\mathcal{G})$ it follows that $q' := \delta(\sigma, q)$ is defined. Therefore

$$\delta_c(\phi(x), \sigma, q) = q';$$

and because $\mathcal{S}$ is complete with respect to $\mathcal{G}_c$,

$$x' := \xi(\sigma, x)$$

is defined. Therefore

$$(\xi \times \delta_c)(\sigma, x, q) = (\xi(\sigma, x), \delta_c(\phi(x), \sigma, q)) = (x', q')$$

is defined, namely

$$s\sigma \in L(\mathcal{S}/\mathcal{G}) = K_3 = \bar{K}_3,$$

so K_3 is controllable.

(*If*). By Proposition 4.1 it is enough to construct a complete supervisor $\mathcal{S}$ such that $L(\mathcal{S}/\mathcal{G}) = K_3$. For this let $S = (X, \Sigma, \xi, x_0, X)$ be a trim recognizer for K_3. Since K_3 is closed and S is trim, the marker set of S is X itself, as indicated; and we have that $\xi(s, x_0)$ is defined iff $s \in K_3$. For $x \in X$ let

$$\Sigma_x^0 = \{\sigma\colon (\exists s) s \in K_3 \text{ and } \xi(s, x_0) = x \text{ and } s\sigma \in L(\mathcal{G}) \text{ and } s\sigma \notin K_3\},$$

$$\Sigma_x^1 = \{\sigma\colon (\exists s) s \in K_3 \text{ and } \xi(s, x_0) = x \text{ and } s\sigma \in K_3\}.$$

We claim that $\Sigma_x^0 \cap \Sigma_x^1 = \varnothing$. In essence this follows by the fact that S is a trim recognizer for K_3. Indeed suppose that $\sigma \in \Sigma_x^0 \cap \Sigma_x^1$ with

$$s^0 \in K_3,\ \xi(s^0, x_0) = x, \qquad s^0\sigma \notin K_3,$$
$$s^1 \in K_3,\ \xi(s^1, x_0) = x, \qquad s^1\sigma \in K_3.$$

Then

$$\xi(\sigma, x) = \xi(\sigma, \xi(s^0, x_0)) = \xi(s^0\sigma, x_0)$$

fails to be defined (since $s^0\sigma \notin K_3$); whereas

$$\xi(\sigma, x) = \xi(\sigma, \xi(s^1, x_0)) = \xi(s^1\sigma, x_0)$$

must be defined (since $s^1\sigma \in K_3$): a contradiction.

By controllability of K_3, $\Sigma_x^0 \subset \Sigma_c$. Let

$$\phi : X \to \{0, 1\}^{\Sigma}$$

be any function such that, if $\phi(x) =: \gamma$, then

$$\gamma(\Sigma_x^0) = 0, \quad \gamma(\Sigma_x^1) = 1, \quad \gamma(\Sigma_u - \Sigma_x^1) = 1.$$

It is clear from the claim just proved that such a function ϕ exists.

Now let $\mathcal{S} = (S, \phi)$. It will be shown that $L(\mathcal{S}/\mathcal{G}) = K_3$. By the definition of S, it is clear that $L(\mathcal{S}/\mathcal{G}) \subset K_3$. For the reverse inclusion, we use induction on the length $|s|$ of strings $s \in L(\mathcal{G})$. If $|s| = 1$, i.e., $s = \sigma$ for some $\sigma \in \Sigma$, then

$$\sigma \in \Sigma_{x_0}^1 \quad \text{if } \sigma \in K_3$$

so $\sigma \in L(\mathcal{S}/\mathcal{G})$ if $\sigma \in K_3$. For a language $L \subset \Sigma^*$ write

$$L^{(j)} := \{s : s \in L \text{ and } |s| = j\}, \qquad j = 0, 1, 2, \cdots,$$

and note that $L = \bigcup_{j=0}^{\infty} L^{(j)}$. Assume for the induction step that

$$L^{(i)}(\mathcal{S}/\mathcal{G}) = K_3^{(i)}, \qquad i = 0, 1, \cdots, j.$$

Let $s \in L^{(j)}(\mathcal{S}/\mathcal{G})$ and consider the string $s\sigma \in L(\mathcal{G})$. Now $x := \xi(s, x_0)$ is defined, and so $\sigma \in \Sigma_x^0 \cup \Sigma_x^1$; therefore $s\sigma \in K_3$ implies

$$\sigma \in \Sigma_x^1 \text{ and } \xi(\sigma, x) \text{ is defined}$$

implies

$$\phi(x)(\sigma) = 1 \text{ and } \xi(\sigma, x) \text{ is defined}$$

implies

$$s\sigma \in L(\mathcal{S}/\mathcal{G}).$$

Therefore

$$L^{(j+1)}(\mathcal{S}/\mathcal{G}) = K_3^{(j+1)}.$$

It only remains to show that $\mathcal{S}$ is complete with respect to $\mathcal{G}$. For this let

$$s \in L(\mathcal{S}/\mathcal{G}) = K_3, \qquad s\sigma \in L(\mathcal{G}),$$

and

$$[\phi \circ \xi(s, x_0)](\sigma) = 1.$$

Now if $s\sigma \notin K_3$, then we must have

$$\sigma \in \Sigma^0_{\xi(s,x_0)}.$$

But this implies that

$$[\phi \circ \xi(s, x_0)](\sigma) = 0,$$

a contradiction. Hence $s\sigma \in K_3$ and $\mathcal{S}$ is complete. □

6. Proper supervisors. To specify controlled behavior in a way that is intuitively satisfying, more stringent conditions must be placed on the three languages

$$L_m(\mathcal{S}/\mathcal{G}), \quad L_c(\mathcal{S}/\mathcal{G}), \quad L(\mathcal{S}/\mathcal{G})$$

that describe the closed-loop system $\mathcal{S}/\mathcal{G}$. We shall say that $\mathcal{S}$ is *nonblocking* if

$$\bar{L}_c(\mathcal{S}/\mathcal{G})[= \overline{L(\mathcal{S}/\mathcal{G}) \cap L_m(\mathcal{G})}] = L(\mathcal{S}/\mathcal{G})$$

and that $\mathcal{S}$ is *nonrejecting* if

$$\bar{L}_c(\mathcal{S}/\mathcal{G}) = \bar{L}_m(\mathcal{S}/\mathcal{G}).$$

By definition we always have $\bar{L}_c(\mathcal{S}/\mathcal{G}) \subset L(\mathcal{S}/\mathcal{G})$. If $\mathcal{S}$ blocks, i.e., fails to be nonblocking, then there exists a string s generated by $\mathcal{S}/\mathcal{G}$ (i.e., $s \in L(\mathcal{S}/\mathcal{G})$) that can never be completed to a word $st \in L_c(\mathcal{S}/\mathcal{G})$, i.e., $s \notin \bar{L}_c(\mathcal{S}/\mathcal{G})$. In this sense the CDEP may be blocked from ever completing a "task." This undesirable situation is illustrated by supervisor (i) of § 3.2. Here, for instance, the string $\alpha\gamma \in L(\mathcal{S}/\mathcal{G}) - \bar{L}_c(\mathcal{S}/\mathcal{G})$.

If $\mathcal{S}$ rejects, i.e., fails to be nonrejecting, then there exists a string $s \in \bar{L}_c(\mathcal{S}/\mathcal{G})$ that can be completed to a "task" in $L_c(\mathcal{S}/\mathcal{G})$ but never to a task that is marked, i.e., (say) recorded. By contrast, if $\bar{L}_c(\mathcal{S}/\mathcal{G}) = \bar{L}_m(\mathcal{S}/\mathcal{G})$, so that

$$L_m(\mathcal{S}/\mathcal{G}) \subset L_c(\mathcal{S}/\mathcal{G}) \subset \bar{L}_m(\mathcal{S}/\mathcal{G}),$$

then for every $s \in L_c(\mathcal{S}/\mathcal{G})$ there is some t such that $st \in L_m(\mathcal{S}/\mathcal{G})$, and then $st \in L_c(\mathcal{S}/\mathcal{G})$ as well. In § 3.2 the supervisor (ii) rejects: only strings of the form $(\alpha\beta)^*$ are marked, while $L_c(\mathcal{S}/\mathcal{G}) = (\alpha\gamma^*\beta)^*$ represents the complete set of tasks that may be performed.

