On Supervisory Control of Concurrent Discrete-Event Systems

Yosef Willner* Michael Heymann†

March 27, 2002

Abstract

When a discrete-event system P consists of several subsystems $P_1, \ldots, P_n$ that operate concurrently, a natural approach to the supervisory control problem is to synthesize a 'local' controller S_i for each subsystem P_i and operate the individually controlled subsystems S_i/P_i concurrently. Such an approach is called *concurrent supervisory control* and is closely related to *decentralized supervisory control* as studied in [4] and [5]. In the present paper simple and easily computable conditions are developed, that guarantee when concurrent supervisory control can achieve the optimal behavior achievable by a global supervisor. To achieve the optimal behavior, two specific concurrent control strategies are introduced.

Key words: discrete event systems, supervisory control, concurrency, decentralization, separability.

1 Introduction

In recent years a control theory for a general class of Discrete Event Systems (DES) has been proposed by Ramadge and Wonham [1-3]. In this theory, the DES is modelled as a tuple

$$P = (Q, \Sigma, \delta, q_0) \ ,$$

*Department of Electrical Engineering, Technion - Israel Institute of Technology, Haifa 32000, Israel.

†Department of Computer Science, Technion - Israel Institute of Technology, Haifa 32000, Israel. Completed in part while the author was on leave as a NRC-Senior Research Associate at NASA-Ames Research Center, Moffett Field, CA 94035.

where Q is a set of *states*, Σ is a set of symbols called the *transition alphabet*, $\delta : \Sigma \times Q \to Q$ is a (partial) function called the *state transition function*, and $q_0 \in Q$ is the initial state.[1]

Let Σ^* denote the set of all strings over Σ, including the empty string ϵ. A subset $\mathcal{L} \subset \Sigma^*$ is called a *language* over Σ. A string $u \in \Sigma^*$ is a *prefix* of $v \in \Sigma^*$, denoted $u < v$, if for some string $w \in \Sigma^*$, $v = uw$. The *closure* of $\mathcal{L} \subset \Sigma^*$, denoted $\overline{\mathcal{L}}$, is the set of all prefixes of strings in $\mathcal{L}$. $\mathcal{L}$ is closed if $\overline{\mathcal{L}} = \mathcal{L}$.

For a DES P, the transition function $\delta : \Sigma \times Q \to Q$ is extended to $\delta : \Sigma^* \times Q \to Q$ in the standard way. Then the language *generated by* P is defined by

$$\mathcal{L}(P) = \{t \in \Sigma^* \mid \delta(t, q_0) \text{ is defined}\} .$$

The transition alphabet Σ consists of two disjoint subsets, ths set Σ_c of *controlled* events that can be disabled by external control, and the set Σ_u of *uncontrolled* events that cannot be prevented from occurring.

A supervisor S for P is a device that follows the behavior of P and at each state of P disables a subset of the controlled events. The language generated by the (*closed-loop*) system, i.e., by the combination of P and S, is denoted by $\mathcal{L}(S/P)$.

Let $\mathcal{M}$ be a closed sublanguage of $\mathcal{L}(P)$. Then there exists a supervisor S such that $\mathcal{L}(S/P) = \mathcal{M}$ if and only if $\mathcal{M}$ is *controllable* with respect to $\mathcal{L}(P)$ (see [1]), namely, if and only if

$$\overline{\mathcal{M}}\Sigma_u \bigcap \mathcal{L}(P) \subset \overline{\mathcal{M}} .$$

For any language $\mathcal{M} \subset \mathcal{L}(P)$ there exists a supremal (in the sense of languages inclusion) controllable sublanguage of $\mathcal{M}$, denoted by $\sup C(\mathcal{M})$. Let a specification of legal behavior for P be given as a nonempty sublanguage $\mathcal{E} \subset \Sigma^*$. The basic control problem is then to construct a supervisor S such that $\mathcal{L}(S/P) \subset \mathcal{E}$. It is then known (see e.g. [1]) that the optimal solution to this problem is the supervisor S that satisfies the condition that $\mathcal{L}(S/P) = \sup C(\mathcal{L}(P) \cap \mathcal{E})$.

Cieslak et al. in [4] and Lin and Wonham in [5] developed *decentralized* control schemes, in which instead of a single global supervisor, there are several local supervisors that operate concurrently. Each local supervisor observes and controls only a subset of the events of the system. In [4] a necessary and sufficient condition was introduced which guarantees that the decentralized control scheme achieves the global optimal behavior.

Most practical real-life discrete event systems are *concurrent* DESs, i.e. they consist of a large number of subsystems that operate concurrently. The event sets of the subsystems may be (pairwise) disjoint, in which case the systems operate concurrently but asynchronously,

[1] We omit from the model the subset of marked states, which is not used in the present paper.

or they may partially overlap, yielding a degree of synchronization through the simultaneity of events.

In the present paper we examine the decentralized control of concurrent DESs. We assume that the DES P is composed of concurrent subsystems, P_i, with event sets Σ_i, $i = 1, 2, \ldots, n$. Suppose that for each subsystem P_i there is a *local* supervisor S_i that observes and controls only the events in Σ_i. The global controlled system can then be obtained as the concurrent operation of the individually controlled subsystems $S_i/P_i, \quad 1 \leq i \leq n$.

While the results of [4] and [5] apply to the present situation as well, they have been developed without any assumption on the structure of the DES under consideration. In the present paper we explore some specific properties of decentralized supervisory control that do not apply in the case of unstructured systems.

The paper is organized as follows. In Section 2 we define the key property of separability of languages and describe some properties of separable languages. The formal definition of concurrent DES and the correspondence between a global controllable language and local controllable languages are discussed in Section 3. In Section 4 we formulate the concurrent supervisory control problem and develop two concurrent control schemes under which optimal global behavior can be achieved.

2 Separable Languages

Let $\phi \neq \hat{\Sigma} \subset \Sigma$ and define the projection $\pi : \Sigma^* \to \hat{\Sigma}^*$ by

$$\begin{aligned} \pi(\epsilon) &= \epsilon \\ \pi(t\sigma) &= \begin{cases} \pi(t)\sigma & \text{if } \sigma \in \hat{\Sigma} \\ \pi(t) & \text{otherwise.} \end{cases} \end{aligned} \tag{2.1}$$

The effect of π on $t \in \Sigma^*$ is to delete from t all symbols not in $\hat{\Sigma}$.

