Masked Prioritized Synchronization for Interaction and Control of Discrete Event Systems

Ratnesh Kumar and Michael Heymann

***Abstract*—This paper extends the formalism of prioritized synchronous composition (PSC), proposed by Heymann for modeling interaction (and control) of discrete event systems, to permit system interaction with their environment via interface masks. This leads to the notion of masked prioritized synchronous composition (MPSC), which we formally define. MPSC can be used to model interaction of systems at single as well as multiple interfaces. We show that MPSC can alternatively be computed by "unmasking" the PSC of "masked" systems, thereby establishing a link between MPSC and PSC. We next prove that MPSC is associative and thus suitable for modeling and analysis of supervisory control of discrete event systems. Finally, we use MPSC of a discrete event plant and a supervisor for controlling the plant behavior and show (constructively) that under the absence of "driven" events, controllability together with normality of the given specification serve as conditions for the existence of a supervisor. This extends the results on supervisory control, which permits control and observation masks to be associated with the plant only.**

***Index Terms*—Controllability, discrete event system, masked prioritized synchronization, normality, supervisory control.**

I. Introduction

SUPERVISORY control of discrete event systems (DESs) has been studied using prioritized synchronous composition (PSC) [4] in [5], [15], [10]–[12], [1], [3]. In PSC, each system component possesses an event priority set specifying the set of events whose execution in the environment requires its participation. Thus, when many systems are interacting, an event can occur if and only if all the systems having priority over the event can actively participate. In this case, the event occurs synchronously in all such systems; otherwise the event is "blocked" from occurring. The systems that do not have priority over the event will also participate in the event execution if they can, which is known as *broadcast synchronization;* otherwise the event takes place without the participation of such systems. Thus the systems with no priority over an event cannot block its execution. The *event control function* of Inan [7] assigns state-dependent event priority sets, thereby generalizing the notion of PSC. However, when applied to supervisory control, the event control functions are taken to be constant, thus becoming equivalent to PSC. It should be noted that in the PSC formalism each system is associated with an event priority set, and the events that belong to that set are determined by the application. For example, in the context of supervisory control, this is determined by the controllability/drivability property of the events as explained below.

The formalism of PSC models the interaction between discrete event plants and supervisors quite effectively when all the events are completely observable at their interface, the need to ensure that interacting systems are control compatible [16], [17] is eliminated. In this setting, the priority set of the plant includes the events that are uncontrollable (such as sensor and failure events) and controllable (such as actuator events), whereas that of the supervisor includes the events that are controllable and *driven* (such as command and control-policy switch events). Thus the controllable events are in the priority sets of both the plant and supervisor and can be blocked by either of them, whereas the uncontrollable (respectively, driven) events can only be blocked by the plant (respectively, supervisor).

In many situations, systems interact via interfaces. For example, in an elevator system, when a user requests an elevator, one of the elevators responds to the request. An internal logic decides which elevator should respond, but this information is masked from the user. Similarly, in a pumping station consisting of several pumps, a command to start a pump may be nonspecific, and the decision of which pump to start may be resolved by an internal logic that is masked from the agent issuing the command. These examples illustrate that certain events of the system may be masked at the system interface from the *control* perspective. Similarly, events may also be masked from the *observation* perspective. For example, different kinds of failure events may be reported to the environment as the same type of failure; thus masking the difference between the failure events from the environment. In fact, we can view *unobservable* events as those that are masked to be indistinguishable from "silent" events. When systems interact through nonidentity interface masks, then their interaction through PSC requires that the systems be observation compatible with respect to their interface masks, which is a limitation of PSC [16], [17]. Thus it is sensible to generalize the notion of PSC to describe prioritized synchronization of systems interacting through nonidentity interface masks. This then will allow us to model interactions of systems without the need to ensure that they are control or observation compatible [16], [17].

Manuscript received January 16, 1998; revised December 22, 1999. Recommended by Associate Editor K. Rudie. The work of R. Kumar was supported in part by the Center for Manufacturing Systems, University of Kentucky, in part by the National Science Foundation under Grants NSF-ECS-9409712 and NSF-ECS-9709796, and in part by the Office of Naval Research under Grant ONR-N00014-96-1-5026. The work of M. Heymann was supported in part by the Technion Fund for Promotion of Research.

R. Kumar was with NASA Ames Research Center, Moffett Field, CA. He is now with the Department of Electrical Engineering, University of Kentucky, Lexington, KY 40506-0046.

M. Heymann was with NASA Ames Research Center, Moffett Field, CA. He is now with the Department of Computer Science, Technion—Israel Institute of Technology, Haifa 32000 Israel.

Publisher Item Identifier S 0018-9286(00)10608-7.

An effort to generalize PSC in such a direction was first presented in [16] and [17], and the generalization was called *masked composition* (MC). In MC, each system was associated with two types of mask functions: a control mask that identified events "from the control perspective" and an observation mask that identified events "from the observation perspective." Modeling the interaction of systems in this formalism is difficult owing to its complexity.

In this paper, we introduce an intuitive generalization of PSC, which we call *masked prioritized synchronous composition* (MPSC). MPSC retains the basic concept of PSC in that each system has its own event priority set, i.e., the set of events in which it must participate in order for them to occur in the composition (equivalently, the set of events it can block by not participating). The new concept that we add here to generalize PSC is that each system is allowed to interact with its environment via interfaces that are modeled as event mask functions. (The masks are similar in spirit to the masks introduced in [2] and [13] but were restricted there to the observation process.) Event mask functions as presented here constitute "static" interfaces, in that they mask the events independently of the system evolution history. It is possible to consider more general "dynamic," trace-dependent event masks such as the "reporter maps" in the work on hierarchical supervisory control [19], [18]. That, however, we do not explore in this paper.

Since a system may have multiple interfaces, the mask functions are unique not to the system but rather to the particular interface of the system. When two or more systems interact at a common interface, they use their respective mask functions to map their respective "internal" events to the "external" or interface events. Since internal events of a system that are masked to a common external event interact with the environment indistinguishably, there is no loss in assuming that a mask function respect the priority partition of the events, i.e., two events can be masked to a common external event if and only if they are either both or none in the priority set of the system.

Below, we formally define masked prioritized synchronization of discrete event systems modeled by nondeterministic state machines. MPSC can be used to model the interaction of systems at multiple interfaces. We show that the MPSC of systems can alternatively be computed by "unmasking" the PSC of suitably "masked" systems, thereby establishing a link between MPSC and PSC. We prove that when systems are connected at a common interface, their MPSC satisfies the desirable property of associativity, showing that MPSC provides a useful formalism for modeling system interaction. Finally, we study the problem of MPSC-based control of a discrete event plant with priority set being the entire event set. The plant interacts via MPSC with a supervisor, modeled as a deterministic state machine, for which the event priority set and the interface mask are given. Under the assumption that the set of driven events is empty, we show that there exists a supervisor so that the behavior of the MPSC of the plant and the supervisor projected onto the event set of the plant equals a specification language if and only if the specification language is controllable and normal. (The more general case of nonempty driven events set is reported in a recent paper [8].) The proof of this existence result is constructive and provides a way to compute a supervisor whenever one exists. These results extend the existing results on supervisory control that permits control masks (limited to projection type) and observation masks to be associated with the plant only. We illustrate our design via a simple example. In particular, this example shows the effect of having masks associated also with the supervisor, and of having general nonprojection-type control mask associated with the plant.

II. Notation and Preliminaries

Given an event set Σ, we let Σ^* denote the set of all finite-length sequences of events from Σ, called *traces*, including the trace of zero length, denoted ϵ. For an event set Σ, we use $\overline{\Sigma}$ to denote $\Sigma \cup \{\epsilon\}$. A subset of Σ^* is called a *language*. Given trace $s \in \Sigma^*$, we let $|s|$ denote its length. For a language $H \subseteq \Sigma^*$, the *prefix closure* of H, denoted $pr(H)$, is the set of all prefixes of traces from H. H is called *prefix closed* if $H = pr(H)$.

Nondeterministic state machines (NSMs) are used to model discrete event systems. An NSM P is a five-tuple $P := (X_P, \Sigma_P, \delta_P, x_P^0, X_P^m)$, where X_P is its set of states, Σ_P is its set of events, δ_P: $X_P \times (\Sigma_P \cup \{\epsilon\}) \to 2^{X_P}$ is its transition function, $x_P^0 \in X_P$ is its initial state, and $X_P^m \subseteq X_P$ is its set of marked, or final, states. For any set of states $\hat{X} \subseteq X_P$ and set of events $\Sigma \subseteq \Sigma_P$, the notation $\delta_P(\hat{X}, \Sigma)$ is used to denote $\bigcup_{x \in \hat{X}} \bigcup_{\sigma \in \Sigma} \delta_P(x, \sigma)$. P is called deterministic if $|\delta_P(x, \sigma)| \leq 1$ for all $x \in X_P$ and $\sigma \in \Sigma_P$, and $|\delta_P(x, \epsilon)| = 0$. A triple $(x, \sigma, x') \in X_P \times (\Sigma_P \cup \{\epsilon\}) \times X_P$ is called a *transition* if $x' \in \delta_P(x, \sigma)$; If $\sigma = \epsilon$, the transition is called silent or an ϵ-transition. Given an interface mask M: $\Sigma_P \to \Sigma \cup \{\epsilon\}$ from the internal events Σ_P to external interface events Σ, the "masked" NSM $M(P) := (X_P, \Sigma, \delta_{M(P)}, x_P^0, X_P^m)$ is obtained by replacing each transition $(x, \sigma, x') \in X_P \times \Sigma_P \times X_P$ of P by the transition $(x, M(\sigma), x')$. The interface mask M_P is extended to be defined over traces in Σ_P^* as follows:

$$M_P(\epsilon) := \epsilon; \quad \forall s \in \Sigma_P^*, \sigma \in \Sigma_P \colon M_P(s\sigma) := M_P(s)M_P(\sigma).$$