A supervisor $\mathcal{S}$ will be said to be *proper* if it is complete, nonblocking and nonrejecting; namely $\mathcal{S}$ is complete and

$$\bar{L}_m(\mathcal{S}/\mathcal{G}) = \bar{L}_c(\mathcal{S}/\mathcal{G}) = L(\mathcal{S}/\mathcal{G}).$$

THEOREM 6.1. *Let $K \subset L_m(\mathcal{G})$, $K \neq \emptyset$.*

(i) *There exists a proper supervisor $\mathcal{S}$ such that $L_m(\mathcal{S}/\mathcal{G}) = K$ iff K is controllable. In that case,*

$$L_c(\mathcal{S}/\mathcal{G}) = L_m(\mathcal{G}) \cap \bar{K}.$$

(ii) *There exists a proper supervisor $\mathcal{S}$ such that $L_c(\mathcal{S}/\mathcal{G}) = K$ iff K is controllable and $L_m(\mathcal{G})$-closed.*

Proof. (i) K is controllable iff $\bar{K}$ is closed and controllable, iff the triple

$$(K_1, K_2, K_3) := (K, \bar{K} \cap L_m(\mathcal{G}), \bar{K})$$

satisfies conditions (i)-(iii) of Proposition 5.1, iff there exists a complete supervisor $\mathcal{S}$ such that

$$(L_m(\mathcal{S}/\mathcal{G}), L_c(\mathcal{S}/\mathcal{G}), L(\mathcal{S}/\mathcal{G})) = (K, \bar{K} \cap L_m(\mathcal{G}), \bar{K})$$

and this condition means that $\mathcal{S}$ is proper.

(ii) K is controllable and $L_m(\mathcal{G})$-closed iff the triple

$$(K_1, K_2, K_3) := (K, K, \bar{K})$$

satisfies the conditions of Proposition 5.1, iff there exists a complete supervisor $\mathcal{S}$ such that

$$(L_m(\mathcal{S}/\mathcal{G}), L_c(\mathcal{S}/\mathcal{G}), L(\mathcal{S}/\mathcal{G})) = (K, K, \bar{K}),$$

and again this means that $\mathcal{S}$ is proper. □

7. Supervisor synthesis problems. Let languages $L_a, L_g \subset \Sigma^*$ be given, with

$$\varnothing \neq L_a \subset L_g \subset L_m(\mathcal{G}).$$

We interpret L_g as "legal behavior," i.e., each word of L_g is a "legal task;" and L_a as "minimal acceptable behavior," i.e., control of the CDEP $\mathcal{G}$ in such a way that a language smaller than L_a is generated is considered inadequate. We now introduce the

Supervisory Marking Problem (*SMP*). Construct a proper supervisor $\mathcal{S}$ for $\mathcal{G}$ such that

$$L_a \subset L_m(\mathcal{S}/\mathcal{G}) \subset L_g.$$

Similarly we define the

Supervisory Control Problem (*SCP*). Construct a proper supervisor $\mathcal{S}$ for $\mathcal{G}$ such that

$$L_a \subset L_c(\mathcal{S}/\mathcal{G}) \subset L_g.$$

If SCP is solvable then by the proof of Theorem 6.1(ii) we can always arrange that $L_m(\mathcal{S}/\mathcal{G}) = L_c(\mathcal{S}/\mathcal{G})$, so that automatically SMP is solvable as well. For a converse to this statement, consider the special but interesting case where L_g is $L_m(\mathcal{G})$-closed, i.e.,

$$L_g = \bar{L}_g \cap L_m(\mathcal{G}).$$

Then L_g is a sublanguage of $L_m(\mathcal{G})$ with the property that if a string $st \in L_g$ and $s \in L_m(\mathcal{G})$ then also $s \in L_g$. Now if SMP is solvable, the language $L_m(\mathcal{S}/\mathcal{G})$ satisfies

$$L_a \subset L_m(\mathcal{S}/\mathcal{G}) \subset L_g,$$

so that

$$L_a \subset L_m(\mathcal{S}/\mathcal{G}) \subset L_c(\mathcal{S}/\mathcal{G}).$$

Also, since $\mathcal{S}$ is proper,

$$\bar{L}_m(\mathcal{S}/\mathcal{G}) = \bar{L}_c(\mathcal{S}/\mathcal{G})$$

so that

$$L_c(\mathcal{S}/\mathcal{G}) \subset \bar{L}_c(\mathcal{S}/\mathcal{G}) = \bar{L}_m(\mathcal{S}/\mathcal{G}) \subset \bar{L}_g.$$

But $L_c(\mathcal{S}/\mathcal{G}) \subset L_m(\mathcal{G})$ by definition, i.e.,

$$L_c(\mathcal{S}/\mathcal{G}) \subset \bar{L}_g \cap L_m(\mathcal{G}) = L_g.$$

Hence

$$L_a \subset L_c(\mathcal{S}/\mathcal{G}) \subset L_g$$

and so SCP is solvable as well.

When SMP or SCP is solvable, it may be considered desirable that the solution be *minimally restrictive* in the sense that $L_m(\mathcal{S}/\mathcal{G})$ or $L_c(\mathcal{S}/\mathcal{G})$, considered as a sublanguage of $L_m(\mathcal{G})$, be as large as possible, subject to the constraint that it is a sublanguage of L_g. The fact that minimally restrictive solutions are possible in principle is due to a certain semilattice property that we now describe. For this, let $L \subset L(\mathcal{G})$ be an arbitrary sublanguage of $L(\mathcal{G})$. Let

$$\mathbf{C}_{\mathcal{G}}(L) := \{K: K \subset L \text{ and } K \text{ is controllable}\},$$

$$\mathbf{F}_{\mathcal{G}}(L) := \{K: K \subset L \text{ and } K = \bar{K} \cap L_m(\mathcal{G})\}.$$

Thus $\mathbf{C}_{\mathcal{G}}(L)$ (respectively, $\mathbf{F}_{\mathcal{G}}(L)$) are the controllable (respectively, $L_m(\mathcal{G})$-closed) sublanguages of L.

PROPOSITION 7.1. *$\mathbf{C}_{\mathcal{G}}(L)$ and $\mathbf{F}_{\mathcal{G}}(L)$ are nonempty classes of languages that are closed under arbitrary unions.*

Proof. Let $\varnothing$ be the empty language (i.e., the empty set in Σ^*). Clearly

$$\varnothing \in \mathbf{C}_{\mathcal{G}}(L) \quad \text{and} \quad \varnothing \in \mathbf{F}_{\mathcal{G}}(L)$$

so $\mathbf{C}_{\mathcal{G}}(L)$ and $\mathbf{F}_{\mathcal{G}}(L)$ are nonempty classes. If $K_\alpha \in \mathbf{C}_{\mathcal{G}}(L)$ for α in some index set A, then

$$\bar{K}_\alpha \Sigma_u \cap L(\mathcal{G}) \subset \bar{K}_\alpha, \qquad \alpha \in A.$$

By the definition of closure it follows immediately that

$$\overline{\bigcup_\alpha K_\alpha} = \bigcup_\alpha \bar{K}_\alpha.$$

Therefore

$$\left(\overline{\bigcup_\alpha K_\alpha}\right)\Sigma_u \cap L(\mathcal{G}) = \left(\bigcup_\alpha \bar{K}_\alpha\right)\Sigma_u \cap L(\mathcal{G}) = \bigcup_\alpha (\bar{K}_\alpha \Sigma_u) \cap L(\mathcal{G})$$

$$= \bigcup_\alpha [\bar{K}_\alpha \Sigma_u \cap L(\mathcal{G})] \subset \bigcup_\alpha \bar{K}_\alpha = \overline{\bigcup_\alpha K_\alpha},$$

and so

$$\bigcup_\alpha K_\alpha \in \mathbf{C}_{\mathcal{G}}(L)$$

as claimed. The proof for $\mathbf{F}_{\mathcal{G}}(L)$ is similar. □

By Proposition 7.1 each of $\mathbf{C}_{\mathcal{G}}(L)$, $\mathbf{F}_{\mathcal{G}}(L)$ contains a unique supremal element with respect to inclusion, which we denote by

$$\sup \mathbf{C}_{\mathcal{G}}(L), \qquad \sup \mathbf{F}_{\mathcal{G}}(L),$$

respectively. In fact, $\mathbf{C}_{\mathcal{G}}(L)$ and $\mathbf{F}_{\mathcal{G}}(L)$ are complete subsemilattices of the semilattice of all sublanguages of L, partially ordered by inclusion, and with join operation the union of languages.