Let $\Sigma_i, \quad 1 \leq i \leq n$, be nonempty (not necessarily disjoint) alphabets, and write

$$\Sigma = \bigcup_{i=1}^{n} \Sigma_i \ . \tag{2.2}$$

Define projections $\pi_i : \Sigma^* \to \Sigma_i^*$ as described in (2.1). For a finite set of languages $\{\mathcal{L}_i \subset \Sigma_i^*\}_{i=1}^n$ we define the *synchronized product* of $\{\mathcal{L}_i\}$, denoted $\|_{i=1}^n \mathcal{L}_i$, as

$$\|_{i=1}^n \mathcal{L}_i = \{t \in \Sigma^* \mid (\forall\, i) \ \ \pi_i(t) \in \mathcal{L}_i\} \ . \tag{2.3}$$

Clearly, (2.3) is equivalent to the following definition

$$\|_{i=1}^n \mathcal{L}_i = \bigcap_{i=1}^{n} \pi_i^{-1}(\mathcal{L}_i) \ . \tag{2.4}$$

If $(\forall\, i \neq j)\ \Sigma_i \bigcap \Sigma_j = \phi$ then $\|_{i=1}^n \mathcal{L}_i$ is the usual language interleaving, or *shuffle product* (see [9]) and if $(\forall\, i, j)\ \Sigma_i = \Sigma_j$ then $\|_{i=1}^n \mathcal{L}_i = \bigcap_{i=1}^n \mathcal{L}_i$.

Definition 2.1 Let Σ be defined as in (2.2) and let $\mathcal{L} \subset \Sigma^*$. We shall say that $\mathcal{L}$ is *separable* with respect to (w.r.t.) $\{\Sigma_i\}_{i=1}^n$ if there exists a set of languages $\mathcal{L}_i \subset \Sigma_i^*,\ \ 1 \leq i \leq n$, called a *generating set* of $\mathcal{L}$, such that $\mathcal{L} = \|_{i=1}^n \mathcal{L}_i$.

Examples

Let $n = 2,\ \ \Sigma_1 = \{\alpha, \beta, \gamma_1\},\ \ \Sigma_2 = \{\alpha, \beta, \gamma_2\}$. The language $\mathcal{L} = \overline{\gamma_1\gamma_2 + \gamma_2\gamma_1}$ is separable w.r.t. $\{\Sigma_1, \Sigma_2\}$ since $\mathcal{L} = \|_{i=1}^2 \mathcal{L}_i$, where $\mathcal{L}_1 = \overline{\gamma_1}$ and $\mathcal{L}_2 = \overline{\gamma_2}$. Consider now the language $\mathcal{L} = \overline{\gamma_1\gamma_2}$. Clearly the languages $\{\overline{\gamma_1}, \overline{\gamma_2}\}$ are not a generating set of $\mathcal{L}$. It is easy to verify that there do not exist languages $\mathcal{L}_1$ and $\mathcal{L}_2$ such that $\mathcal{L} = \|_{i=1}^2 \mathcal{L}_i$, thus $\mathcal{L}$ is not separable. Let $\mathcal{L} = \overline{\gamma_1\gamma_2\alpha + \gamma_2\gamma_1\alpha}$. It is easy to verify that for $\mathcal{L}_{11} = \overline{\gamma_1\alpha}$ and $\mathcal{L}_{12} = \overline{\gamma_2\alpha}$ as well as for $\mathcal{L}_{21} = \overline{\gamma_1\alpha\beta}$ and $\mathcal{L}_{22} = \overline{\gamma_2\alpha\alpha}$, $L = \|_{i=1}^2 \mathcal{L}_{ji}$ for $j = 1, 2$.

As we can see from the last example, for a separable language there may exist several generating sets. For a nonempty separable language $\mathcal{L} \subset \Sigma^*$, let $B(\mathcal{L})$ be the set of all generating sets of $\mathcal{L}$, that is,

$$B(\mathcal{L}) = \{(\mathcal{L}_{1\alpha}, \mathcal{L}_{2\alpha}, \ldots, \mathcal{L}_{n\alpha}) \mid (\forall\, \alpha)\mathcal{L}_{i\alpha} \subset \Sigma_i^* \quad \text{and} \quad \mathcal{L} = \|_{i=1}^n \mathcal{L}_{i\alpha}\}\ . \tag{2.5}$$

The following two properties are automatic

$$\|_{i=1}^n \left(\bigcup_{\alpha} \mathcal{L}_{i\alpha} \right) \supset \mathcal{L}\ , \tag{2.6}$$

$$\|_{i=1}^n \left(\bigcap_{\alpha} \mathcal{L}_{i\alpha} \right) \subset \mathcal{L}\ . \tag{2.7}$$

For each $i,\ \ 1 \leq i \leq n$, let $\{\mathcal{L}_{i\alpha}\}$ be the family of languages consisting of the i-th components of all the elements in $B(\mathcal{L})$. A question that arises is whether the inclusion (2.7) is, in fact, an equality. In other words, whether $B(\mathcal{L})$ contains a generating set $(\underline{\mathcal{L}}_1, \ldots, \underline{\mathcal{L}}_n)$ such that each $\underline{\mathcal{L}}_i$ is the infimal element (in the sense of language inclusion) of $\{\mathcal{L}_{i\alpha}\}$. The answer is given in the following proposition.