For $x \in X_P$, the *ϵ closure* of x, denoted $\epsilon_P^*(x)$, is the set of states reached by the execution of zero or more ϵ-transitions from the state x and is defined recursively as follows:

$$x \in \epsilon_P^*(x); \quad x' \in \epsilon_P^*(x) \Rightarrow \delta_P(x', \epsilon) \subseteq \epsilon_P^*(x).$$

The ϵ-closure map can also be used to extend the definition of transition function from events to traces. Thus we obtain δ_P^*: $X_P \times \Sigma_P^* \to 2^{X_P}$, which is defined inductively as follows:

$$\forall x \in X \colon [\delta_P^*(x, \epsilon) := \epsilon_P^*(x); \quad \forall s \in \Sigma_P^*, \\ \sigma \in \Sigma_P \colon \delta_P^*(x, s\sigma) := \epsilon_P^*(\delta_P(\delta_P^*(x, s), \sigma))]$$

where for any $\hat{X} \subseteq X_P$, $\delta_P(\hat{X}, \sigma) := \bigcup_{x \in \hat{X}} \delta_P(x, \sigma)$. Using this extended transition function, the *generated* and the *marked* languages of P, denoted, respectively, as $L(P)$ and $L_m(P)$, are defined as

$$L(P) := \left\{ s \in \Sigma_P^* | \delta_P^*(x_P^0, s) \neq \emptyset \right\};$$
$$L_m(P) := \left\{ s \in \Sigma_P^* | \delta_P^*(x_P^0, s) \cap X_P^m \neq \emptyset \right\}.$$

When two systems P and Q interact via prioritized synchronization, their interface events are the same as their own internal events since in PSC the interface masks of both P and Q are the identity function, i.e., $\Sigma_P = \Sigma_Q := \Sigma$. Letting $A, B \subseteq \Sigma$ denote the event priority sets of P and Q respectively, their PSC, denoted $P_A\|_B Q$, is the NSM defined as $P_A\|_B Q := (X, \Sigma, \delta, x^0, X^m)$, where $X := X_P \times X_Q$, $x^0 := (x_P^0, x_Q^0)$, $X^m := X_P^m \times X_Q^m$, and the transition function δ: $X \times (\Sigma \cup \{\epsilon\}) \to 2^X$ is defined as follows:

$$\forall x = (x_p, x_q) \in X, \sigma \in \Sigma:$$

$$\delta(x, \sigma) := \begin{cases} \delta_P(x_p, \sigma) \times \delta_Q(x_q, \sigma) & \text{if } \delta_P(x_p, \sigma) \neq \emptyset, \\ & \delta_Q(x_q, \sigma) \neq \emptyset \\ \delta_P(x_p, \sigma) \times \{x_q\} & \text{if } \delta_P(x_p, \sigma) \neq \emptyset, \\ & \delta_Q(x_q, \sigma) = \emptyset, \sigma \notin B \\ \{x_p\} \times \delta_Q(x_q, \sigma) & \text{if } \delta_Q(x_q, \sigma) \neq \emptyset, \\ & \delta_P(x_p, \sigma) = \emptyset, \sigma \notin A \\ \emptyset & \text{otherwise} \end{cases}$$

$$\delta(x, \epsilon) := [\delta_P(x_p, \epsilon) \times \{x_q\}] \cup [\{x_p\} \times \delta_Q(x_q, \epsilon)].$$

The event priority set of $P_A\|_B Q$ is given by $A \cup B$.

Thus, if an event is executable in both systems, then it occurs synchronously with the participation of both the systems. Otherwise if it is executable in only one of the systems, and the other system cannot block it (it is not in the event priority set of the other system), then it occurs without the participation of the other system. Finally, the composition can execute ϵ-transitions asynchronously. In the special case when the event priority sets of both systems are the entire event set Σ, then in their composition each event can only occur synchronously, resulting in the reduction of the PSC to the strict synchronous composition (SSC).

We recall the conditions of controllability and normality of discrete event systems, which we shall need later. Given an event set Σ, a prefix-closed language $H \subseteq \Sigma^*$, a set of events $\hat{\Sigma} \subseteq \Sigma$, and a mask function M defined over Σ, a language $K \subseteq H$ is said to be $(H, \hat{\Sigma})$-*controllable* [14] if

$$pr(K)\hat{\Sigma} \cap H \subseteq pr(K)$$

and is said to be (H, M)-*normal* [13] if

$$M^{-1}M(pr(K)) \cap H \subseteq pr(K).$$

We further recall that controllability (respectively, normality) is preserved under language union. Consequently, the supremal controllable (respectively, the supremal normal) sublanguage of a given language exists. Similarly, controllability (respectively, normality) of prefix-closed languages is preserved under language intersection, whence the infimal prefix-closed and controllable (respectively, the infimal prefix-closed and normal) superlanguage of any given language exists.

III. Masked Prioritized Synchronization

In this section, we formalize the notion of MPSC of two systems as discussed in the introduction. Two systems, modeled as NSMs P and Q, are connected as shown in Fig. 1. P and Q evolve over their "internal" events Σ_P and Σ_Q, respectively,

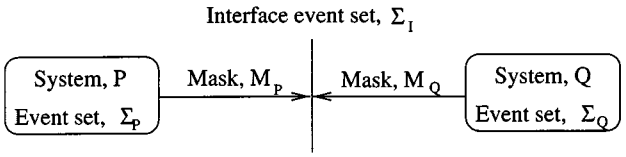

Fig. 1. P and Q interacting at a common interface.

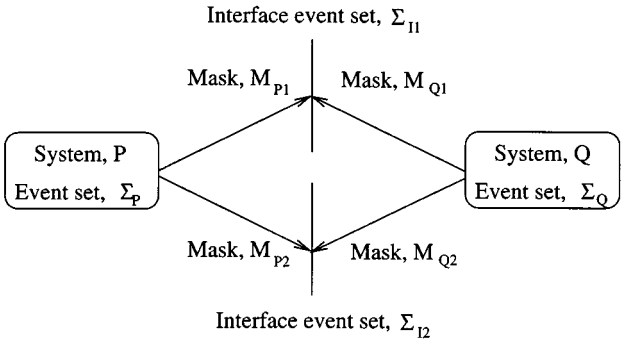

Fig. 2. P and Q interacting at two interfaces.

and their event priority sets are $A \subseteq \Sigma_P$ and $B \subseteq \Sigma_Q$, respectively. The systems interact at a common interface consisting of the interface (or "external") events Σ_I. The interface mask of P is given by M_P: $\Sigma_P \to \Sigma_I \cup \{\epsilon\}$, and that of Q is given by M_Q: $\Sigma_Q \to \Sigma_I \cup \{\epsilon\}$. The composed system is denoted by $P_A\|_B Q$, where the two interface masks M_P and M_Q are not explicitly included in the notation (to keep the notation simple).

The interface masks respect the event priority-partition consistency condition, that is

$$\forall \sigma, \sigma' \in \Sigma_P: [M_P(\sigma) = M_P(\sigma') \neq \epsilon] \Rightarrow [\sigma, \sigma' \in A] \vee [\sigma, \sigma' \notin A].$$

A similar condition is satisfied by the interface mask M_Q. Interface masks that respect the event priority-partition consistency condition are called *priority consistent masks*. If we define $\overline{M}_P(A) := M_P(A) - \{\epsilon\}$ and $\overline{M}_Q(B) := M_Q(B) - \{\epsilon\}$, then the priority consistency conditions can be rewritten as $M_P^{-1}\overline{M}_P(A) \subseteq A$ and $M_Q^{-1}\overline{M}_Q(B) \subseteq B$.

Remark 1: The formalism of MPSC is also applicable to systems interacting at multiple interfaces. For example, in Fig. 2, P and Q interact at a pair of interfaces. The interface mask of P for the two interfaces are M_{P1}: $\Sigma_P \to \overline{\Sigma}_{I1}$ and M_{P2}: $\Sigma_P \to \overline{\Sigma}_{I2}$, respectively, and similarly those for Q are M_{Q1}: $\Sigma_Q \to \overline{\Sigma}_{I1}$ and M_{Q2}: $\Sigma_Q \to \overline{\Sigma}_{I2}$, respectively. This scenario where P and Q interact at two interfaces can be transformed into the one where they interact at a single interface through masks M_P and M_Q, respectively, defined as follows:

$$\begin{aligned} M_P&: \Sigma_P \to \overline{\overline{\Sigma}_{I1} \times \overline{\Sigma}_{I2}} \\ M_P(\sigma) &:= (M_{P1}(\sigma), M_{P2}(\sigma)) \qquad \forall \sigma \in \Sigma_P \\ M_Q&: \Sigma_Q \to \overline{\overline{\Sigma}_{I1} \times \overline{\Sigma}_{I2}} \\ M_Q(\sigma) &:= (M_{Q1}(\sigma), M_{Q2}(\sigma)) \qquad \forall \sigma \in \Sigma_Q. \end{aligned}$$

Example 1: Consider a pumping station G consisting of two identical pumps P_1 and P_2 and a synchronizer R sharing a