On the basis of Theorem 6.1 and Proposition 7.1 we immediately obtain our first main result.

THEOREM 7.1. (i) *SMP is solvable iff*

$$\sup \mathbf{C}_{\mathcal{G}}(L_g) \supset L_a.$$

(ii) *SCP is solvable iff*

$$\sup \{\mathbf{C}_{\mathcal{G}}(L_g) \cap \mathbf{F}_{\mathcal{G}}(L_g)\} \supset L_a.$$

In each case the corresponding supervisor is minimally restrictive. □

8. Projections of supervisors. Let $\mathcal{S} = (S, \phi)$ and $\hat{\mathcal{S}} = (\hat{S}, \hat{\phi})$ each be supervisors for $\mathcal{G}$, where as usual

$$S = (X, \Sigma, \xi, x_0, X_m), \qquad \phi : X \to \{0,1\}^{\Sigma},$$
$$\hat{S} = (\hat{X}, \Sigma, \hat{\xi}, \hat{x}_0, \hat{X}_m), \qquad \hat{\phi} : \hat{X} \to \{0,1\}^{\Sigma}.$$

We shall say that a (total) function $\pi : X \to \hat{X}$ is a *projection* from $\mathcal{S}$ to $\hat{\mathcal{S}}$, and write $\pi : \mathcal{S} \to \hat{\mathcal{S}}$, provided

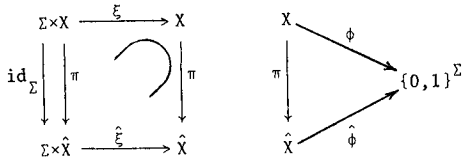
(i) $\pi : X \to \hat{X}$ is surjective,

(ii) $\pi(x_0) = \hat{x}_0$ and $X_m = \pi^{-1}(\hat{X}_m)$,

(iii)[6] $\hat{\xi} \circ (\mathrm{id}_\Sigma \times \pi)(\sigma, x) = \pi \circ \xi(\sigma, x)$ for all (σ, x) where $\xi(\sigma, x)$ is defined,

(iv) $\hat{\phi} \circ \pi = \phi$.

Under these conditions we shall refer to $\hat{S}$ as the *quotient* of S under π. The situation is displayed in the diagrams[7] below.



Projections represent very close relationships between supervisors, as expressed in the following. We assume that $\mathrm{id}_\Sigma \times \pi$ is extended to a map $\mathrm{id}_\Sigma \times \pi : \Sigma^* \times X \to \Sigma^* \times \hat{X}$ in the natural way.

PROPOSITION 8.1. *Let $\mathcal{S}$ be complete with respect to $\mathcal{G}$, and let $\pi : \mathcal{S} \to \hat{\mathcal{S}}$ be a projection. Then*

(i) *π is unique,*

(ii) $(L_m, L_c, L)(\mathcal{S}/\mathcal{G}) = (L_m, L_c, L)(\hat{\mathcal{S}}/\mathcal{G})$,

(iii) *$\hat{\mathcal{S}}$ is complete with respect to $\mathcal{G}$,*

(iv) *$\mathcal{S}$ is nonblocking (respectively, nonrejecting, proper) iff $\hat{\mathcal{S}}$ is nonblocking (respectively, nonrejecting, proper).*

Proof. As usual we write

$$\mathcal{S} = (S, \phi), \qquad S = (X, \Sigma, \xi, x_0, X_m)$$

and similarly for $\hat{\mathcal{S}}$. Recall that, by definition, both S and $\hat{S}$ are accessible.

(i) We have $\pi(x_0) = \hat{x}_0$. If $x \in X$ then, since $\mathcal{S}$ is accessible, there is a string $s \in \Sigma^*$ such that $x = \xi(s, x_0)$, so

$$\pi(x) = \pi \circ \xi(s, x_0) = \hat{\xi} \circ (\mathrm{id}_\Sigma \times \pi)(s, x_0) = \hat{\xi}(s, \hat{x}_0)$$

and this formula determines $\pi(x)$ uniquely.

(ii) Write

$$L = L(\mathcal{S}/\mathcal{G}), \qquad \hat{L} = L(\hat{\mathcal{S}}/\mathcal{G}).$$

If $s \in L$ with $|s| = 1$, i.e., $s = \sigma$, then $\phi(x_0)(\sigma) = 1$, and

$$(\xi \times \delta_c)(\sigma, x_0, q_0) = (\xi(\sigma, x_0), \delta_c(\phi(x_0), \sigma, q_0)) = (\xi(\sigma, x_0), \delta(\sigma, q_0))$$

---

[6] By definition $\mathrm{id}_\Sigma \times \pi : \Sigma \times X \to \Sigma \times \hat{X} : (\sigma, x) \mapsto (\sigma, \pi(x))$.

[7] The symbol $\supset$ means that the left-hand diagram is only "partially commutative," in the sense of (iii).

is defined; hence $\hat{\phi}(\hat{x}_0)(\sigma) = 1$ and

$$(\hat{\xi} \times \delta_c)(\sigma, \hat{x}_0, q_0) = (\hat{\xi}(\sigma, \hat{x}_0), \delta(\sigma, q_0)) = (\pi \circ \xi(\sigma, x_0), \delta(\sigma, q_0))$$
$$= (\pi \times \mathrm{id}_Q) \circ (\xi \times \delta_c)(\sigma, x_0, q_0)$$

is defined. This shows that

$$L^{(1)} \subset \hat{L}^{(1)}.$$

By induction on $|s|$ it is readily seen that

$$L^{(j)} \subset \hat{L}^{(j)}, \qquad j = 0, 1, \cdots,$$

hence $L \subset \hat{L}$.

For the reverse inclusion suppose first that $(\hat{\xi} \times \delta_c)(\sigma, \hat{x}_0, q_0)$ is defined. Then $\sigma \in \hat{L}$ and $\hat{\phi}(\hat{x}_0)(\sigma) = 1$. So

$$\phi(x_0)(\sigma) = (\hat{\phi} \circ \pi(x_0))(\sigma) = \hat{\phi}(\hat{x}_0)(\sigma) = 1.$$

Also

$$\delta(\sigma, q_0) = \delta_c(\hat{\phi}(\hat{x}_0), \sigma, q_0)$$

is defined, so by virtue of completeness $\xi(\sigma, x_0)$ is defined. Therefore

$$(\xi \times \delta_c)(\sigma, x_0, q_0) = (\xi(\sigma, x_0), \delta(\sigma, q_0))$$

is defined, and so $\hat{L}^{(1)} \subset L^{(1)}$. Assuming $\hat{L}^{(i)} \subset L^{(i)}$ $(i = 0, 1, \cdots, j)$, let $s \in \hat{L}^{(j)}$ and consider $s\sigma \in \hat{L}^{(j+1)}$. Then $s \in L^{(j)}$, so

$$x := (\xi \times \delta_c)(s, x_0, q_0), \qquad \hat{x} := (\hat{\xi} \times \delta_c)(s, \hat{x}_0, q_0)$$

are defined, and $\pi(x) = \hat{x}$. By exactly the same argument as before, applied to $(x, \hat{x})$ in place of $(x_0, \hat{x}_0)$, we conclude that $s\sigma \in L^{(j+1)}$. So $\hat{L} \subset L$ and $\hat{L} = L$. It is now immediate that

$$L_c(\mathscr{S}/\mathscr{G}) = L(\mathscr{S}/\mathscr{G}) \cap L_m(\mathscr{G}) = L(\hat{\mathscr{S}}/\mathscr{G}) \cap L_m(\mathscr{G}) = L_c(\hat{\mathscr{S}}/\mathscr{G}).$$