Proposition 2.1 *For each $i,\ \ 1 \leq i \leq n$, the nonempty family $\{\mathcal{L}_{i\alpha}\}$ is closed under arbitrary intersection and*

$$\bigcap_{\alpha} \mathcal{L}_{i\alpha} = \pi_i(\mathcal{L})\ . \tag{2.8}$$

Corollary 2.1 *A language $\mathcal{L} \subset \Sigma^*$ is separable w.r.t. $\{\Sigma_i\}_{i=1}^n$ if and only if*

$$\mathcal{L} = \|_{i=1}^n \pi_i(\mathcal{L})\ . \tag{2.9}$$

Corollary 2.2 *Let* $(\mathcal{L}_1, \ldots, \mathcal{L}_n) \in B(\mathcal{L})$. *Then for each* i,

$$(\mathcal{L}_1, \ldots, \mathcal{L}_{i-1}, \pi_i(\mathcal{L}), \mathcal{L}_{i+1}, \ldots, \mathcal{L}_n) \in B(\mathcal{L}) \ .$$

The last corollary states that if we replace in a generating set $(\mathcal{L}_1, \ldots, \mathcal{L}_n)$ of $\mathcal{L}$, any of the components, say $\mathcal{L}_{i_1}, \ldots, \mathcal{L}_{i_k}$, with the corresponding projections $\pi_{i_1}(\mathcal{L}), \ldots, \pi_{i_k}(\mathcal{L})$, then the result is also a generating set of $\mathcal{L}$.

The next proposition states that the family $\{\mathcal{L}_{i\alpha}\}$ is closed also under union.

Proposition 2.2 *For each* i, $1 \leq i \leq n$, *the nonempty family* $\{\mathcal{L}_{i\alpha}\}$ *is closed under arbitrary union, and*

$$\bigcup_{\alpha} \mathcal{L}_{i\alpha} = \pi_i(\mathcal{L}) \bigcup (\Sigma_i^* - \pi_i(\hat{\mathcal{L}}_i)) \ , \tag{2.10}$$

where $\hat{\mathcal{L}}_i = \bigcap_{\substack{j=1 \\ j \neq i}}^{n} \pi_j^{-1}(\pi_j(\mathcal{L}))$.

We conclude this section with the following result.

Proposition 2.3 *The set of all separable (w.r.t.* $\{\Sigma_i\}_{i=1}^n$*) languages, over* Σ, *is closed under arbitrary intersection.*

3 Concurrent Discrete Event Systems

Interconnected subsystems can be represented by modelling each subsystem as a DES and describing the connections between them in such a way that the combined DES can be obtained in a mechanical way. One such connection is as follows (see e.g. Hoare [10, section 2.3]) Let $P_i = (Q_i, \Sigma_i, \delta_i, q_{0i})$, $i = 1, 2 \ldots, n$, be DESs with control partitions $\Sigma_i = \Sigma_{ic} \bigcup \Sigma_{iu}$ such that

$$(\forall\, i \neq j) \ \ \Sigma_{iu} \bigcap \Sigma_j = \phi \ . \tag{3.1}$$

This assumption means that there is no synchronization between uncontrolled events of one subsystem with events of another (because we regard uncontrolled events as resulting from spontaneous and unpredictable behavior of the individual subsystems). For each $\sigma \in \Sigma$ let $In(\sigma) = \{i \mid \sigma \in \Sigma_i\}$. The model of the entire system P is defined by

$$P = (Q, \Sigma, \delta, q_0) \ , \tag{3.2}$$

where $Q = Q_1 \times Q_2 \times \cdots \times Q_n$, $\Sigma = \bigcup_{i=1}^{n} \Sigma_i$ (with $\Sigma_c = \bigcup_{i=1}^{n} \Sigma_{ic}$, $\Sigma_u = \bigcup_{i=1}^{n} \Sigma_{iu}$), where $q_0 = (q_{01}, q_{02}, \ldots, q_{0n})$, and where $\delta : \Sigma \times Q \to Q$ is given by

$$\delta(\sigma, (q_1, q_2, \ldots, q_n)) = \begin{cases} (q_1', q_2', \ldots, q_n') & \text{where } q_i' = \delta_i(\sigma, q_i) \text{ for all } i \in In(\sigma) \\ & \text{and } q_i' = q_i \text{ for all } i \notin In(\sigma) \end{cases} \quad \begin{array}{l} \text{provided that} \\ \delta_i(\sigma, q_i) \text{ is defined} \\ \text{for all } i \in In(\sigma) \end{array}$$

$$\text{undefined} \qquad \text{otherwise}$$

We shall use the notation $P = \|_{i=1}^{n} P_i$ for the entire system[2].

Thus, in the system P, an event that belongs to exactly one subsystem can occur asynchronously and independently. But if an event belongs to several subsystems, then it occurs in the composite system if and only if it occurs simultaneously in all the corresponding subsystems. In particular, if the Σ_i are all disjoint, then P is the shuffle product of P_i, (see [1]).

The next proposition follows immediately from the definition of P.

Proposition 3.1 *Let $P = \|_{i=1}^{n} P_i$, then $\mathcal{L}(P) = \|_{i=1}^{n} \mathcal{L}(P_i)$.*

Example 3.1 *We consider two machines P_1, P_2 that operate concurrently. The P_i are DESs with state diagrams as shown in Fig. 3.1. In state I_i, P_i is idle and in state W_i it is working. Initially the system is in state (I_1, I_2). The sets of uncontrolled events are $\Sigma_{1u} = \{b_1\}$, $\Sigma_{2u} = \{b_2\}$, respectively. The system operates as follows. There is an external storage of three types of workpieces for processing. Machine P_1 takes a workpiece either from the external storage (event a_{11}, if the workpiece is of the first type, or event a_{12} if it is of the second type), or from machine P_2 (shared event α). If the processing of a workpiece is successfully completed, then P_1 passes the workpiece to an external storage of processed workpieces (event b_1). Otherwise, it passes the workpiece to machine P_2 (shared event β). Machine P_2 operates in the same way, but takes from the external storage only workpieces of the third type (event a_2). The state diagram of the entire system $P = \|_{i=1}^{2} P_i$ is given in Fig. 3.2.*

As we mentioned in Section 1, the controllability property plays the key role in supervisor synthesis problems. For concurrent DES $P = \|_{i=1}^{n} P_i$ the controllability property has two aspects; global controllability, i.e.; w.r.t. $\mathcal{L}(P)$, and local controllability, i.e.; w.r.t. $\mathcal{L}(P_i)$. More specifically, let $\mathcal{M} \subset \mathcal{L}(P)$ be a "global" sublanguage, the corresponding local

[2]We denote by $\|_{i=1}^{n}$ the synchronized product of languages as well as the concurrent composition of systems. The precise meaning will be clear from the context.

languages are $\pi_i(\mathcal{M})$, $1 \leq i \leq n$. The question now of interest is whether the global controllability of $\mathcal{M}$ implies the local controllability of $\pi_i(\mathcal{M})$ or vice versa. (Similar questions appear in [4] for unstructured systems). The answer is given in the Main Lemma.