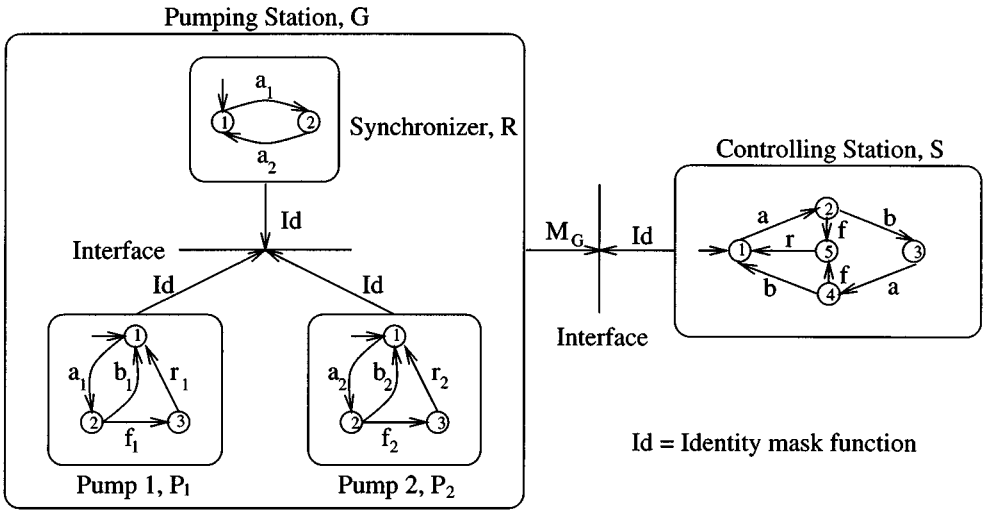

Fig. 3. Interacting pumping and controlling stations.

common interface, as shown in Fig. 3. (In all our examples, we assume that all states are marked and omit indicating the markings from the figures.) The event set of pump i consists of $\{a_i, b_i, f_i, r_i\}$, representing start, stop, fail, and repair, respectively, and that of the synchronizer consists of $\{a_1, a_2\}$. The synchronizer assures that the two pumps are started alternately and the first pump is started initially. The priority set of pump i consists of $\{a_i, b_i, f_i\}$, and that of the synchronizer consists of $\{a_1, a_2\}$. Also, the three systems interact with each other via the identity interface mask. So the MPSC of the three systems is equivalent to their PSC and can be obtained using the definition of PSC. The event priority set of the pumping station G is given by the union of the event priority sets of its three subsystems.

The pumping station G interacts with a controlling station S at a different interface and offers a start and a stop button and a fail indicator at this interface. The controlling station can start (event a), stop (event b), or issue a repair command (event r) whenever a fail (event f) is indicated. The priority set of the controlling station consists of $\{a, b, r\}$, and its interface mask is the identity function. The interface mask M_G of the pumping station identifies a_is to a, b_is to b, f_is to f, and r_is to r, at the interface with the controlling station. Clearly, M_G is priority consistent. Since a_is and b_is are priority events of G and are masked to a and b, respectively, which are priority events of S, a_is and b_is are controllable events. However, they are only nonuniquely controllable since both a_is (respectively, b_is) are enabled in G when a (respectively, b) is enabled in S. On the other hand, f_is are uncontrollable events since they are in the priority set of G and are masked to f, which is a nonpriority event of S. Similarly, r_is are the driven events since they are nonpriority events of G and are masked to r, which is a priority event of S.

Definition 1: Consider systems P and Q interacting at a common interface with events Σ_I, as shown in Fig. 1, their respective event priority sets A and B, and their respective priority consistent interface masks M_P and M_Q. Then the *masked prioritized composition* of P and Q is given by $P_A \| _B Q := (X, \Sigma, \delta, x^0, X^m)$, where $X := X_P \times X_Q$, $\Sigma := \overline{\Sigma}_P \times \overline{\Sigma}_Q$, $x^0 := (x_P^0, x_Q^0)$, $X^m := X_P^m \times X_Q^m$, and δ: $X \times (\Sigma \cup \{\epsilon\}) \to 2^X$, is defined as follows:

$$\forall x = (x_p, x_q) \in X, \quad \sigma = (\sigma_p, \sigma_q) \in \Sigma_P \times \Sigma_Q:$$

$$\delta(x, \sigma) := \begin{cases} \delta_P(x_p, \sigma_p) \times \delta_Q(x_q, \sigma_q), & \text{if } M_P(\sigma_p) = M_Q(\sigma_q) \neq \epsilon \\ & \delta_P(x_p, \sigma_p) \\ & \delta_Q(x_q, \sigma_q) \neq \emptyset \\ \emptyset, & \text{otherwise} \end{cases}$$

$$\delta(x, (\sigma_p, \epsilon)) := \begin{cases} \delta_P(x_p, \sigma_p) \times \{x_q\}, & \text{if } \delta_P(x_p, \sigma_p) \neq \emptyset, \text{ and} \\ & [[M_P(\sigma_p) = \epsilon] \\ & \vee [\delta_Q(x_q, M_Q^{-1} M_P(\sigma_p)) \\ & M_Q^{-1} M_P(\sigma_p) \cap B = \emptyset]] \\ \emptyset, & \text{otherwise} \end{cases}$$

$$\delta(x, (\epsilon, \sigma_q)) := \begin{cases} \{x_p\} \times \delta_Q(x_q, \sigma_q), & \text{if } \delta_Q(x_q, \sigma_q) \neq \emptyset, \text{ and} \\ & [[M_Q(\sigma_q) = \epsilon] \\ & \vee [\delta_P(x_p, M_P^{-1} M_Q(\sigma_q)) \\ & M_P^{-1} M_Q(\sigma_q) \cap A = \emptyset]] \\ \emptyset, & \text{otherwise} \end{cases}$$

$$\delta(x, \epsilon) := [\delta_P(x_p, \epsilon) \times \{x_q\}] \cup [\{x_p\} \times \delta_Q(x_q, \epsilon)].$$

The event priority set of $P_A \| _B Q$ is given by

$$A \oplus B := \left[A \times \overline{\Sigma}_Q\right] \cup \left[\overline{\Sigma}_P \times B\right].$$

Intuitively, P and Q interact by either 1) executing synchronously events σ_p and σ_q, respectively, whenever these are executable in the respective states and are masked to a common interface event that is observable at the interface, or 2) executing individual events (without participation of the other system) whenever either the event is unobservable at the interface or no event of the other system that has the same observation as

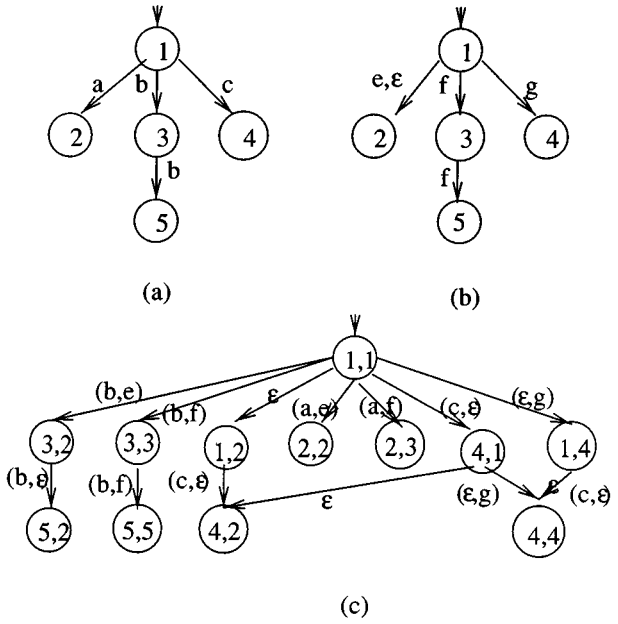

Fig. 4. Illustration of MPSC.

the former event is executable at the current state, and the event cannot be blocked by the other system (in the sense of PSC). An ϵ-transition can, of course, occur asynchronously in the composition. Note that if P and Q have m and n states, respectively, then the number of states in their MPSC is $O(mn)$, and hence there can be $O(mn)$ transitions defined at each state.

Example 2: Consider the state machines P and Q shown in Fig. 4(a) and (b). We have $\Sigma_P = \{a, b, c\}$, $\Sigma_Q = \{e, f, g\}$, $A = \{a, b\}$, and $B = \{e, f\}$. Also, $M_P(a) = M_P(b) = M_Q(e) = M_Q(f) \neq \epsilon$, $M_P(c) = \epsilon$, and $M_Q(g) = g$. The masked prioritized composition $P_A \| _B Q$ is shown in Fig. 4(c). The transitions labeled (b, e), (b, f), (a, e), (a, f) are as in the first case of Definition 1, those labeled (c, ϵ), (b, ϵ) are as in the second case of the definition, that labeled (ϵ, g) is as in the third case of the definition, and that labeled ϵ is as in the fourth case of the definition.

Remark 2: Note that in the second clause of Definition 1, $M_Q^{-1} M_P(\sigma_p) \cap B = \emptyset$ if and only if $M_Q^{-1} M_P(\sigma_p) \cap M_Q^{-1} M_Q(B) = \emptyset$. A similar statement applies to the third clause. So it follows that

$$P_A \| _B Q = P_{M_P^{-1} M_P(A)} \| _{M_Q^{-1} M_Q(B)} Q \tag{1}$$

and hence there is no loss of generality in requiring that the masks be priority consistent. [If the masks are not priority consistent, then the priority sets can be redefined as $\hat{A} = M_P^{-1} M_P(A)$ and $\hat{B} := M_Q^{-1} M_Q(B)$ so that the MPSC remains unaltered but the masks become priority consistent.]

The "external" behavior of the MPSC of P and Q observed at the interface, called the *projection of* $P_A \| _B Q$ *on* Σ_I and denoted $(P_A \| _B Q) \uparrow \Sigma_I$, is obtained by replacing transitions of $P_A \| _B Q$ as follows:

1) $\forall (x, (\sigma_p, \sigma_q), x') \in X \times (\Sigma_P \times \Sigma_Q) \times X$, replace it by $(x, M_P(\sigma_p), x') = (x, M_Q(\sigma_q), x')$;
2) $\forall (x, (\sigma_p, \epsilon), x') \in X \times (\Sigma_P \times \{\epsilon\}) \times X$, replace it by $(x, M_P(\sigma_p), x')$;
3) $\forall (x, (\epsilon, \sigma_q), x') \in X \times (\{\epsilon\} \times \Sigma_Q) \times X$, replace it by $(x, M_Q(\sigma_q), x')$.

Thus the behavior observed at the interface consists of only the external events Σ_I.