Finally,

$$s \in L_m(\mathscr{S}/\mathscr{G})$$
$$\text{iff } (\xi \times \delta_c)(s, x_0, q_0) \in X_m \times Q_m,$$
$$\text{iff } \xi(s, x_0) \in X_m \text{ and } s \in L_c(\mathscr{S}/\mathscr{G}),$$
$$\text{iff } \xi(s, x_0) \in \pi^{-1}(\hat{X}_m) \text{ and } s \in L_c(\mathscr{S}/\mathscr{G}),$$
$$\text{iff } \pi \circ \xi(s, x_0) \in \hat{X}_m \text{ and } s \in L_c(\hat{\mathscr{S}}/\mathscr{G}),$$
$$\text{iff } \hat{\xi}(s, \hat{x}_0) \in \hat{X}_m \text{ and } s \in L_c(\hat{\mathscr{S}}/\mathscr{G}),$$
$$\text{iff } (\hat{\xi} \times \delta_c)(s, \hat{x}_0, q_0) \in \hat{X}_m \times Q_m,$$
$$\text{iff } s \in L_m(\hat{\mathscr{S}}/\mathscr{G}).$$

(iii) To verify that $\hat{\mathscr{S}}$ is complete with respect to $\mathscr{G}$, let

$$s \in L(\hat{\mathscr{S}}/\mathscr{G}), \qquad s\sigma \in L(\mathscr{G})$$

and

$$(\hat{\phi} \circ \hat{\xi}(s, \hat{x}_0))(\sigma) = 1.$$

Then $s \in L(\mathscr{S}/\mathscr{G})$ by (ii), and

$$\phi \circ \xi(s, x_0)(\sigma) = (\hat{\phi} \circ \pi \circ \xi(s, x_0))(\sigma) = (\hat{\phi} \circ \hat{\xi}(s, \hat{x}_0))(\sigma) = 1.$$

Since $\mathcal{S}$ is complete it follows that $\xi(s\sigma, x_0)$ is defined, hence (because π is a projection) $\hat{\xi}(s\sigma, \hat{x}_0)$ is defined as well, namely $\hat{\mathcal{S}}$ is complete.

(iv) Immediate from (ii) and (iii). □

9. Efficient supervisor. In this section we give a simple abstract characterization of an "efficiently constructed" supervisor for a given nonempty, controllable and $L_m(\mathcal{G})$-closed language $K \subset \Sigma^*$. By Theorem 6.1(ii) we know that a proper supervisor $\mathcal{S} = (S, \phi)$ exists such that $K = L_m(\mathcal{S}/\mathcal{G}) = L_c(\mathcal{S}/\mathcal{G})$, so that

$$\bar{K} = L(\mathcal{S}/\mathcal{G}).$$

Furthermore, by the construction used in the proof of Proposition 5.1 ("if" statement), we can arrange that, for a string $s \in \bar{K}$, the state x reached by S is such that, for all $\sigma \in \Sigma_c$,

$$\phi(x)(\sigma) = \begin{cases} 0, & s\sigma \notin \bar{K}, \\ 1, & s\sigma \in \bar{K}. \end{cases} \tag{9.1}$$

On the basis of (9.1) we define an equivalence relation on Σ^* as follows. Strings $s, s' \in \Sigma^*$ are *control-equivalent*, written $s \sim s'$, if for all $\sigma \in \Sigma_c$, $s\sigma \in \bar{K}$ iff $s'\sigma \in \bar{K}$. Thus two strings are control-equivalent if the control action (9.1) immediately following either one is the same for every $\sigma \in \Sigma$.

Recall from automaton theory (for example, Harrison [1965]) that an equivalence relation $\mathbf{e}$ on Σ^* is a *right-congruence* if, whenever $s, s' \in \Sigma^*$ and $s \equiv s' \pmod{\mathbf{e}}$, then for all $t \in \Sigma^*$, $st \equiv s't \pmod{\mathbf{e}}$. Now let $\{\mathbf{e}_\alpha : \alpha \in A\}$ be an arbitrary nonempty family of equivalence relations on Σ^*. Their lattice-theoretic join, written

$$\mathbf{e} = \sup\{\mathbf{e}_\alpha : \alpha \in A\}, \tag{9.2}$$

is defined as follows (cf. Szász [1963]): $s \equiv s' \pmod{\mathbf{e}}$ if there exists an integer $k \geqq 1$, elements $\alpha_0, \cdots, \alpha_k \in A$, and strings $s_1, \cdots, s_k \in \Sigma^*$ such that

$$\begin{aligned} s &\equiv s_1 \pmod{\mathbf{e}_{\alpha_0}} \\ s_1 &\equiv s_2 \pmod{\mathbf{e}_{\alpha_1}} \\ &\vdots \\ s_{k-1} &\equiv s_k \pmod{\mathbf{e}_{\alpha_{k-1}}} \\ s_k &\equiv s' \pmod{\mathbf{e}_{\alpha_k}}. \end{aligned}$$

It is easy to check that if, in particular, the $\mathbf{e}_\alpha$ are right-congruences, then so is $\mathbf{e}$.

The lattice-theoretic ordering of equivalence relations on Σ^* is defined as follows: $\mathbf{e}_1 \leqq \mathbf{e}_2$ if, for all $s, s' \in \Sigma^*$, $s \equiv s' \pmod{\mathbf{e}_1}$ implies $s \equiv s' \pmod{\mathbf{e}_2}$; $\mathbf{e}_1$ is said to be *finer than* $\mathbf{e}_2$ (or $\mathbf{e}_2$ is *coarser than* $\mathbf{e}_1$). Then $\mathbf{e}$ in (9.2) is the finest equivalence relation on Σ^* that is coarser then each $\mathbf{e}_\alpha$, $\alpha \in A$.

In general, control-equivalence $\sim$ is not a right-congruence. However, if we define $s \equiv s' \pmod{\mathbf{o}}$ if $s = s'$, then trivially $\mathbf{o}$ is a right-congruence and $\mathbf{o} \leqq \sim$. It follows from the preceding that the equivalence

$$\approx := \sup\{\mathbf{e}: \mathbf{e} \text{ a right-congruence on } \Sigma^* \text{ and } \mathbf{e} \leqq \sim\}$$

exists, and is the coarsest right-congruence on Σ^* that is finer than $\sim$.

For $s \in \Sigma^*$ let $[s]$ be the equivalence class of s mod $\approx$. In standard fashion we construct the corresponding automaton, defined on strings in $\bar{K}$. Let

$$\bar{S} = (\bar{X}, \Sigma, \bar{\xi}, \bar{x}_0, \bar{X}).$$

Here

$$\bar{X} = \{[s]: s \in \bar{K}\}, \qquad \bar{x}_0 = [1]$$

(note that $1 \in \bar{K}$ as $\bar{K}$ is nonempty and closed); finally $\bar{\xi}(\sigma, \bar{x}) = \bar{x}'$ if $\bar{x} = [s]$, $s \in \bar{K}$, $s\sigma \in \bar{K}$ and $[s\sigma] = \bar{x}'$; otherwise $\bar{\xi}$ is not defined. Next we define a control law

$$\bar{\phi} : \bar{X} \to \{0, 1\}^{\Sigma}$$

according to

$$\bar{\phi}(\bar{x})(\sigma) = 0;$$

if $\sigma \in \Sigma_c$, there exists $s \in \bar{K}$ with $[s] = \bar{x}$ and $s\sigma \notin \bar{K}$; otherwise $\bar{\phi}(\bar{x})(\sigma) = 1$. By our construction of $\bar{X}$, $\bar{\xi}$ and $\bar{\phi}$ are unambiguously determined. We can now define the efficient supervisor

$$\bar{\mathscr{S}} = (\bar{S}, \bar{\phi}).$$

Evidently $\bar{\mathscr{S}}$ is "efficient" in the sense that any automaton $\hat{S}$, that supports the defined control action (9.1) on each string $s \in \bar{K}$, must have a state structure (right-congruence on Σ^*) at least as fine as that of $\bar{S}$.