Lemma 3.1 (**Main Lemma**).

Consider a DES $P = \|_{i=1}^{n} P_i$.

(a) *If* $\mathcal{M} \subset \mathcal{L}(P)$ *is closed and controllable w.r.t.* $\mathcal{L}(P)$, *then for each* i, $1 \leq i \leq n$, $\pi_i(\mathcal{M}) \subset \mathcal{L}(P_i)$ *and* $\pi_i(\mathcal{M})$ *is closed and controllable w.r.t.* $\mathcal{L}(P_i)$.

(b) *Let* $\mathcal{M}_i \subset \mathcal{L}(P_i)$, $1 \leq i \leq n$, *be closed and controllable w.r.t.* $\mathcal{L}(P_i)$ *and write* $\mathcal{M} = \|_{i=1}^{n} \mathcal{M}_i$. *Then* $\mathcal{M} \subset \mathcal{L}(P)$ *and* $\mathcal{M}$ *is closed and controllable w.r.t.* $\mathcal{L}(P)$.

The Main Lemma motivates all the results about synthesis of supervisors for concurrent DES that we develop in the next Section.

4 The Supremal Controllable Legal Behavior and Supervisor Synthesis

Consider a DES $P = \|_{i=1}^{n} P_i$ as defined in (3.2). Let $\mathcal{E} \subset \Sigma^*$ be a nonempty closed language, which we interpret as a specification of legal behavior for P. The basic control problem is to synthesize a control device whose task is to restrict the behavior of P to within the specified legal behavior $\mathcal{E}$. As was mentioned in Section 1, a global supervisor S achieves optimal behavior of P by synthesizing the language

$$\mathcal{K} = \sup C(\mathcal{L}(P) \bigcap \mathcal{E}) \subset \Sigma^* \ . \tag{4.1}$$

Instead of using a single global supervisor, we propose a decentralized control scheme, called *concurrent control*. By concurrent control we mean that for each substem P_i there is a local supervisor S_i. Each supervisor S_i observes and controls only the events that belong to Σ_i. The concurrent operation of all controlled subsystems S_i/P_i generates a new global system P_{cl}, namely $P_{cl} = \|_{i=1}^{n} (S_i/P_i)$. The behavior of P_{cl} is given by

$$\tilde{\mathcal{K}} := \mathcal{L}(P_{cl}) = \|_{i=1}^{n} \mathcal{L}(S_i/P_i) = \bigcap_{i=1}^{n} \pi_i^{-1}(\mathcal{L}(S_i/P_i)) \ . \tag{4.2}$$

In the remainder of this section, we discuss the relations between the language $\tilde{\mathcal{K}}$, that can be achieved by concurrent control, and the global optimal language $\mathcal{K}$.

The following proposition states the (fairly obvious) fact that concurrent control can never be better than global control.

Proposition 4.1 *Suppose that there exist local supervisors S_i, $1 \leq i \leq n$, such that $\tilde{\mathcal{K}} \subset \mathcal{E}$. Then $\tilde{\mathcal{K}} \subset \mathcal{K}$.*

The following theorem gives conditions under which a concurrent control scheme can achieve the optimal global behavior.

Theorem 4.1 *Let $P = \|_{i=1}^{n} P_i$, where $P_i = (Q_i, \Sigma_i, \delta_i, q_{0i})$, $1 \leq i \leq n$. There exist local supervisors S_i, $1 \leq i \leq n$, such that $\tilde{\mathcal{K}} = \mathcal{K}$ if and only if $\mathcal{K}$ is separable w.r.t. $\{\Sigma_i\}_{i=1}^{n}$.*

Remark: In [4] the following decentralized control scheme was examined. Let P be a DES with controlled events Σ_c, where Σ_c is the union of n subsets $\Sigma_{1c}, \ldots, \Sigma_{nc}$. Also given are n event sets $\Delta_1, \ldots, \Delta_n$ and maps (or *masks*) $M_i : \Sigma \to \Delta_i \bigcup \{\epsilon\}$. A decentralized supervisor is a collection of local supervisors S_i, where each supervisor S_i observes only the symbols belonging to Δ_i and controls only the events of Σ_{ic}. It was proved in Lemma 4.2 of [4] that $\mathcal{L}((\wedge_i S_i)/P) = \mathcal{K}$, (with $\mathcal{K}$ as defined in (4.1) above), if and only if $\mathcal{K}$ is $(\{M_i\}, \{\Sigma_{ic}\}, \mathcal{L}(P))$-controllable. Namely, for $\sigma \in \Sigma_c$, for a set $\{s_i\}$ of strings in $\mathcal{K}$ indexed by $i \in In(\sigma)$, and for $s' \in \mathcal{K}$

(1) $s_i\sigma \in \mathcal{K}$ for all $i \in In(\sigma)$,

(2) $s'\sigma \in \mathcal{L}(P)$

(3) $M_i(s_i) = M_i(s')$ for all $i \in In(\sigma)$

together imply

$$s'\sigma \in \mathcal{K}\,.$$

It can be shown that if the masks M_i are deletion masks, that is, $M_i(\sigma) \in \{\sigma, \epsilon\}$ for all $\sigma \in \Sigma$, and if $\Sigma_{ic} \subset \Delta_i$ for all $i = 1, 2, \ldots, n$, then $\mathcal{K}$ is $(\{M_i\}, \{\Sigma_{ic}\}, \mathcal{L}(P))$-controllable if and only if

$$\mathcal{K} = \mathcal{L}(P) \bigcap_{i=1}^{n} \pi_i^{-1}(\pi_i(\mathcal{K}))\,.$$

If, in addition, $P = \|_{i=1}^{n} P_i$, where $P_i = (Q_i, \Sigma_i, \delta_i, q_{0i})$, and $\Delta_i \subset \Sigma_i$ for all $i = 1, 2, \ldots, n$, then $\mathcal{K}$ is $(\{M_i\}, \{\Sigma_{ic}\}, \mathcal{L}(P))$-controllable if and only if

$$\mathcal{K} = \bigcap_{i=1}^{n} \pi_i^{-1}(\pi_i(\mathcal{K}))\,.$$

Thus under the assumptions of Theorem 4.1, the condition of [4] is equivalent to the separability of $\mathcal{K}$. The separability property can be verified algorithmically. The algorithmic aspects will be discussed later.