Similarly, the behavior of the MPSC of P and Q *projected to the events of* P, denoted $(P_A \| _B Q) \uparrow \Sigma_P$, is obtained by "erasing" each Σ_Q-event label from all transitions of $P_A \| _B Q$ as follows:

$$\forall (x, (\sigma_p, \sigma_q), x') \in X \times (\overline{\Sigma}_P \times \overline{\Sigma}_Q) \times X, \quad \text{replace it by } (x, \sigma_p, x').$$

It is easily seen that the generated (respectively, marked) language of $(P_A \| _B Q) \uparrow \Sigma_P$ is contained in the generated (respectively, marked) language of P. Thus MPSC of P with Q restricts the behavior of P. This fact can be used to employ MPSC as a mechanism of control.

Example 3: The MPSC of the pumping station G and the controlling station S of Example 1 is shown in Fig. 5. In each state label of the composition, the first index denotes the state of the synchronizer, the second that of the pump 1, the third that of the pump 2, and the fourth that of the controlling station. Each transition is labeled by a pair of symbols—the first (respectively, second) is the event label of the corresponding transition in G (respectively, S).

In the initial state (1111), a_1 is enabled in R and P_1, a_2 in P_2, and a in S. Since a_1 is in the priority set of R and P_1 and is masked to a, which is in the priority set of S, a_1 synchronizes with a, causing a transition to the state (2212). On the other hand, since a_2 is in the priority set of R also, which refuses it in its initial state, a_2 is initially blocked in the composition. Similar analysis can be used to derive the entire NSM of the composed system as depicted in Fig. 5.

Next we show that the MPSC of two systems can alternatively be obtained by first "masking" the individual systems, next computing their PSC, and finally relabeling the transitions by "unmasking" them as described in the following algorithm.

Algorithm 1: Consider systems P and Q interacting at a common interface with events Σ_I, as shown in Fig. 1, their respective event priority sets A and B, and their respective priority consistent interface masks M_P and M_Q. Then their MPSC $P_A \| _B Q := (X, \Sigma, \delta, x^0, X^m)$ can be obtained as follows.

1) Compute the "masked" NSMs $M_P(P)$ and $M_Q(Q)$ and masked event priority sets $\overline{M}_P(A) := M_P(A) - \{\epsilon\} \subseteq \Sigma_I$ and $\overline{M}_Q(B) := M_Q(B) - \{\epsilon\} \subseteq \Sigma_I$.
2) Compute the PSC $M_P(P)_{\overline{M}_P(A)} \| _{\overline{M}_Q(B)} M_Q(Q)$.
3) Replace each transition in $M_P(P)_{\overline{M}_P(A)} \| _{\overline{M}_Q(B)} M_Q(Q)$ to obtain NSM R as follows:
   a) $\forall ((x_p, x_q), \sigma, (x_p', x_q')) \in X \times \Sigma_I \times X$, replace it by the set of transitions
      i) $((x_p, x_q), (\sigma_p, \sigma_q), (x_p', x_q')) \in X \times (\Sigma_P \times \Sigma_Q) \times X$ if $x_p' \in \delta_P(x_p, \sigma_p)$, $x_q' \in \delta_Q(x_q, \sigma_q)$, $M_P(\sigma_p) = M_Q(\sigma_q) = \sigma$;
      ii) $((x_p, x_q), (\sigma_p, \epsilon), (x_p', x_q')) \in X \times (\Sigma_P \times \{\epsilon\}) \times X$ if $x_p' \in \delta_P(x_p, \sigma_p)$, $x_q =$

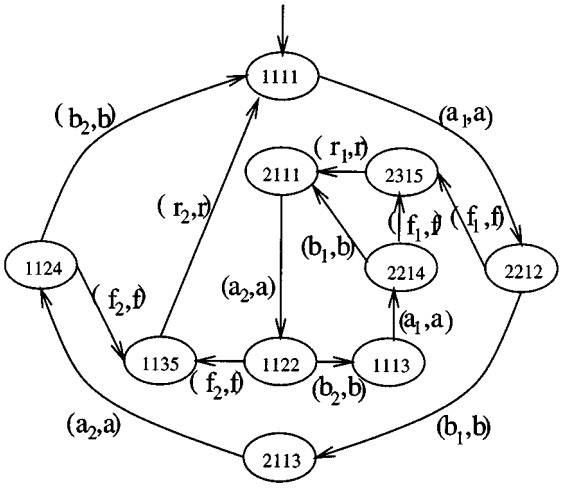

Fig. 5. MPSC of pumping and controlling stations.

$x'_q, \delta_Q(x_q, M_Q^{-1}(\sigma)), M_Q^{-1}(\sigma) \cap B = \emptyset, M_P(\sigma_p) = \sigma$;

iii) $((x_p, x_q), \sigma_q, (x'_p, x'_q)) \in X \times (\{\epsilon\} \times \Sigma_Q) \times X$ if $x_p = x'_p, \delta_P(x_p, M_P^{-1}(\sigma)), M_P^{-1}(\sigma) \cap A = \emptyset, x'_q \in \delta_Q(x_q, \sigma_q), M_Q(\sigma_q) = \sigma$

b) $\forall ((x_p, x_q), \epsilon, (x'_p, x'_q)) \in X \times \{\epsilon\} \times X$, add the set of transitions

i) $((x_p, x_q), (\sigma_p, \epsilon), (x'_p, x'_q)) \in X \times (\Sigma_P \times \{\epsilon\}) \times X$ if $x'_p \in \delta_P(x_p, \sigma_p), x_q = x'_q, M_P(\sigma_p) = \epsilon$;

ii) $((x_p, x_q), (\epsilon, \sigma_q), (x'_p, x'_q)) \in X \times (\{\epsilon\} \times \Sigma_Q) \times X$ if $x_p = x'_p, x'_q \in \delta_Q(x_q, \sigma_q), M_Q(\sigma_q) = \epsilon$.

Note that the complexity of Algorithm 1 is of the same order as the number of transitions in the MPSC of P and Q. [if P and Q have m and n states, respectively, then there are $O(m^2n^2)$ transitions in their MPSC.]

The following theorem proves the correctness of Algorithm 1.

Theorem 1: Let P, Q, A, B, M_P, M_Q, R be as in Algorithm 1. Then $P_A \| _B Q = R$.

Proof: Since the two NSMs $P_A \| _B Q$ and R have identical states $X := X_P \times X_Q$, identical events $\Sigma := (\Sigma_P \times \Sigma_q)$, identical initial state $x^0 := (x_P^0, x_Q^0)$, and identical final states $X^m := X_P^m \times X_Q^m$, we only need to show that they also have the identical set of transitions. We first show that each transition of R is also a transition of $P_A \| _B Q$.

Consider first a transition $((x_p, x_q), (\sigma_p, \sigma_q), (x'_p, x'_q)) \in X \times (\Sigma_P \times \Sigma_Q) \times X$ of R as in clause 3a-i) of Algorithm 1. Then, from this clause, $x'_p \in \delta_P(x_p, \sigma_p), x'_q \in \delta_Q(x_q, \sigma_q)$, and $M_P(\sigma_p) = M_Q(\sigma_q) \neq \epsilon$. So from the first clause in Definition 1, it follows that $((x_p, x_q), (\sigma_p, \sigma_q), (x'_p, x'_q))$ is a transition of $P_A \| _B Q$.

Next, consider a transition $((x_p, x_q), (\sigma_p, \epsilon), (x'_p, x_q)) \in X \times (\Sigma_P \times \{\epsilon\}) \times X$ of R as in clause 3a-ii) or 3b-i) of Algorithm 1. In the first case, from clause 3a-ii), we have $x'_p \in \delta_P(x_p, \sigma_p)$ and $\delta_Q(x_q, M_Q^{-1} M_P(\sigma_p)), M_Q^{-1} M_P(\sigma_p) \cap B = \emptyset$; whereas in the second case, from clause 3b-i), $x'_p \in \delta_P(x_p, \sigma_p)$ and $M_P(\sigma_p) = \epsilon$. So in either case, it follows from the second clause in Definition 1 that $((x_p, x_q), \sigma_p, (x'_p, x_q))$ is a transition of $P_A \| _B Q$. By symmetry, a transition $((x_p, x_q), (\epsilon, \sigma_q), (x_p, x'_q)) \in X \times (\{\epsilon\} \times \Sigma_Q) \times X$ of R is also a transition of $P_A \| _B Q$.

Finally, consider a transition $((x_p, x_q), \epsilon, (x'_p, x_q)) \in X \times \{\epsilon\} \times X$ of R as in clause 3b)i) of Algorithm 1. Then from this clause $x'_p \in \delta_P(x_p, \epsilon)$. So from the last clause of Definition 1, $((x_p, x_q), \epsilon, (x'_p, x_q))$ is also a transition of $P_A \| _B Q$. By symmetry, a transition $((x_p, x_q), \epsilon, (x_p, x'_q)) \in X \times \{\epsilon\} \times X$ of R is also a transition of $P_A \| _B Q$.

It remains to prove the converse that each transition of $P_A \| _B Q$ is also a transition of R. Consider first a transition $((x_p, x_q), (\sigma_p, \sigma_q), (x'_p, x'_q)) \in X \times (\Sigma_P \times \Sigma_Q) \times X$ of $P_A \| _B Q$ as in the first clause of Definition 1. Then from this clause $x'_p \in \delta_P(x_p, \sigma_p)$, $x'_q \in \delta_Q(x_q, \sigma_q)$ and $M_P(\sigma_p) = M_Q(\sigma_q) \neq \epsilon$. Define $\sigma := M_P(\sigma_p) = M_Q(\sigma_q)$. Then $x'_p \in \delta_{M_P(P)}(x_p, \sigma)$ and $x'_q \in \delta_{M_Q(Q)}(x_q, \sigma)$. So it follows from the first clause in the definition of PSC that $((x_p, x_q), \sigma, (x'_p, x'_q))$ is a transition of $M_P(P)_{\overline{M}_P(A)} \| _{\overline{M}_Q(B)} M_Q(Q)$. So by applying clause 3a=i) of Algorithm 1, we conclude that the transition $((x_p, x_q), (\sigma_p, \sigma_q), (x'_p, x'_q))$ is also a transition of R.