It is easy to see that $\bar{\mathscr{S}}$ is complete, since

$$s \in \bar{K}, \quad [s] = \bar{x}, \quad \bar{\phi}(\bar{x})(\sigma) = 1, \quad s\sigma \in L(\mathscr{G}),$$

implies $s\sigma \in \bar{K}$ and therefore $\bar{\xi}(\sigma, \bar{x})$ is defined. Much as in the proof of Proposition 8.1 it is straightforward to verify that $L(\bar{\mathscr{S}}/\mathscr{G}) = \bar{K} = L(\mathscr{S}/\mathscr{G})$. Finally, as $\bar{X}_m = \bar{X}$,

$$L_m(\bar{\mathscr{S}}/\mathscr{G}) = L(\bar{\mathscr{S}}/\mathscr{G}) \cap L_m(\mathscr{G}) = \bar{K} \cap L_m(\mathscr{G}) = K$$

since K is $L_m(\mathscr{G})$-closed; so that

$$K = L_m(\bar{\mathscr{S}}/\mathscr{G}) = L_c(\bar{\mathscr{S}}/\mathscr{G})$$

and $\bar{\mathscr{S}}$ is proper. Thus $\bar{\mathscr{S}}$ performs the same control action on $\mathscr{G}$ as the supervisor $\mathscr{S}$ with which we started.

Let $\equiv (\mathrm{mod}\ \bar{K})$ denote $\bar{K}$-equivalence on Σ^*: $s \equiv s' \ (\mathrm{mod}\ \bar{K})$ if for all $t \in \Sigma^*$, $st \in \bar{K}$ iff $s't \in \bar{K}$. Clearly $\equiv (\mathrm{mod}\ \bar{K})$ is a right-congruence. Also, if $s \equiv s' \ (\mathrm{mod}\ \bar{K})$ then in particular for all $\sigma \in \Sigma_c$, $s\sigma \in \bar{K}$ iff $s'\sigma \in \bar{K}$. It follows that

$$\equiv (\mathrm{mod}\ \bar{K}) \leqq \approx;$$

i.e., for all $s, s' \in \Sigma^*$, $s \equiv s' \ (\mathrm{mod}\ \bar{K})$ implies $s \approx s'$. In particular, if $s, s' \in \bar{K}$ and $s \equiv s' \ (\mathrm{mod}\ \bar{K})$ then

$$\bar{\xi}(s, \bar{x}_0) = \bar{\xi}(s', \bar{x}_0).$$

We shall refer to the latter property by saying that the automaton $\bar{S}$ is $\bar{K}$-*reduced.* By its construction, $\bar{S}$ is also $\bar{K}$-*trim*, namely every state of $\bar{S}$ is visited by a word of $\bar{K}$; that is, for every $\bar{x} \in \bar{X}$ there is $s \in \bar{K}$ such that $\bar{\xi}(s, \bar{x}_0) = \bar{x}$. In the next section it is shown that any supervisor with these two properties can be projected from a supervisor based on a recognizer for $\bar{K}$.

10. Quotient structure theorem. We can now prove the second main result of this paper. It states, roughly, that "every efficiently constructed supervisor is a quotient (high-level, or lumped, model) of the desired closed-loop behavior."

Let $\mathscr{S} = (S, \phi)$ be a complete supervisor for $\mathscr{G}$. Write $K_1 := L_m(\mathscr{S}/\mathscr{G})$, $K_3 := L(\mathscr{S}/\mathscr{G})$ and assume that S is K_3-reduced and K_3-trim. These properties hold for the "efficient" supervisor of the previous section. Finally let

$$\hat{S}^0 = (X^0, \Sigma, \xi^0, x_0^0, X^0)$$

be a trim recognizer for K_3.

THEOREM 10.1. *Subject to the foregoing hypotheses, there exist a subset* $X_m^0 \subset X^0$ *and a state feedback map* $\phi^0 : X^0 \to \{0,1\}^\Sigma$ *with the following properties*:

(i) *The supervisor*

$$\mathcal{S}^0 := (S^0, \phi^0), \qquad S^0 := (X^0, \Sigma, \xi^0, x_0^0, X_m^0)$$

is a complete supervisor for $\mathcal{G}$ *with*

$$L_m(\mathcal{S}^0/\mathcal{G}) = K_1, \qquad L(\mathcal{S}^0/\mathcal{G}) = K_3.$$

(ii) *There is a projection* $\pi : \mathcal{S}^0 \to \mathcal{S}$.

(iii) *If* $\mathcal{S}$ *is proper then so is* $\mathcal{S}^0$.

Proof. Write

$$S = (X, \Sigma, \xi, x_0, X_m).$$

Let $x^0 \in X^0$. Since $\hat{S}^0$ is trim there exists $s \in K_3$ such that

$$\xi^0(s, x_0^0) = x^0.$$

Let $\xi(s, x_0) =: x \in X$ and define $\pi : X^0 \to X$ according to $\pi(x^0) = x$.

To show that π is well-defined, let $t \in K_3$, $\xi^0(t, x_0^0) = x^0$, and let $\xi(t, x_0) =: y \in X$. Since $\hat{S}^0$ is a recognizer for K_3 and $\xi^0(s, x_0^0) = \xi^0(t, x_0^0)$, we have $s \equiv t \pmod{K_3}$. Since S is K_3-reduced, $\xi(s, x_0) = \xi(t, x_0)$, i.e., $x = y$.

We claim that π is a projection. First let $x \in X$. Since S is K_3-trim, there exists $s \in K_3$ such that $\xi(s, x_0) = x$. Let $\xi^0(s, x_0^0) =: x^0$. Then as already shown, $\pi(x^0) = x$, so π is surjective. To verify that π respects ξ^0, let $\xi^0(\sigma, x^0) = y^0$. We have $x^0 = \xi^0(s, x_0^0)$ for some $s \in K_3$, and then $s\sigma \in K_3$. Since $K_3 = L(\mathcal{S}/\mathcal{G})$, $\xi(\sigma, \pi(x^0))$ is defined and, as shown already, coincides with $\pi \circ \xi^0(\sigma, x^0)$. To establish that π is a projection it only remains to define $X_m^0 = \pi^{-1}(X_m)$ together with $\phi^0 := \phi \circ \pi$.

It must be shown that $L(\mathcal{S}^0/\mathcal{G}) = K_3$. By the argument used in the proof of Proposition 5.1 it is enough to show that

(i) $(\forall \sigma, x^0)(\exists s)\xi^0(s, x_0^0) = x^0$ and $s\sigma \in L(\mathcal{G})$ and $s\sigma \notin K_3 \Rightarrow \phi^0(x^0)(\sigma) = 0$,

(ii) $(\forall \sigma, x^0)(\exists s)\xi^0(s, x_0^0) = x^0$ and $s\sigma \in K_3 \Rightarrow \phi^0(x^0)(\sigma) = 1$.

For (i), let $\xi^0(s, x_0^0) = x^0$, $s\sigma \in L(\mathcal{G})$, $s\sigma \notin K_3$. Clearly $s \in K_3$. If $\xi(s, x_0) = x$ then, as in the proof of Proposition 5.1, it follows necessarily that $\phi(x)(\sigma) = 0$ and therefore

$$\phi^0(x^0)(\sigma) = (\phi \circ \pi(x^0))(\sigma) = \phi(x)(\sigma) = 0.$$

The proof of (ii) is similar.

To show that $\mathcal{S}^0$ is complete with respect to $\mathcal{G}$ we note that

$$s \in L(\mathcal{S}^0/\mathcal{G}), \quad s\sigma \in L(\mathcal{G}) \quad \text{and} \quad [\phi^0 \circ \xi^0(s, x_0^0)](\sigma) = 1,$$

iff

$$s \in L(\mathcal{S}/\mathcal{G}), \quad s\sigma \in L(\mathcal{G}) \quad \text{and} \quad [\phi \circ \xi(s, x_0)](\sigma) = 1.$$

But since $\mathcal{S}$ is complete the latter condition implies that $s\sigma \in L(\mathcal{S}/\mathcal{G}) = L(\mathcal{S}^0/\mathcal{G})$.