Example 4.1 *Consider the system $P = \|_{i=1}^{2} P_i$ as defined in example 3.1. Suppose that our aim is to satisfy the following specification. Each processed workpiece of the first type must be passed to the external storage only by P_1. In other words, if P_1 passes a workpiece of the first type to P_2 then P_2 must return this workpiece to P_1. Formally, the legal behavior $\mathcal{E}$ is described by the state diagram displayed in Fig. 4.1. In this diagram $\Sigma = \{a_{11}, a_{12}, b_1, a_2, b_2, \alpha, \beta\}$ and we use also arcs labelled with a set of symbols instead of an individual arc for each symbol. The supremal controllable sublanguage of $\mathcal{L}(P) \bigcap \mathcal{E}$, $\mathcal{K}$ and its projections on Σ_1^* and Σ_2^* are given in Figs. 4.2 and 4.3, respectively.*

It is easily verified that $\mathcal{K} = \|_{i=1}^{2} \pi_i(\mathcal{K})$, whence the optimal behavior can be achieved by using only a local supervisor for P_1 such that $\mathcal{L}(S_1/P_1) = \pi_1(\mathcal{K})$.

In the case that the language $\mathcal{K} \subset \Sigma^*$ is regular it can be determined whether $\mathcal{K}$ is separable w.r.t. subsets $\Sigma_1, \Sigma_2, \ldots, \Sigma_n$. Let $A = (X, \Sigma, \xi, x_0)$ be a deterministic automaton with m states such that $\mathcal{L}(A) = \mathcal{K}$. Clearly the separability of $\mathcal{K}$ can be checked simply by employing the definition of separable languages. The complexity of such an algorithm is $O(m^{n+1})$. Clearly, this exponential complexity makes the verification of separability generally infeasible.

Next we introduce an algorithm for checking the separability of a language $\mathcal{K}$ when the subsets $\Sigma_1, \ldots, \Sigma_n$ are pairwise disjoint. It is easy to see that in this case the separability of a language is equivalent to its being the shuffle product of its projections. We shall see that the algorithm has only polynomial complexity.

First we prove the following motivating lemma.

Lemma 4.1 *Let $\mathcal{L} \subset \Sigma^*$ be a prefix closed language, and let $\Sigma_1, \ldots, \Sigma_n$ be pairwise disjoint subsets of Σ. Then $\mathcal{L}$ is separable w.r.t. $\{\Sigma_i\}_{i=1}^{n}$ if and only if for any $t \in \mathcal{L}$ and $\sigma \in \Sigma_i$*

$$t\sigma \notin \mathcal{L} \Longrightarrow (\forall\, s \in \mathcal{L}) \quad \textit{either} \quad \pi_i(s) \neq \pi_i(t), \quad \textit{or} \quad s\sigma \notin \mathcal{L}\,.$$

We introduce now the algorithm. Again, let $A = (X, \Sigma, \xi, x_0, X)$ be a deterministic automaton that accepts $\mathcal{K}$ and has m states. Let $\Sigma_1, \ldots, \Sigma_n$ be pairwise disjoint subsets of Σ.

Algorithm 4.1 **(1)** *For each $i = 1, 2, \ldots, n$, construct the automaton $A_i = (X, \Sigma_i, \xi_i, X_{i0}X)$ as defined in step (2) of Algorithm 4.1.*

(2) *For each pair (i, x), where $i \in \{1, 2, \ldots, n\}$ and $x \in X$, define $A_{ix} = (X, \Sigma_i, \xi_i, X_{i0}\{x\})$, and define $d_i(x) = \{\sigma \in \Sigma_i \mid \xi_i(\sigma, x) = \phi\}$.*

(2a) *Construct the product automaton* $A_{ix} \times A_i = (X \times X, \Sigma_i, \delta, X_{i0} \times X_{i0}, \{x\} \times X)$. *Define* $X_{ix} \subset X$ *to be the set of all states* $x' \in X$ *such that* (x, x') *is an accessible state in* $A_{ix} \times A_i$, *that is*

$$X_{ix} = \{x' \in X \mid (\exists t \in \Sigma_i^*, \exists x_{10}, x_{20} \in X_{i0})(x, x') \in \delta(t, (x_{10}, x_{20})\} \; .$$

(2b) *If there exists* $x' \in X_{ix}$ *such that* $d_i(x') \not\supset d_i(x)$, *then stop.*

(3) *If all the pairs* (i, x) *were checked then stop. Else repeat step (2) for another pair* (i, x).

The intuitive concept of the algorithm is as follows. For each (i, x), $\mathcal{L}(A_{ix})$ is the set of all strings $\pi_i(t)$, such that $t \in \mathcal{K}$ and $\xi(t, x_0) = x$, and $d_i(x)$ is the set of all $\sigma \in \Sigma_i$ such that $t\sigma \notin \mathcal{K}$. By constructing $A_{ix} \times A_i$ the algorithm identifies the set X_{ix} which is the set of all states $x' \in X$ such that there exist $s, t \in \mathcal{K}$ which satisfies that $\pi_i(s) = \pi_i(t)$ and that $\xi(t, x_0) = x$ and $\xi(s, x_0) = x'$. Now if $d_i(x') \not\supset d_i(x)$ then there exists $\sigma \in \Sigma$ such that $t\sigma \notin \mathcal{K}$ and $s\sigma \in \mathcal{K}$ which contradicts the condition of Lemma 4.1.

The correctness of this algorithm is formally stated in the following proposition.

Proposition 4.2 $\mathcal{K}$ *is separable w.r.t.* $\{\Sigma_i\}_{i=1}^n$ *if and only if Algorithm 4.2 stops at step (3).*

The complexity of algorithm 4.2 is $O(n \cdot m^3)$.