Next, consider a transition $((x_p, x_q), (\sigma_p, \epsilon), (x'_p, x_q)) \in X \times (\Sigma_P \times \{\epsilon\}) \times X$ of $P_A \| _B Q$ as in the second clause of Definition 1. Then, from this clause, we have $x'_p \in \delta_P(x_p, \sigma_p)$ and $\delta_Q(x_q, M_Q^{-1} M_P(\sigma_p)), M_Q^{-1} M_P(\sigma_p)) \cap B = \emptyset$, or $M_P(\sigma_p) = \epsilon$. We consider the two cases separately. In the first case, when $M_P(\sigma_p) \neq \epsilon$, define $\sigma := M_P(\sigma_p)$. Then we have $x'_p \in \delta_{M_P(P)}(x_p, \sigma)$ and $\delta_{M_Q(Q)}(x_q, \sigma) = \delta_Q(x_q, M_Q^{-1}(\sigma)) = \emptyset$. Also, since $M_Q^{-1}(\sigma) \cap B = \emptyset$, $\sigma \notin M_Q(B) \supset \overline{M}_Q(B)$. So it follows from the second clause in the definition of PSC that $((x_p, x_q), \sigma, (x'_p, x_q))$ is a transition of $M_P(P)_{\overline{M}_P(A)} \| _{\overline{M}_Q(B)} M_Q(Q)$. So, by applying clause 3a-ii) of Algorithm 1, we conclude that the transition $((x_p, x_q), (\sigma_p, \epsilon), (x'_p, x_q))$ is a transition of R. On the other hand, in the second case, when $M_P(\sigma_p) = \epsilon$, from the last clause in the definition of PSC, we have that $((x_p, x_q), M_P(\sigma_p), (x'_p, x_q))$ is a transition of $M_P(P)_{\overline{M}_P(A)} \| _{\overline{M}_Q(B)} M_Q(Q)$. So, by applying clause 3b-i) of Algorithm 1, we conclude that $((x_p, x_q), (\sigma_p, \epsilon), (x'_p, x_q))$ is a transition of R. By symmetry, we have that a transition $((x_p, x_q), (\epsilon, \sigma_q), (x_p, x'_q)) \in X \times (\{\epsilon\} \times \Sigma_Q) \times X$ of $P_A \| _B Q$ is also a transition of R.

Finally, consider a transition $((x_p, x_q), \epsilon, (x'_p, x_q)) \in X \times \{\epsilon\} \times X$ of $P_A \| _B Q$. Then from the last clause in Definition 1, $x'_p \in \delta_P(x_p, \epsilon)$, which implies $x'_p \in \delta_{M_P(P)}(x_p, \epsilon)$. Hence from the last clause in the definition of PSC, we have that $((x_p, x_q), \epsilon, (x'_p, x_q))$ is a transition of $M_P(P)_{\overline{M}_P(A)} \| _{\overline{M}_Q(B)} M_Q(Q)$. So, by applying clause 3b-i of Algorithm 1, we conclude that $((x_p, x_q), \epsilon, (x'_p, x_q))$ is a transition of R. By symmetry, we have that a transition $((x_p, x_q), \epsilon, (x_p, x'_q)) \in X \times \{\epsilon\} \times X$ of $P_A \| _B Q$ is also a transition of R. This completes the proof. ■

Theorem 1 establishes a link between MPSC and PSC. By definition, MPSC generalizes PSC; conversely, it follows from

Theorem 1 that MPSC can be computed using the definition of PSC by applying Algorithm 1, which requires a "premasking" and a "postunmasking" operation. This fact is not explored in this paper any further. A consequence of Theorem 1, however, is that $P_A \| _B Q \uparrow \Sigma_I = M_P(P)_{\overline{M}_{P(A)}} \| _{\overline{M}_{Q(B)}} M_Q(Q)$, which can be used to derive results regarding the control of the behavior observed at the interface.

We next investigate the associativity of MPSC. It is known from [5, Theorem 13.4] (see also [10], where a detailed proof was given) that PSC is associative, i.e., given NSMs P, Q, R that evolve over a common event set Σ along with their respective priority sets A, B, $C \subseteq \Sigma$, the following holds:

$$(P_A \| _B Q)\, _{A \cup B} \| _C R = P_A \| _{B \cup C} (Q_B \| _C R).$$

Thus associativity lets us compute the composition of several systems by computing it two at a time.

We show that the property of associativity also holds for MPSC of systems interacting at a common interface. Consider, for example, three NSMs P, Q, R with respective event priority set $A \subseteq \Sigma_P$, $B \subseteq \Sigma_Q$, $C \subseteq \Sigma_R$ interacting at a common interface as shown in Fig. 6. The interface masks of P, Q, R are given by M_P, M_Q, M_R, respectively.

In order to demonstrate associativity of MPSC, we show that MPSC of P, Q, R can be computed by first computing the MPSC of any of the two systems and next composing this with the third system. Two ways of achieving this are shown in Fig. 7. In Fig. 7(a), composition of P, Q is first obtained, and next this is composed with R, whereas in Fig. 7(b), composition of Q, R is first obtained, which is then composed with P. We use the mask function pair (M_P, M_Q) to denote the mask function of the composition $P_A \| _B Q$, the first (respectively, second) component of which applies to transitions with an event label in Σ_P (respectively, Σ_Q). Note that whenever a transition in $P_A \| _B Q$ is labeled by an event pair $(\sigma_p, \sigma_q) \in \Sigma_P \times \Sigma_Q$, we have $M_P(\sigma_p) = M_Q(\sigma_q) \neq \epsilon$, i.e., both the events are masked to the same interface event, which is observable at the interface. So there is no confusion of event synchronization when the composed system $P_A \| _B Q$ interacts with R.

Theorem 2: Consider systems P, Q, R interacting at a common interface with events Σ_I, their respective event priority sets A, B, C, and their respective priority consistent interface masks M_P, M_Q, M_R. Then

$$[P_A \| _B Q]\, _{(A \ominus B)} \| _C R = P_A \| _{(B \ominus C)} [Q_B \| _C R].$$

Proof: Let $X := X_P \times X_Q \times X_R$, $\Sigma := \overline{\Sigma}_P \times \overline{\Sigma}_Q \times \overline{\Sigma}_R$ denote the state set and the event set of the composition. Then the proof follows from the fact that the composition of the three systems in either of the two configurations of Fig. 7(a) and (b) has identical transition function and is given by the equation shown at the bottom of the next page.

Finally, the event priority set of the composition of the three systems in either of the two configurations of Fig. 7(a) and (b) equals

$$\begin{aligned}[A \oplus B] \oplus C &= A \oplus [B \oplus C] \\ &= [A \times \overline{\Sigma}_Q \times \overline{\Sigma}_R] \cup [\overline{\Sigma}_P \times B \times \overline{\Sigma}_R] \\ &\quad \cup [\overline{\Sigma}_P \times \overline{\Sigma}_Q \times C].\end{aligned}$$

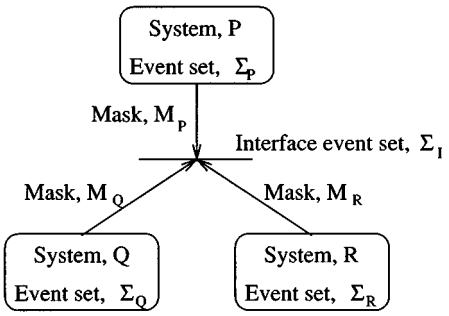

Fig. 6. P, Q, R interacting at a common interface.

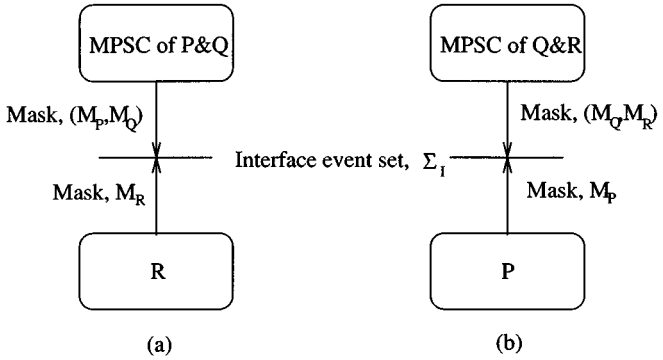

Fig. 7. Two ways of associating composition of P, Q, R.

■

Remark 3: The transition function of the three systems given in the proof of Theorem 2 also defines the transition function of the composition of the systems shown in Fig. 6.

IV. Supervisory Control

In this section, we extend the supervisory control theory to the present setting where a supervisor controls a discrete event plant by interacting with it at a common interface via masked prioritized synchronization similar to that shown in Fig. 1 under the restriction that the set of driven events is empty. The plant is modeled by an NSM $G := (X_G, \Sigma_G, \delta_G, x_G^0, X_G^m)$ having event priority set $A \subseteq \Sigma_G$ and priority consistent interface mask $M_G : \Sigma_G \to \Sigma_I$, where Σ_I is the set of interface events. Since the supervisor exercises its control based on its observation of the *event traces* generated by the plant, it is modeled by a *deterministic state machine* $S := (X_S, \Sigma_S, \delta_S, x_S^0, X_S^m)$. The event priority set of the supervisor $B \subseteq \Sigma_S$ and its interface mask M_S: $\Sigma_S \to \Sigma_I$ are given.