Finally let $s \in K_3$. Then

$$\begin{aligned} s \in K_1 \\ &\text{iff } \xi(s, x_0) \in X_m \text{ and } \delta(s, q_0) \in Q_m, \\ &\text{iff } \pi \circ \xi^0(s, x_0^0) \in X_m \text{ and } \delta(s, q_0) \in Q_m, \\ &\text{iff } \xi^0(s, x_0^0) \in X_m^0 \text{ and } \delta(s, q_0) \in Q_m, \\ &\text{iff } s \in L_m(\mathcal{S}^0/\mathcal{G}). \end{aligned}$$

So $L_m(\mathcal{S}^0/\mathcal{G}) = L_m(\mathcal{S}/\mathcal{G})$. In particular if $\mathcal{S}$ is proper, so is $\mathcal{S}^0$. □

11. Example 1. We consider two users of a single resource, each modeled as in § 2.3, giving the state transition graphs $\mathcal{G}_1$, $\mathcal{G}_2$ of Fig. 11.1. For $\mathcal{G}$ we take the "shuffle" of $\mathcal{G}_1$, $\mathcal{G}_2$, namely the process determined by the concurrent actions of $\mathcal{G}_1$ and $\mathcal{G}_2$ under the assumption that these actions are asynchronous and independent. This assumption rules out the simultaneous occurrence of an event in $\mathcal{G}_1$ with an event in $\mathcal{G}_2$, but otherwise places no constraint on their joint behavior. The graph of $\mathcal{G}$ is thus as shown in Fig. 11.2. Here the state ⊙ is both q_0 and (as a singleton) Q_m, while $L_m(\mathcal{G})$ consists of all words over the alphabet

$$\Sigma = \{\alpha_1, \beta_1, \gamma_1, \alpha_2, \beta_2, \gamma_2\}$$

corresponding to paths in the graph that begin and end at ⊙.

The objective of supervisory control is to manipulate the binary controls c_1, c_2 in order to satisfy the following synchronization requirements.

(i) *Mutual exclusion*: $\mathcal{G}_1$, $\mathcal{G}_2$ never simultaneously occupy their respective USE states.

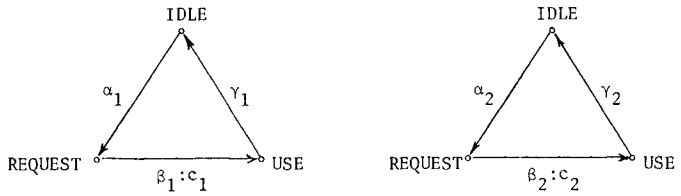


FIG. 11.1. *Example* 1: *Independent* CDEPs $\mathcal{G}_1$ *and* $\mathcal{G}_2$.

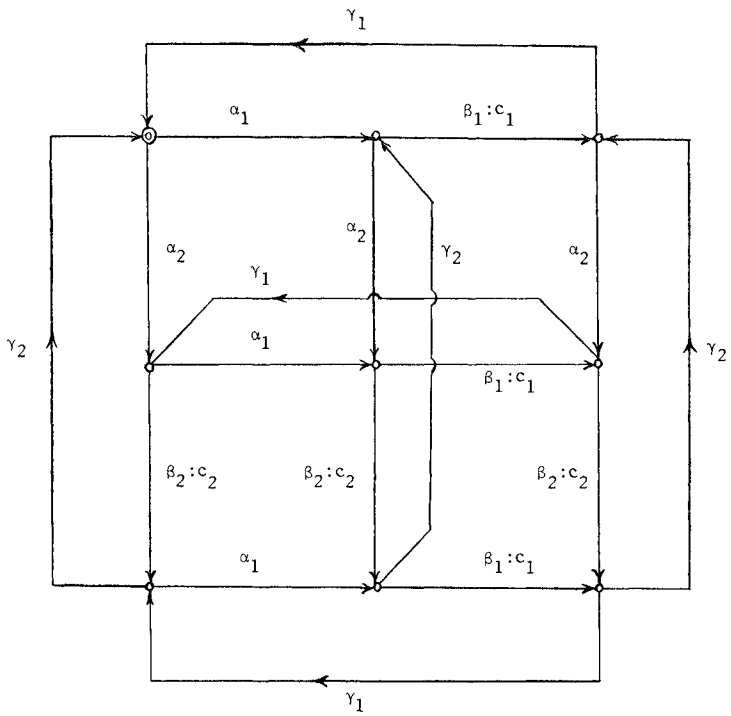


FIG. 11.2. *Example* 1: *Shuffle* $\mathcal{G}$ *of* $\mathcal{G}_1$ *and* $\mathcal{G}_2$.

(ii) *Fair usage*: The USE states of $\mathcal{G}_1$, $\mathcal{G}_2$ are occupied according to first-come-first-served discipline, namely the index sequence of events β_i must coincide with the index sequence of events α_j.

In practical terms this standard problem could, of course, be solved by a queue; but instead we shall approach it via the ideas of previous sections. However, we defer to a future article the theoretical issue of how conditions like (i) and (ii) may be formalized, simply taking it for granted that from them the "legal" behavior $L_g \subset L_m(\mathcal{G})$ can be explicitly determined. In fact the reader may convince himself that L_g is described by the generator displayed in Fig. 11.3.[8]

By inspection of Fig. 11.3 it is easy to see that L_g is both controllable and $L_m(\mathcal{G})$-closed. That is,

$$L_g = \sup\{\mathbf{C}_{\mathcal{G}}(L_g) \cap \mathbf{F}_{\mathcal{G}}(L_g)\}.$$

By Theorem 6.1(ii) there exists a proper supervisor $\mathcal{S} = (S, \phi)$ such that $L_c(\mathcal{S}/\mathcal{G}) = L_g$. As demonstrated in the proof of Proposition 5.1, the state transition diagram for L_g (Fig. 11.3) can serve to define S; it just remains to identify the state feedback map ϕ. For each state x of S, $\phi(x)$ is a map

$$\phi(x): \{c_1, c_2\} \to \{0, 1\},$$

i.e., a binary evaluation of each of the controls c_1, c_2. So, with reference to Fig. 11.3, it is enough to define

$$\phi(x)(c_1) = \begin{cases} 1 & \text{if an edge labeled } \beta_1 \text{ issues from } x, \\ 0 & \text{otherwise,} \end{cases}$$

and similarly for $\phi(x)(c_2)$. The resulting control patterns are tabulated in Fig. 11.4. The supervisor $\mathcal{S} = (S, \phi)$ then certainly determines

$$L(\mathcal{S}/\mathcal{G}) = \bar{L}_g, \qquad L_m(\mathcal{S}/\mathcal{G}) = L_g.$$

We remark that in this example the alternative supervisor

$$\mathcal{S}^\circ = (S^\circ, \phi^\circ)$$

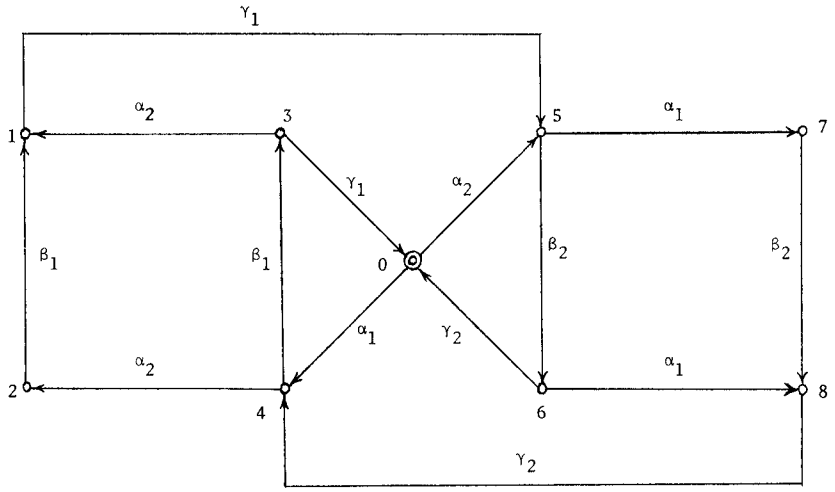


FIG. 11.3. *Example* 1: *Recognizer for* L_g.