In the system P_{cl} there are two mechanisms that restrict the behavior of the subsystems P_i. One is the control mechanism, i.e.; the local supervisor S_i prevents the occurrence of some events in P_i. The second is the synchronization mechanism that prevents a shared event $\sigma \in \Sigma_i$ from occurring in P_i if σ cannot occur simultaneously in all subsystems P_j that satisfy $\sigma \in \Sigma_j$. These mechanisms can overlap in the sense that an event $\sigma \in \Sigma_i$ can be prevented from occurring, in some local state of P_i, by both mechanisms.

The possibility of overlapping disabling mechanisms leads us to think about two possible control strategies. The first is, the *most restrictive control*, in which the behaviors of the subsystems are restricted by the supervisors *as much as possible*. In this case the effect of the synchronization mechanism on the subsystems behavior is minimal. The second is, the *least restrictive control*, in which the effect of the synchronization mechanism is maximal and the supervisors disable an event only if it is not already prevented from occuring by the synchronization mechanism. To formalize this issue, let $C(\mathcal{K})$ be a class of all sets of supervisors that by acting concurrently synthesize the language $\mathcal{K}$,

$$C(\mathcal{K}) = \{\{S_{1\alpha}, \ldots, S_{n\alpha}\} \mid (\forall\, \alpha) \quad \mathcal{L}\,(\|_{i=1}^n (S_{i\alpha}/P_i)) = \mathcal{K}\} \; .$$

The following theorem addresses the issue of the most restrictive control.

Theorem 4.2 *If $\mathcal{K}$ is separable w.r.t. $\{\Sigma_i\}_{i=1}^n$, then there exists a set of supervisors $\{\overline{S_i}, \ldots, \overline{S_n}\} \in C(\mathcal{K})$ such that for all $\{S_1, \ldots, S_n\} \in C(\mathcal{K})$,*

$$\mathcal{L}(\overline{S_i}/P_i) = \pi_i(\mathcal{K}) \subset \mathcal{L}(S_i/P_i), \qquad i = 1, \ldots, n \ .$$

There does not, in general, exist a least restrictive control law. To see this, let us examine the following example.

Example 4.2 *Consider the systems P_1, P_2, P and the language $\mathcal{K}$ as shown in Fig. 4.4. Let (S_{11}, S_{12}) and (S_{21}, S_{22}) be pairs of supervisors such that*

$$\mathcal{L}(S_{11}/P_1) = \overline{a_1}, \quad \mathcal{L}(S_{12}/P_2) = \mathcal{L}(P_2), \ \mathcal{L}(S_{21}/P_1) = \ CL(P_1) \ \text{ and } \mathcal{L}(S_{22}/P_2) = \overline{a_2} \ .$$

Clearly, for $j = 1, 2$, $\mathcal{L}(\|_{i=1}^2 (S_{ji}/P_i)) = \mathcal{K}$. Thus there does not exit a pair (S_1, S_2) that synthesizes the language $\mathcal{K}$ and that S_1 and S_2 are the least restrictive supervisors for P_1 and P_2, respectively.

We wish to show that if the supervisors S_i are synthesized according to some order, then there exists a least restrictive control law. Let $I = \{1, 2, \ldots, n\}$ and let $\prec$ be an order on the elements of I. We denote by $i_1, \ldots, i_n$ the elements of I that are ordered according to $\prec$ i.e.; $i_j \prec i_{j+1}$ for $j = 1, 2, \ldots, n-1$. Suppose that we synthesize the supervisors S_i, $1 \leq i \leq n$, according to $\prec$, namely the first supervisor is S_{i_1}, the second is S_{i_2} and so on. In practical applicatins the order $\prec$ can frequently be determined according to the ability to implement the control law. That is, for $i, j \in I$, $i \prec j$ if it is more difficult or less desirable to implement the disablement of events in P_i than in P_j.

The question now of interest is whether for each j we can synthesize a least restrictive supervisor S_{i_j} under the condition that the previous supervisors $S_{i_1}, \ldots, S_{i_{j-1}}$ are already given. In other words, does there exist a set of supervisors $\{\overline{S_{i_1}}, \ldots, \overline{S_{i_n}}\} \in C(\mathcal{K})$ such that $\overline{S_{i_1}}$ is the least restrictive supervisor for P_{i_1}, $\overline{S_{i_2}}$ is the least restrictive supervisor for P_{i_2} under the condition that $\overline{S_{i_1}}$ is given and so on. For simplicity we assume below, without loss of generality, that $i_j = j$ for $j = 1, \ldots, n$.

Suppose that $\mathcal{K} \subset \mathcal{L}(P)$ is a nonempty separable language (w.r.t. $\{\Sigma_i\}_{i=1}^n$) and denote by $BP(\mathcal{K})$ the set of all generating sets of $\mathcal{K}$ where each component is a closed sublanguage of the corresponding subsystem, that is,

$$BP(\mathcal{K}) = \left\{ (\mathcal{L}_{1\alpha}, \ldots, \mathcal{L}_{n\alpha}) \mid (\forall\, \alpha)\mathcal{L}_{i\alpha} \subset \mathcal{L}(P_i), \ \mathcal{L}_{i\alpha} \ \text{ is closed and } K = \bigcap_{i=1}^{n} \pi_i^{-1}(\mathcal{L}_{i\alpha}) \right\} \ .$$

We define, inductively, a sequence of closed languages $\mathcal{K}_i \subset \mathcal{L}(P_i)$, $i = 1, 2, \ldots, n$, as follows:

$$\mathcal{K}_1 = \bigcup_{\alpha} \mathcal{L}_{1\alpha} \ .$$

Let $(\mathcal{K}_1, \ldots, \mathcal{K}_{l-1})$ be given and denote by $BP_l(\mathcal{K}_1, \ldots, \mathcal{K}_{l-1}) \subset BP(\mathcal{K})$ the collection of all generating sets of $\mathcal{K}$ in which the first $l-1$ components are $\mathcal{K}_1, \ldots, \mathcal{K}_{l-1}$. That is,

$$BP_l(\mathcal{K}_1, \ldots, \mathcal{K}_{l-1}) = \{(\mathcal{K}_1, \ldots, \mathcal{K}_{l-1}\mathcal{L}_{l\beta}, \ldots, \mathcal{L}_{n\beta}) \mid (\forall\, \beta)(\mathcal{K}_1, \ldots, \mathcal{K}_{l-1}, \mathcal{L}_{l\beta}, \ldots, \mathcal{L}_{n\beta}) \in BP(\mathcal{K})\} \ .$$