It is natural to require that each event of the plant be either in the priority set of the plant or identified via the interface masks with some event that is in the priority set of the supervisor, i.e., $\Sigma_G - A \subseteq M_G^{-1}(\overline{M}_S(B))$, where $\overline{M}_S(B) := M_S(B) - \{\epsilon\}$. In other words, the set of plant events is the union of controllable events $A \cap M_G^{-1}(\overline{M}_S(B))$, the uncontrollable events $A - M_G^{-1}(\overline{M}_S(B)) := \Sigma_u$, and the driven events $M_G^{-1}(\overline{M}_S(B)) - A = \Sigma_G - A$. This requirement is consistent with the corresponding requirement in the setting of PSC that each event of the plant is either in its own priority set or in the priority set of the supervisor, and rules out the possibility that a nonpriority event of the plant is identified with no priority

event of the supervisor or is masked to ϵ. The above requirement together with the assumption that there are no driven events implies that $\Sigma_G = A$.

Table I summarizes the controllability and observability property of each event of the plant that results from the event priorities and interface masks of the plant and the supervisor. Similarly, Table II summarizes the properties of the events of the supervisor. Note that while the supervisor can execute an observable driven event to issue a command, it can execute an unobservable driven event to change its control policy.

The control specification is given by a language $K \subseteq \Sigma_G^*$ describing the permitted event sequences of the controlled plant $(G_A \| _B S) \uparrow \Sigma_G$. The control task is to design a deterministic supervisor S such that the controlled plant behavior satisfies the specification under the restriction that no driven events are present.

Example 4: Consider the pumping station G and interface M_G of Example 1 as the uncontrolled plant. The control task is to design a controlling station S that restricts the plant to operate so that *at least one pump is idle at any given time*. This desired specification is shown in Fig. 8. The event priority set and the interface mask function of a controlling station to be designed enforcing such a specification are as given in Example 3, except we assume that the repair events are also controllable. (Recall our assumption for this section that the set of driven events is empty.)

We are interested in obtaining a necessary and sufficient condition for the existence of a supervisor for the supervisory control problem described above. Under the assumption of no driven events, we show that $(L(G), \Sigma_u)$-controllability together with $(L(G), M_G)$-normality of the desired behavior K serves as a necessary and sufficient condition for the existence of a supervisor. We first prove two preliminary results about controllability and normality.

The first lemma provides an alternate characterization of (Σ_G^*, M_G)-normality.

$$\forall\, x = (x_p, x_q, x_r) \in X_P \times X_Q \times X_R, \quad \sigma = (\sigma_p, \sigma_q, \sigma_r) \in \Sigma_P \times \Sigma_Q \times \Sigma_R\text{:}$$

$$\delta(x, \sigma) := \begin{cases} \delta_P(x_p, \sigma_p) \times \delta_Q(x_q, \sigma_q) \times \delta_R(x_r, \sigma_r), & \text{if } M_P(\sigma_p) = M_Q(\sigma_q) = M_R(\sigma_r) \neq \epsilon, \\ & \delta_P(x_p, \sigma_p), \delta_Q(x_q, \sigma_q), \delta_R(x_r, \sigma_r) \neq \emptyset \\ \emptyset, & \text{otherwise} \end{cases}$$

$$\delta(x, (\sigma_p, \sigma_q, \epsilon)) := \begin{cases} \delta_P(x_p, \sigma_p) \times \delta_Q(x_q, \sigma_q) \times \{x_r\}, & \text{if } \overline{\sigma} := M_P(\sigma_p) = M_Q(\sigma_q) \neq \epsilon, \\ & \delta_P(x_p, \sigma_p), \delta_Q(x_q, \sigma_q) \neq \emptyset, \\ & \delta_R(x_r, M_R^{-1}(\overline{\sigma})), M_R^{-1}(\overline{\sigma}) \cap C = \emptyset \\ \emptyset, & \text{otherwise} \end{cases}$$

$$\delta(x, (\epsilon, \sigma_q, \sigma_r)) := \begin{cases} \{x_p\} \times \delta_Q(x_q, \sigma_q) \times \delta_R(x_r, \sigma_r), & \text{if } \overline{\sigma} := M_Q(\sigma_q) = M_R(\sigma_r) \neq \epsilon, \\ & \delta_Q(x_q, \sigma_q), \delta_R(x_r, \sigma_r) \neq \emptyset, \\ & \delta_P(x_p, M_P^{-1}(\overline{\sigma})), M_P^{-1}(\overline{\sigma}) \cap A = \emptyset \\ \emptyset, & \text{otherwise} \end{cases}$$

$$\delta(x, (\sigma_p, \epsilon, \sigma_r)) := \begin{cases} \delta_P(x_p, \sigma_p) \times \{x_q\} \times \delta_R(x_r, \sigma_r), & \text{if } \overline{\sigma} := M_P(\sigma_p) = M_R(\sigma_r) \neq \epsilon, \\ & \delta_P(x_p, \sigma_p), \delta_R(x_r, \sigma_r) \neq \emptyset, \\ & \delta_Q\left(x_q, M_Q^{-1}(\overline{\sigma})\right), M_Q^{-1}(\overline{\sigma}) \cap B = \emptyset \\ \emptyset, & \text{otherwise} \end{cases}$$

$$\delta(x, (\sigma_p, \epsilon, \epsilon)) := \begin{cases} \delta_P(x_p, \sigma_p) \times \{x_q\} \times \{x_r\}, & \text{if } \delta_P(x_p, \sigma_p) \neq \emptyset, [[M_P(\sigma_p) = \epsilon] \\ & \vee [\delta_Q(x_q, M_Q^{-1} M_P(\sigma_p)), M_Q^{-1} M_P(\sigma_p) \cap B, \\ & \delta_R(x_r, M_R^{-1} M_P(\sigma_p)), M_R^{-1} M_P(\sigma_p) \cap C = \emptyset]] \\ \emptyset, & \text{otherwise} \end{cases}$$

$$\delta(x, (\epsilon, \sigma_q, \epsilon)) := \begin{cases} \{x_p\} \times \delta_Q(x_q, \sigma_q) \times \{x_r\}, & \text{if } \delta_Q(x_q, \sigma_q) \neq \emptyset, [[M_Q(\sigma_q) = \epsilon] \\ & \vee [\delta_P(x_p, M_P^{-1} M_Q(\sigma_q)), M_P^{-1} M_Q(\sigma_q) \cap A, \\ & \delta_R(x_r, M_R^{-1} M_Q(\sigma_q)), M_R^{-1} M_Q(\sigma_q) \cap C = \emptyset]] \\ \emptyset, & \text{otherwise} \end{cases}$$

$$\delta(x, (\epsilon, \epsilon, \sigma_r)) := \begin{cases} \{x_p\} \times \{x_q\} \times \delta_R(x_r, \sigma_r), & \text{if } \delta_R(x_r, \sigma_r) \neq \emptyset, [[M_R(\sigma_r) = \epsilon] \\ & \vee [\delta_P(x_p, M_P^{-1} M_R(\sigma_r)), M_P^{-1} M_R(\sigma_r) \cap A, \\ & \delta_Q(x_q, M_Q^{-1} M_R(\sigma_r)), M_Q^{-1} M_R(\sigma_r) \cap B = \emptyset]] \\ \emptyset, & \text{otherwise} \end{cases}$$

$$\begin{aligned} \delta(x, \epsilon) := & [\delta_P(x_p, \epsilon) \times \{x_q\} \times \{x_r\}] \cup [\{x_p\} \times \delta_Q(x_q, \epsilon) \times \{x_r\}] \\ & \cup [\{x_p\} \times \{x_q\} \times \delta_R(x_r, \epsilon)] \end{aligned}$$

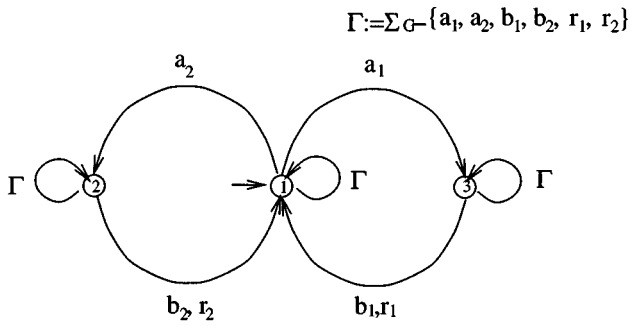

Fig. 8. Design specification for the pumping station.

TABLE I
CONTROLLABILITY AND OBSERVABILITY PROPERTY OF PLANT EVENTS

plant event	identified with supervisor event	event type
priority	priority	controllable & observable
priority	non-priority	uncontrollable & observable
priority	no event (epsilon)	uncontrollable & unobservable
non-priority	priority	driven & observable
non-priority	non-priority	non-existent
non-priority	no event (epsilon)	non-existent

TABLE II
CONTROLLABILITY AND OBSERVABILITY PROPERTY OF SUPERVISOR EVENTS

supervisor event	identified with plant event	event type
priority	priority	controllable & observable
priority	non-priority	driven & observable
priority	no event (epsilon)	driven & unobservable
non-priority	priority	uncontrollable & observable
non-priority	non-priority	non-existent
non-priority	no event (epsilon)	non-existent

Lemma 1: Consider $K \subseteq \Sigma_G^*$ and a mask M_G. Then the following are equivalent:

1) K is (Σ_G^*, M_G)-normal;
2) $\forall\, s, s', t \in \Sigma_G^* : [st \in pr(K), M_G(s) = M_G(s')] \Rightarrow [s't \in pr(K)]$;
3) $\forall\, u, v \in \Sigma_G^*, \sigma, \sigma' \in (\Sigma_G \cup \{\epsilon\}) : [u\sigma v \in pr(K), M_G(\sigma) = M_G(\sigma')] \Rightarrow [u\sigma' v \in pr(K)]$.

Proof: We begin by showing the equivalence of the first two assertions. To see that the first assertion implies the second, it suffices to note that $M_G(st) = M_G(s't)$; so from (Σ_G^*, M_G)-normality of K, it follows that $s't \in pr(K)$. To see the converse, pick $s \in pr(K)$, $s' \in \Sigma_G^*$ such that $M_G(s) = M_G(s')$. Then by setting $t := \epsilon$ in the second assertion, it follows that $s' \in pr(K)$.