[8] Alternatively the generator of Fig. 11.3 could be taken as providing the definition of L_g.

State	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
ϕ	00	00	10	00	10	01	00	01	00
ϕ°	00	10	10	10	10	01	01	01	01
π	x_0'	x_1'	x_1'	x_2'	x_2'	x_3'	x_3'	x_4'	x_4'

FIG. 11.4. *Example* 1: *Control data for* $\mathscr{S}$, $\mathscr{S}^\circ$ *and* $\mathscr{S}'$.

defined by setting $S^\circ = S$, and with ϕ° as tabulated in Fig. 11.4, determines exactly the same language controlled in $\mathscr{G}$ as $\mathscr{S}$ does, namely

$$L(\mathscr{S}^\circ/\mathscr{G}) = L(\mathscr{S}/\mathscr{G}) = \bar{L}_g.$$

To verify this statement note that, for instance, states x_1, x_3 of S are entered only on the occurrence of the event β_1; but since β_1 can be immediately followed in $\mathscr{G}_1$ only by γ_1, the enablement of β_1 by ϕ° in x_1 and x_3 can have no effect on the language controlled in $\mathscr{G}$.

It may be left to the reader to verify that $\mathscr{S}^\circ$ is complete with respect to $\mathscr{G}$. From $\mathscr{S}^\circ$ we construct a new supervisor $\mathscr{S}' = (S', \phi')$ and a projection $\pi: \mathscr{S}^\circ \to \mathscr{S}'$ as tabulated in Fig. 11.4; the result is displayed in Fig. 11.5. By Proposition 8.1

$$(L_m, L_c, L)(\mathscr{S}'/\mathscr{G}) = (L_m, L_c, L)(\mathscr{S}^\circ/\mathscr{G}),$$

namely control and marking action are preserved. The simplified supervisor $\mathscr{S}'$ has just 5 states and is equivalent, in fact, to a queue (of maximum length 2) that stores events α in order of occurrence and is popped by the corresponding events γ.

12. Example 2. In a manufacturing system we consider two machines M_1, M_2 connected in tandem and separated by a buffer B (Fig. 12.1). Each machine M_i is modeled as a CDEP over the alphabet $\{\alpha_i, \beta_i, \lambda_i, \mu_i\}$ and having binary-valued controls $\{u_i, v_i\}$ (Fig. 12.2). The machine states are IDLE (I), WORKING (W) and DOWN (D).

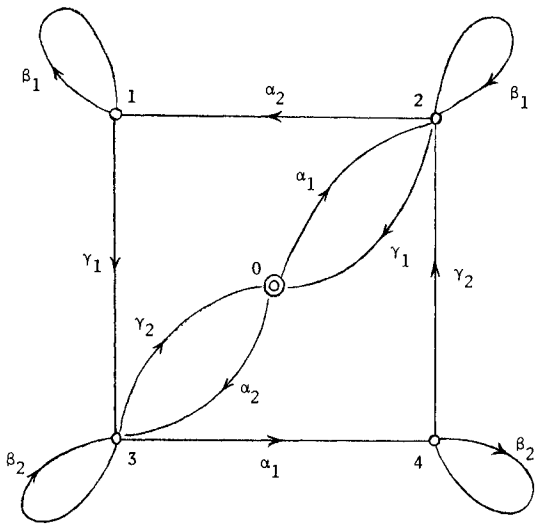


FIG. 11.5. *Example* 1: *Quotient supervisor* $\mathscr{S}'$.

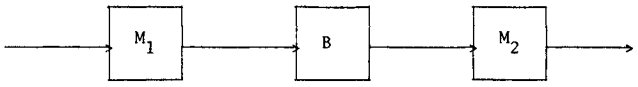


FIG. 12.1. *Example 2: Machines coupled by a buffer.*

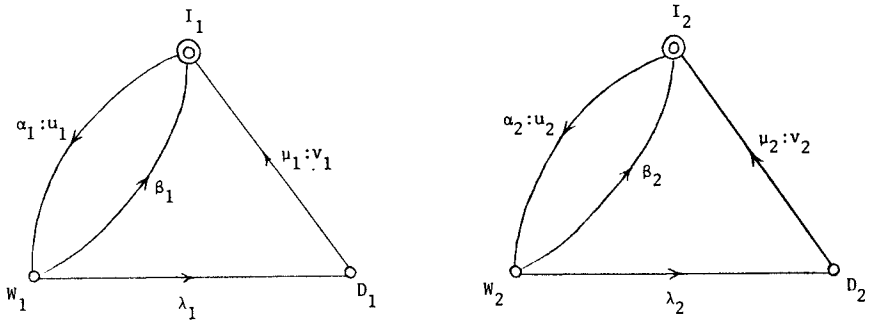


FIG. 12.2. *Example 2: State diagrams of machines.*

The control u enables/disables the transition from I to W ($u = 1$ allows M to "accept a workpiece"); while v enables/disables the transition from D to I ($v = 1$ means, when M is in state D, that M is "under repair"). The buffer B has one slot, i.e., is EMPTY (E) or FULL (F); it is not a CDEP but simply an automaton driven by M_1 and M_2 (Fig. 12.3). The system operates as follows. Machine M_1 takes a workpiece (event α_1), and either successfully completes processing and passes the workpiece to the buffer (event β_1); or breaks down and discards the workpiece (event λ_1), but in that case may later be repaired (event μ_1). Machine M_2 operates in the same way, but takes its workpiece from the buffer B, provided one is there.

The problem is to manipulate the controls in order to satisfy the four requirements stated informally below.

(i) M_1 executes α_1 only if B is in E.
(ii) M_2 executes α_2 only if B is in F (thereby driving B to E).
(iii) M_1 cannot execute α_1 while M_2 is in D_2.
(iv) If M_1 is in D_1 and M_2 is in D_2 then $v_1 = 0$.

Condition (iv) means that if both machines are down then M_2 must be repaired before M_1.

As in Example 1, we shall not formalize these requirements or present the details of how the legal language L_g is derived from them, but merely display the result. The language L_g that incorporates requirements (i)-(iv) with the system constraints is generated as shown in Fig. 12.4. The corresponding recognizer defines a supervisor $\mathcal{S}^\circ$ such that $L(\mathcal{S}^\circ/\mathcal{G}) = \bar{L}_g$; the control pattterns are tabulated in Fig. 12.6. It can be verified that $\mathcal{S}^\circ$ admits the quotient $\mathcal{S}$ displayed in Fig. 12.5; the required projection is also tabulated in Fig. 12.6. The quotient represents a reduction from 12 states to 6.

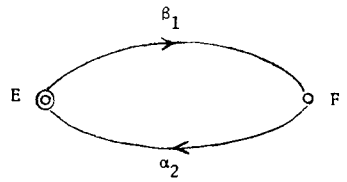


FIG. 12.3. *Example 2: State diagram of buffer.*

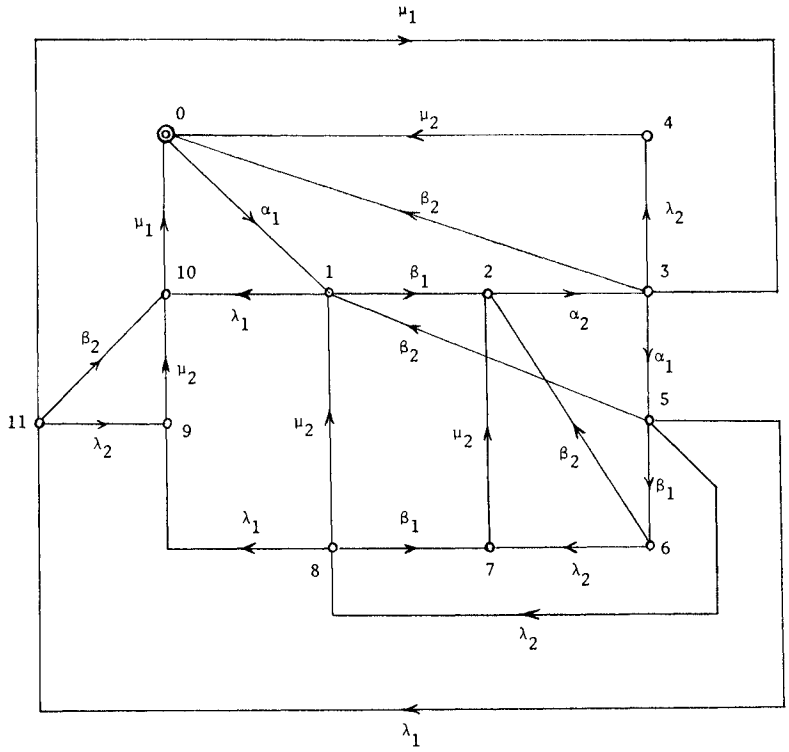


FIG. 12.4. *Example* 2: *Recognizer for* L_g.