Clearly $BP_1 = BP(\mathcal{K})$. The l-th language is defined by

$$\mathcal{K}_l = \bigcup_{\beta} \mathcal{L}_{l\beta} \ .$$

The properties of the sequence $(\mathcal{K}_1, \ldots, \mathcal{K}_n)$ are given in the following two lemmas the first of which states that the above construction is sound, i.e.,

Lemma 4.2

$$(\mathcal{K}_1, \ldots, \mathcal{K}_n) \in BP(\mathcal{K}) \ .$$

The meaning of the last lemma is as follows. If $\mathcal{K}$ is separable, then among all the generating sets of $\mathcal{K}$ for which the components that correspond to the first j subsystems are fixed, there is a set in which the component corresponding to the $j+1$ subsystem is maximal.

Lemma 4.3 *For each j, $1 \leq j \leq n$, the language $\mathcal{K}_j$ is controllable w.r.t. $\mathcal{L}(P_j)$.*

The last two lemmas motivate the following result.

Theorem 4.3 *If $\mathcal{K}$ is separable w.r.t. $\{\Sigma_i\}_{i=1}^n$, then there exists a set of supervisors $\{\overline{S_1}, \ldots, \overline{S_n}\} \in C(\mathcal{K})$ such that for each l, $l = 1, 2, \ldots, n$, the following property holds. If $\{S_1, \ldots, S_l, \ldots, S_n\} \in C(\mathcal{K})$ such that for all $j < l$, $\mathcal{L}(S_j/P_j) = \mathcal{L}(\overline{S_j}/P_j)$, then*

$$\mathcal{L}(S_l/P_l) \subset \mathcal{L}(\overline{S_l}/P_l) = \mathcal{K}_l \ .$$

We present now an algorithm to compute the languages $\mathcal{K}_j$. Let

$$\hat{\mathcal{K}_j} = \left[\bigcap_{l=1}^{j-1} \pi_l^{-1}(\mathcal{K}_l)\right] \bigcap \left[\bigcap_{l=j+1}^{n} \pi_l^{-1}(\pi_l(\mathcal{K}))\right], \quad j = 1, 2, \ldots, n \ .$$

The following result states that for given languages $\mathcal{K}_1, \ldots, \mathcal{K}_{j-1}$, the supremal element $\mathcal{K}_j$ can be computed as follows. First construct $\hat{\mathcal{K}_j}$ the synchronized product of $\mathcal{K}_1, \ldots, \mathcal{K}_{j-1}$ and of the infimal elements $\pi_{j+1}(\mathcal{K}), \ldots, \pi_n(\mathcal{K})$. Then compute the supremal sublanguage $\mathcal{L}_{j,max}$ of $\mathcal{L}(P_j)$, such that the synchronized product of $\hat{\mathcal{K}_j}$ and $\mathcal{L}_{j,max}$ yields $\mathcal{K}$.

Proposition 4.3

$$\mathcal{K}_j = \pi_j(\mathcal{K}) \bigcup [\mathcal{L}(P_j) - \pi_j(\hat{\mathcal{K}_j})], \qquad j = 1, 2, \ldots, n \ .$$

Example 4.3 *Let $P = \|_{i=1}^{2} P_i$ be the system described in Example 3.1. When P_1 and P_2 operate concurrently, there is a possibility that one subsystem, say P_1, is an "exploiter". Namely, P_1 does not process any workpiece, it passes all the workpieces to P_2. Clearly this is an undesirable situation. Suppose that we require that the passing of workpieces between the subsystems be fair. In particular, it may be required that if P_i passes a workpiece to P_j then it will not do so again until P_j passes a workpiece to P_i. Thus, a string t of the closed loop behavior is legal only if its projection on $\{\alpha+\beta\}^*$ belongs to $(\alpha\beta)^* + (\beta\alpha)^*$. The legal behavior $\mathcal{E}$ is described by the generator displayed in Fig. 4.5. The optimal language $\mathcal{K}$ and the projections of $\mathcal{K}$ on Σ_1^*, Σ_2^* are described in Figs. 4.6 and 4.7, respectively. It can be verified that $\mathcal{K} = \|_{i=1}^{2} \pi_i(\mathcal{K})$, so $\mathcal{K}$ is separable w.r.t. (Σ_1, Σ_2).*

Most Restrictive Control

¿From Theorem 4.2 it follows that the most restrictive control can be implemented by supervisors S_1, S_2 such that $\mathcal{L}(S_i/P_i) = \pi_i(\mathcal{K})$.

Least Restrictive Control

Case (i): $1 \prec 2$

In this case $i_1 = 1$ and $i_2 = 2$, that is, first we synthesize the least restrictive supervisor S_1 and afterwards the least restrictive supervisor S_2 under the assumption that S_1 is given. To construct S_i we use Proposition 4.4.

$$\begin{aligned} \hat{\mathcal{K}_1} &= \pi_2^{-1}(\pi_2(\mathcal{K})), \\ \mathcal{K}_1 &= \pi_1(\mathcal{K}) \bigcup [\mathcal{L}(P_1) - \pi_1(\pi_2^{-1}(\pi_2(\mathcal{K})))] \, . \end{aligned}$$

It can be verified that $\pi_1(\pi_2^{-1}(\pi_2(\mathcal{K}))) \bigcap \mathcal{L}(P_1) = \pi_1(\mathcal{K})$, thus

$$\mathcal{K}_1 = \mathcal{L}(P_1) \, .$$

Now we compute $\mathcal{K}_2$.

$$\begin{aligned} \hat{\mathcal{K}_2} &= \pi_1^{-1}(\mathcal{L}(P_1)), \\ \mathcal{K}_2 &= \pi_2(\mathcal{K}) \bigcup [\mathcal{L}(P_2) - \pi_2(\pi_1^{-1}(\mathcal{L}(P_1)))] \, . \end{aligned}$$