Next we prove the equivalence of the last two assertions. To see that the second assertion implies the third, simply set $s := u\sigma$, $s' := u\sigma'$, $t := v$. Then from the hypothesis of the third assertion, $M_G(s) = M_G(s')$. So from the second assertion, $s't = u\sigma' v \in pr(K)$. To prove the converse, we fix t and proceed by induction on $|s|+|s'|$. For the base step, let $|s|+|s'| = 1$, and without loss of generality let $s' = \epsilon$. Set $u := \epsilon$, $v := t$, $\sigma := s$, $\sigma' := s'$. From the hypothesis of the second assertion $u\sigma v = st \in pr(K)$ and $M_G(\sigma) = M_G(s) = M_G(s') = M_G(\sigma')$. So from the third assertion, $u\sigma' v = s't \in pr(K)$. Next, for the induction step, let $s := \overline{s}\sigma$, $s' := \overline{s}'\sigma'$, where $\sigma, \sigma' \in \Sigma_G$. Then we have three possible cases:

1) $M_G(\overline{s}) = M_G(\overline{s}')$ and $M_G(\sigma) = M_G(\sigma')$;
2) $M_G(s) = M_G(\overline{s}')$ and $M_G(\sigma') = \epsilon$;
3) $M_G(\overline{s}) = M_G(s')$ and $M_G(\sigma) = \epsilon$.

We only analyze the first case; the others can be analyzed similarly. In the first case, since $M_G(\overline{s}) = M_G(\overline{s}')$, and since from the hypothesis of the second assertion $\overline{s}\sigma t = st \in pr(K)$, it follows from the induction hypothesis that $\overline{s}'\sigma t \in pr(K)$. Set $u := \overline{s}'$, $v := t$. Then since $M_G(\sigma) = M_G(\sigma')$, and since $u\sigma v = \overline{s}'\sigma t \in pr(K)$, it follows from the third assertion that $u\sigma' v = \overline{s}'\sigma' t = s't \in pr(K)$, completing the induction step. ■

Remark 4: Let S be a trim [6] deterministic state machine that accepts a (Σ_G^*, M_G)-normal language $K \subseteq \Sigma_G^*$ so that $L(S) = pr(K)$. Then it follows from the third assertion of Lemma 1 that for any $u \in L(S)$ and indistinguishable events $\sigma, \sigma' \in \Sigma_G$, $u\sigma$ and $u\sigma'$ are Nerode-equivalent. Hence S can be chosen such that transitions on a pair of indistinguishable events (under M_G) from any state whenever defined have the same successor state, and transitions on unobservable events (under M_G) from any state whenever defined are self-loops. We exploit this fact in constructing a supervisor for our supervisory control problem.

Let K^{Σ_u, M_G} denote the infimal prefix-closed (Σ_G^*, Σ_u)-controllable and (Σ_G^*, M_G)-normal superlanguage of K. Then the next lemma states that $K^{\Sigma_u, M_G} \cap L(G)$ equals $pr(K)$ if and only if K is $(L(G), \Sigma_u)$-controllable and $(L(G), M_G)$-normal.

Lemma 2: Consider plant G, language $K \subseteq L(G)$, set of uncontrollable events $\Sigma_u \subseteq \Sigma_G$, and interface mask M_G. Then $K^{\Sigma_u, M_G} \cap L(G) = pr(K)$ if and only if K is $(L(G), \Sigma_u)$-controllable and $(L(G), M_G)$-normal.

Proof: We first prove the necessity. To see the $(L(G), \Sigma_u)$-controllability of K, pick $s \in pr(K)$ and $\sigma \in \Sigma_u$ such that $s\sigma \in L(G)$. Then from (Σ_G^*, Σ_u)-controllability of K^{Σ_u, M_G}, $s\sigma \in K^{\Sigma_u, M_G}$. This implies $s\sigma \in K^{\Sigma_u, M_G} \cap L(G) = pr(K)$ as desired. Similarly to see the $(L(G), M_G)$-normality of K, pick $s \in pr(K)$ and $t \in L(G)$ such that $M_G(s) = M_G(t)$. Then from the (Σ_G^*, M_G)-normality of K^{Σ_u, M_G}, $t \in K^{\Sigma_u, M_G}$. This implies $t \in K^{\Sigma_u, M_G} \cap L(G) = pr(K)$ as desired.

Next to prove sufficiency it is enough to show that $K^{\Sigma_u, M_G} \cap L(G) \subseteq pr(K)$, since the reverse inclusion holds trivially. Clearly, this is true for $K = \emptyset$. So we assume $K \neq \emptyset$. We prove the desired inclusion by induction on the length of traces. Since $\epsilon \in pr(K)$ (recall $K \neq \emptyset$), it suffices to show that for any $s \in \Sigma_G^*$, $\sigma \in \Sigma_G$ such that $s\sigma \in K^{\Sigma_u, M_G} \cap L(G)$, $s\sigma \in pr(K)$. By the induction hypothesis, $s \in pr(K)$, and since $s\sigma \in K^{\Sigma_u, M_G}$, it follows from its definition that either 1) $\sigma \in \Sigma_u$ or 2) there exists $t \in pr(K)$ such that $M_G(t) = M_G(s\sigma)$. [Otherwise, the proper sublanguage of K^{Σ_u, M_G}, $K^{\Sigma_u, M_G} - \{s\sigma\}\Sigma_G^*$, obtained by disabling σ after s is a prefix-closed, (Σ_G^*, Σ_u)-controllable, and (Σ_G^*, M_G)-normal superlanguage of K, a contradiction.] Since we also have $s\sigma \in L(G)$, $(L(G), \Sigma_u)$-controllability

of K in case 1) [respectively, $(L(G), M_G)$-normality of K in case 2)] implies $s\sigma \in pr(K)$ as desired. ■

We next state the result for the existence of the supervisor.

Theorem 3: Consider a plant G with priority set $A = \Sigma_G$ and priority consistent interface mask $M_G : \Sigma_G \to \Sigma_I$. Let $K \subseteq L(G)$ be a prefix-closed nonempty desired language, Σ_S be the event set of the supervisor, $B \subseteq \Sigma_S$ its event priority set, and $M_S : \Sigma_S \to \Sigma_I$ its priority consistent interface mask. Then there exists a deterministic supervisor S such that $L((G_A \| _B S) \uparrow \Sigma_G) = K$ if and only if K is $(L(G), \Sigma_u)$-controllable and $(L(G), M_G)$-normal, where $\Sigma_u := A - M_G^{-1}(M_S(B) - \{\epsilon\})$.

Proof: We first prove the necessity. Let $G_A \| _B S := (X, \Sigma, \delta, x^0, X^m)$ and $(G_A \| _B S) \uparrow \Sigma_G := (X, \Sigma_G, \delta^{\uparrow}, x^0, X^m)$. To see $(L(G), \Sigma_u)$-controllability and $(L(G), M_G)$-normality of K, pick $u \in K$, $\sigma \in \Sigma_u$, $v \in L(G)$ such that $u\sigma \in L(G)$ and $M_G(u) = M_G(v)$. Then since $\Sigma_G = A$, G must participate in the execution of every event in the trace $u\sigma$. This together with the fact that $u \in K = L((G_A \| _B S) \uparrow \Sigma_G)$ and $u\sigma \in L(G)$ implies that there exists $x_g \in X_G$, $x_s \in X_S$ such that $(x_g, x_s) \in \delta^{\uparrow}(x^0, u)$ and $\delta_G(x_g, \sigma) \neq \emptyset$. Then since $\sigma \in \Sigma_u$ and every event in $M_S^{-1}\overline{M}_G(\Sigma_u)$ is defined at every state of S, it follows that either $\delta((x_g, x_s), (\sigma, M_S^{-1}\overline{M}_G(\sigma))) \neq \emptyset$ [which is the case when $M_G(\sigma) \neq \epsilon$] or $\delta((x_g, x_s), (\sigma, \epsilon)) \neq \emptyset$. In either case, we have $\delta^{\uparrow}((x_g, x_s), \sigma) \neq \emptyset$, which implies $\delta^{\uparrow}(x^0, u\sigma) \neq \emptyset$ proving that $u\sigma \in L((G_A \| _B S) \uparrow \Sigma_G) = K$. Next since $v \in L(G)$, there exists $x_g' \in X_G$ such that $x_g' \in \delta_G(x_G^0, v)$. Moreover, since $M_G(v) = M_G(u)$, it follows from the determinism of S that the same set of states are reached by "tracking" the two traces u and v in S, including the state x_s. This implies $(x_g', x_s) \in \delta^{\uparrow}(x^0, v)$. Thus $v \in L((G_A \| _B S) \uparrow \Sigma_G) = K$ as desired.

Next, to prove the sufficiency, construct a deterministic supervisor as follows. First construct a deterministic trim state machine

$$\overline{S} := (X_S, \Sigma_G, \delta_{\overline{S}}, x_S^0, X_S)$$

that generates K^{Σ_u, M_G}, the infimal prefix-closed (Σ_G^*, Σ_u)-controllable and (Σ_G^*, M_G)-normal superlanguage of K. Then, as explained in Remark 4, since K^{Σ_u, M_G} is $(\Sigma_G^*, L(G))$-normal, $\overline{S}$ can be chosen so that transitions on indistinguishable events on any state whenever defined have the same successor state, and transitions on unobservable events on any state whenever defined are self-loops.

Next, obtain a supervisor

$$S := (X_S, \Sigma_S, \delta_S, x_S^0, X_S)$$

by modifying each transition $(x, \sigma_g, x') \in X_S \times \Sigma_G \times X_S$ of $\overline{S}$ as follows.