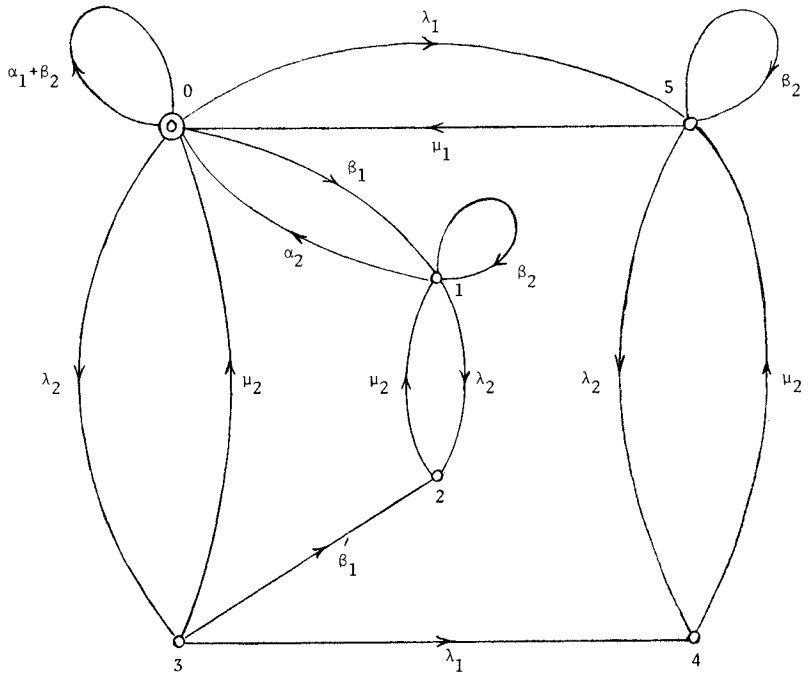


FIG. 12.5. *Example* 2: *Quotient supervisor* $\mathcal{S}$.

x^0	x	u_1	v_1	u_2	v_2
0	0	1	-	0	-
1	0	1*	-	0	-
2	1	0	-	1	-
3	0	1	-	0	-
4	3	0	-	0	1
5	0	1*	-	0	-
6	1	0	-	1*	-
7	2	0	-	-	1
8	3	0	-	0	1
9	4	0	0	0	1
10	5	-	1	0	-
11	5	-	1	0	-

FIG. 12.6. *Example* 2: *Control data for* $\mathcal{S}$ *and* $\mathcal{S}^0$. *Assignments* (*) *are determined by consistency for the quotient*; *entries* (-) *may be assigned arbitrarily, consistent with the quotient.*

As will be shown in a future article, it can actually be obtained directly from two modular "subsupervisors," of which one is modeled on the buffer, and the other incorporates the logic of breakdown and repair.

13. Conclusion. In this article we have introduced a broad class of controlled discrete event processes together with some general concepts and results relating to their control or "supervision." Our main conclusion, the Quotient Structure Theorem, is similar in spirit to the Internal Model Principle of regulator theory; it may be roughly paraphrased by saying that "supervisors must be modeled on the task to be accomplished."

In future articles we shall discuss constructive methods for computing the supremal controllable (or closed controllable) sublanguage of a given language, as well as concrete methods for system specification and supervisor synthesis.

REFERENCES

R. AVEYARD [1974], *A boolean model for a class of discrete event systems*, IEEE Trans. Syst. Man and Cyb., SMC-4, pp. 249-258.

J. BEAUQUIER AND M. NIVAT [1980], *Application of formal language theory to problems of security and synchronization*, in Formal Language Theory—Perspective and Open Problems, R. V. Book, ed., Academic Press, New York, pp. 407-454.

S. EILENBERG [1974], *Automata, Languages, and Machines, Vol.* A, Academic Press, New York.

G. S. FISHMAN [1978], *Principles of Discrete Event Simulation*, John Wiley, New York.

B. T. HAILPERN AND S. S. OWICKI [1983], *Modular verification of computer communication protocols*, IEEE Trans. Comm., COM-31, pp. 56-68.

M. A. HARRISON [1965], *Introduction to Switching and Automata Theory*, McGraw-Hill, New York.

C. A. R. HOARE [1983], *Notes on communicating sequential processes*, Tech. Monograph PRG-33, Programming Research Group, Oxford Univ. Computing Laboratory, Oxford.

J. E. HOPCROFT AND J. D. ULLMAN [1979], *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, MA.

G. MILNE AND R. MILNER [1979], *Concurrent processes and their syntax*, J. Assoc. Comput. Mach., 26, pp. 302-321.

H. NIJMEIJER [1983], *Nonlinear Multivariable Control: A Differential Geometric Approach*, thesis, Rijksuniv. te Groningen, the Netherlands.

D. PARK [1981], *Concurrency and automata on infinite sequences*, in Theoretical Computer Science, Lecture Notes in Computer Science 104, Springer-Verlag, New York, pp. 167-183.

J. L. PETERSON [1981], *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Englewood Cliffs, NJ.

A. PNUELI [1979], *The temporal semantics of concurrent programs*, in Semantics of Concurrent Computation, Lecture Notes in Computer Science 70, Springer-Verlag, New York, pp. 1–20.

P. J. RAMADGE [1983], *Control and Supervision of Discrete Event Processes*, Ph.D. thesis, Dept. Electrical Engineering, Univ. Toronto, Toronto, Ontario.

P. J. RAMADGE AND W. M. WONHAM [1982a], *Supervisory control of discrete event processes*, in Feedback Control of Linear and Nonlinear Systems, Lecture Notes in Control and Information Sciences 39, Springer-Verlag, New York, pp. 202–214.

———, [1982b], *Supervision of discrete event processes*, Proc. 21st IEEE Conference on Decision and Control, December, pp. 1228–1229.

———, [1984], *Supervisory control of a class of discrete event processes*, Proc. Sixth International Conference Analysis and Optimization of Systems, Nice, June 1984, in Analysis and Optimization of Systems, A. Bensoussan and J. L. Lions, eds., Lecture Notes in Computer and Information Science 63, Springer-Verlag, New York, 1984, Part 2, pp. 477–498.

A. C. SHAW [1978], *Software descriptions with flow expressions*, IEEE Trans. Software Engrg., SE-4 (3), pp. 242–254.

M. W. SHIELDS [1979], *COSY train journeys*, Rpt. ASM/67, Computing Laboratory, Univ. Newcastle-upon-Tyne.

M. STEENSTRUP, M. A. ARBIB AND E. G. MANES [1981], *Port automata and the algebra of concurrent processes*, Computer and Information Science Tech. Rpt. 81-25, Univ. Massachusetts, Amherst, MA.

G. SZÁSZ [1963], *Introduction to Lattice Theory*, Academic Press, New York.

W. M. WONHAM [1979], *Linear Multivariable Control: A Geometric Approach*, sec. ed., Springer-Verlag, New York.

B. P. ZEIGLER [1984], *Multifacetted Modelling and Discrete Event Simulation*, Academic Press, New York.