It is easy to verify that $\pi_2(\pi_1^{-1}(\mathcal{L}(P_1))) = \Sigma_2^*$, thus

$$\mathcal{K}_2 = \pi_2(\mathcal{K}) \, .$$

Hence the least restrictive control under the assumption that $1 \prec 2$ is the supervisor S_2 such that

$$\mathcal{L}(S_2/P_2) = \pi_2(\mathcal{K}) \, .$$

Case (ii): $2 \prec 1$

By using the same arguments as in the previous case, it can be shown that the least restrictive control under the assumption that $2 \prec 1$ *is the supervisor* S_1 *such that*

$$\mathcal{L}(S_1/P_1) = \pi_1(\mathcal{K}) \ .$$

All the results up to this point are concerned with the language $\mathcal{K}$. To compute $\mathcal{K}$ it is necessary to compute the global behavior of the system, $\mathcal{L}(P)$. It is well known that the number of states of P, where $P = \|_{i=1}^{n} P_i$ increases exponentially with n, the number of subsystems. This fact often makes the computation of $\mathcal{L}(P)$ infeasible. Therefore, it is important to find conditions which ensure that the equality $\tilde{\mathcal{K}} = \mathcal{K}$ holds and which can be verified without the computation of $\mathcal{L}(P)$. Such conditions are given in the following theorem.

Theorem 4.4 *Consider a DES* $P = \|_{i=1}^{n} P_i$*, where* $P_i = (Q_i, \Sigma_i, \delta_i, q_{0i})$*,* $1 \leq i \leq n$*. Suppose that* $\mathcal{E} \subset \Sigma^*$ *is separable w.r.t.* $\{\Sigma_i\}_{i=1}^{n}$ *and let* $(\mathcal{E}_1, \ldots, \mathcal{E}_n)$ *be a generating set of* $\mathcal{E}$*, where* $\mathcal{E}_i$ *are closed languages. Let* $\mathcal{K}_i = \sup C_i(\mathcal{L}(P_i) \bigcap \mathcal{E}_i)$*, where* $\sup C_i(\cdot)$ *denotes the supremal controllable sublanguage w.r.t.* $\mathcal{L}(P_i)$*. Then there exist supervisors* S_i*,* $1 \leq i \leq n$*, such that*

$$\begin{aligned} \mathcal{L}(S_i/P_i) &= \mathcal{K}_i, \quad 1 \leq i \leq n, \quad \text{and} \\ \tilde{\mathcal{K}} &= \mathcal{K} \ . \end{aligned}$$

If the legal behavior $\mathcal{E}$ is given and is separable, then to synthesize the local supervisors it is necessary to compute the language $\pi_i(\mathcal{E})$ and $\mathcal{K}_i = \sup C_i(\mathcal{L}(P_i) \bigcap \pi_i(\mathcal{E}))$, $i = 1, 2, \ldots, n$. So the computation of $\mathcal{L}(P)$ is not needed.

So far, we assumed that the desired legal behavior of the DES is represented by the language $\mathcal{E}$ within which, the language of the controlled system must reside. In some cases, however, the desired legal behavior is specified by n individual requirements each of which constitutes a restriction on the sequences in $\mathcal{L}(P_i)$, $1 \leq i \leq n$. In other words, the i-th requirement specifies the desired legal behavior of P_i. Let $\mathcal{E}_i \subset \Sigma_i^*$ be the set of all strings in Σ_i^* that satisfy the i-th requirement, we assume that $\mathcal{E}_i$ are closed languages. Suppose further that the desired global legal behavior is given by

$$\mathcal{E} = \bigcap_{i=1}^{n} \pi_i^{-1}(\mathcal{E}_i) \ . \tag{4.3}$$

Since $(\mathcal{E}_1, \ldots, \mathcal{E}_n)$ is a generating set of $\mathcal{E}$, then by using Theorem 4.4 we get the following result.

Theorem 4.5 *Consider a DES* $P = \|_{i=1}^{n} P_i$ *and let* $\mathcal{E}$ *be as defined in (4.6). Then there exist supervisors* S_i*,* $1 \leq i \leq n$*, such that* $\mathcal{L}(S_i/P_i) = \sup C_i(\mathcal{L}(P_i) \bigcap \mathcal{E}_i)$ *and* $\tilde{\mathcal{K}} = \mathcal{K}$*.*

The last theorem states that if the constraints are given locally, then the synthesis procedure, of supervisors that achieve the optimal behavior, is based only on local information.

5 Conclusion

A specific version of decentralized supervisory control for concurrent discrete-event systems has been introduced. A necessary and sufficient condition and another sufficient condition (which can be verified by using only local information) were obtained which guarantee that the decentralized control is equivalent to global control.

The special aspects of the concept of least restrictive supervision in concurrent systems were also discussed.

References

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

[**2**] Ramadge P.J. and Wonham W.M., "Modular feedback logic for discrete even systems", SIAM J. on Control and Optimization, 25(5), pp. 1202-1218, September 1987.

[**3**] Wonham W.M. and Ramadge P.J., "On the supremal controllable sublanguage of a given language", SIAM J. on Control and Optimization, 25(3), pp. 637-659, May 1987.

[**4**] Cieslak C., Declaux C., Fawaz A. and Varaiya P., "Supervisory control of discrete event systems with partial observations", IEEE Trans. Automatic Control, Vol. AC-33, pp. 249-260, March 1988.

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

[**6**] Lu W. and Chen Y., "Structuralized control logic for discrete event system", Proc. of the International Conference on Control and Application, Jerusalem, Israel, 1989.

[**7**] Kuratowski K., Topology, Academic Press, New York, 1966.

[**8**] Tadmor G. and Maimon O., "Control of large discrete event systems: constructive algorithms", IEEE Trans. Automatic Control, Vol. AC-34, pp. 1164-1168, November 1989.

[**9**] Eilenberg S., Automata, languages and machines, Volume A, Academic Press, New York, 1974.

[**10**] Hoare C.A.R., Communicating sequential processes, Prentice Hall, Englewood Cliffs, 1985.

[**11**] Hopcroff J.E. and Ullman J.D., Introduction to Automata Theory Languages and Computation, Addison-Wesley Pub. Co. Reading, 1979.