1) If $M_G(\sigma_g) \neq \epsilon$, then replace it by a transition $(x, \sigma_s, x') \in X_S \times \Sigma_S \times X_S$ such that $M_S(\sigma_s) = M_G(\sigma_g)$.
2) If $M_G(\sigma_g) = \epsilon$, then delete this transition.

Then it is easy to see that S is deterministic. It remains to show that $L((G_A \| _B S) \uparrow \Sigma_G) = K$. Since K is $(L(G), \Sigma_u)$-controllable and $(L(G), M_G)$-normal, from Lemma 2 it suffices to show

$$L((G_A \| _B S) \uparrow \Sigma_G) = L(G) \cap K^{\Sigma_u, M_G}. \tag{2}$$

As before, let $G_A \| _B S := (X, \Sigma, \delta, x^0, X^m)$ and $(G_A \| _B S) \uparrow \Sigma_G := (X, \Sigma_G, \delta^{\uparrow}, x^0, X^m)$.

We first prove the forward containment in (2). Since $L((G_A \| _B S) \uparrow \Sigma_G) \subseteq L(G)$, it suffices to show that $L((G_A \| _B S) \uparrow \Sigma_G) \subseteq K^{\Sigma_u, M_G}$. First, consider a trace s that $G_A \| _B S$ generates. Then since G has priority over each event (the set of driven events is empty), it participates in each transition of the trace. On the other hand, since S has priority over controllable events and has transition defined on every observable uncontrollable event at each state, it participates in the execution of each observable event of the trace. We show by induction on length of $s \in L((G_A \| _B S) \uparrow \Sigma_G)$ that $s \in K^{\Sigma_u, M_G}$. Clearly, this holds for length zero since K^{Σ_u, M_G} is nonempty and prefix-closed, which establishes the base step. For the induction hypothesis, let $s = \overline{s}\sigma_g$. If σ_g is uncontrollable or unobservable, then by definition of K^{Σ_u, M_G} and induction hypothesis [which implies $\overline{s} \in K^{\Sigma_u, M_G} = L(\overline{S})$], it follows that $s \in K^{\Sigma_u, M_G}$. On the other hand, if σ_g is controllable and observable, then a corresponding event occurs synchronously in S. This means that if x is the state reached by the execution of $\overline{s}$ in $\overline{S}$, then σ_g is defined at x, which of course implies that $s = \overline{s}\sigma_g \in K^{\Sigma_u, M_G}$ [recall that $L(\overline{S}) = K^{\Sigma_u, M_G}$]. This proves the induction hypothesis.

To complete the proof of the sufficiency part, we next need to show the reverse containment in (2). Pick $u_g \in K^{\Sigma_u, M_G} \cap L(G) = L(\overline{S}) \cap L(G)$. Then there exists unique state $x_s \in X_S$ and a state $x_g \in X_G$ such that $\delta_{\overline{S}}(x_S^0, u_g) = x_s$ and $x_g \in \delta_G(x_G^0, u_g)$. Let $u_s \in L(S)$ be the trace corresponding to u_g obtained by relabeling/deleting the various transitions of u_g in $\overline{S}$. Then $M_S(u_s) = M_G(u_g)$ and $\delta_S(x_S^0, u_s) = x_s$. Then it is easy to see that $(x_g, x_s) \in \delta^{\uparrow}(x^0, u_g)$, which implies $u_g \in L((G_A \| _B S) \uparrow \Sigma_G)$ as desired. ■

The supervisor constructed in the proof of Theorem 3 is based upon a generator of K^{Σ_u, Σ_G}. The following lemma provides a modular way of doing that. We let K^{Σ_u}, K^{M_G} denote the infimal prefix-closed (Σ_G^*, Σ_u)-controllable and the infimal prefix-closed (Σ_G^*, M_G)-normal superlanguage of K, respectively. It is known that $K^{\Sigma_u} = pr(K)\Sigma_u^*$ and $K^{M_G} = M_G^{-1}M_G(pr(K))$.

Lemma 3: Consider $K \subseteq \Sigma_G^*$, a set of uncontrollable events $\Sigma_u \subseteq \Sigma_G$, and a mask M_G over Σ_G. Then $K^{\Sigma_u, M_G} = (K^{\Sigma_u})^{M_G}$.

Proof: The backward containment can be shown as follows. By definition, we have $K^{\Sigma_u} \subseteq K^{\Sigma_u, M_G}$, which implies $(K^{\Sigma_u})^{M_G} \subseteq (K^{\Sigma_u, M_G})^{M_G} = K^{\Sigma_u, M_G}$, where the last equality follows from the fact that K^{Σ_u, M_G} is prefix-closed and (Σ_G^*, M_G)-normal. For the forward containment, it suffices to show that $(K^{\Sigma_u})^{M_G}$ is a prefix-closed (Σ_G^*, Σ_u)-controllable and (Σ_G^*, M_G)-normal superlanguage of K since K^{Σ_u, M_G} is the infimal such language. By definition, $(K^{\Sigma_u})^{M_G}$ is a prefix-closed and (Σ_G^*, M_G)-normal superlanguage of K, and it remains to show that it is also (Σ_G^*, Σ_u)-controllable. To see this, pick $s \in (K^{\Sigma_u})^{M_G}$ and $\sigma \in \Sigma_u$. Then there

exists $t \in K^{\Sigma_u}$ such that $M_G(t) = M_G(s)$. By prefix-closure and (Σ_G^*, Σ_u)-controllability of K^{Σ_u}, it follows that $t\sigma \in K^{\Sigma_u}$. Finally, since $M_G(t\sigma) = M_G(s\sigma)$, it follows that $s\sigma \in (K^{\Sigma_u})^{M_G}$ as desired. ■

Remark 5: Lemma 3 provides a modular way of constructing K^{Σ_u, M_G}. Given a trim acceptor for K [which generates $pr(K)$], we first obtain a generator for $K^{\Sigma_u} = pr(K)\Sigma_u^*$ by augmenting the state space of the acceptor of K with a "dump" state, and then by adding transitions from each state to the dump state on those uncontrollable events that are undefined at that state. Next, we obtain the generator for $(K^{\Sigma_u})^{M_G} = M_G^{-1} M_G(K^{\Sigma_u})$ by replacing the event label σ of any transition in the generator of K^{Σ_u} by event labels in the set $M_G^{-1}(\sigma)$, and also adding self-loops on all unobservable events at each state of the generator of K^{Σ_u}.

Example 5: We now return to Example 4, where a specification for the pumping station G of Example 1 was formulated, with a slight modification that the events r_1 and r_2 are controllable events (instead of being the driven events) so that we can apply the results of Theorem 3 (recall that Theorem 3 requires the restriction that the set of driven events be empty). This specification, shown in Fig. 8, when intersected with the generated language of the pumping station imposes the language $K \subseteq L(G)$, as shown in Fig. 9. In this figure, each state has four components: the first component denotes the state of the synchronizer (Fig. 3), the second that of pump 1 (Fig. 3), the third that of pump 2 (Fig. 3), and the last that of the specification (Fig. 8).

Since K is prefix-closed and nonempty, from Theorem 3 there exists a deterministic supervisor S such that $L((G_A \| _B S) \uparrow \Sigma_G) = K$ if and only if K is $(L(G), \Sigma_u)$-controllable and $(L(G), M_G)$-normal. In this case, $\Sigma_u = \{f_1, f_2\}$ and M_G identifies a_is to a, b_is to b, f_is to f, and r_is to r. We use Lemma 2 to verify $(L(G), \Sigma_u)$-controllability and $(L(G), M_G)$-normality of K. The generator for K^{Σ_u, M_G} is shown in Fig. 10(a). Then it is easy to see that the synchronous composition of this with G yields the same state machine as the generator for K shown in Fig. 9. This establishes that K is $(L(G), \Sigma_u)$-controllable and $(L(G), M_G)$-normal.

Using the procedure described in the sufficiency part of the proof of Theorem 3, we arrive at the supervisor shown in Fig. 10(b) that enforces K as the projected behavior on the event set Σ_G of the composed system $G_A \| _B S$.

Remark 6: Theorem 3 provides a necessary and sufficient condition for the existence of a deterministic supervisor S with event priority set B and interface mask M_S for a given plant G with event priority set $A = \Sigma_G$ and interface mask M_G so that the projected behavior on the plant events of the composed system equals a given specification language K, i.e., $L((G_A \| _B S) \uparrow \Sigma_G) = K$, in terms of the familiar conditions of controllability and normality. The existing tests for controllability and normality, which are of polynomial complexity, can thus be applied to verify the existence of a supervisor (see, for example, [9, Sections 3.2.3, 4.2.3]).

In the proof of the sufficiency part of Theorem 3, we also provide a technique to obtain a supervisor whenever it exists: First obtain a minimal deterministic state machine $\overline{S}$ that generates K^{Σ_u, M_G}, the infimal prefix-closed (Σ_G^*, Σ_u)-controllable

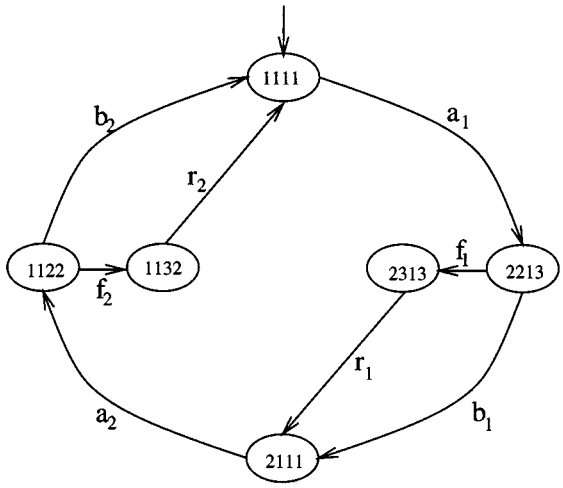

Fig. 9. Specification $K \subseteq L(G)$ for pumping station G.

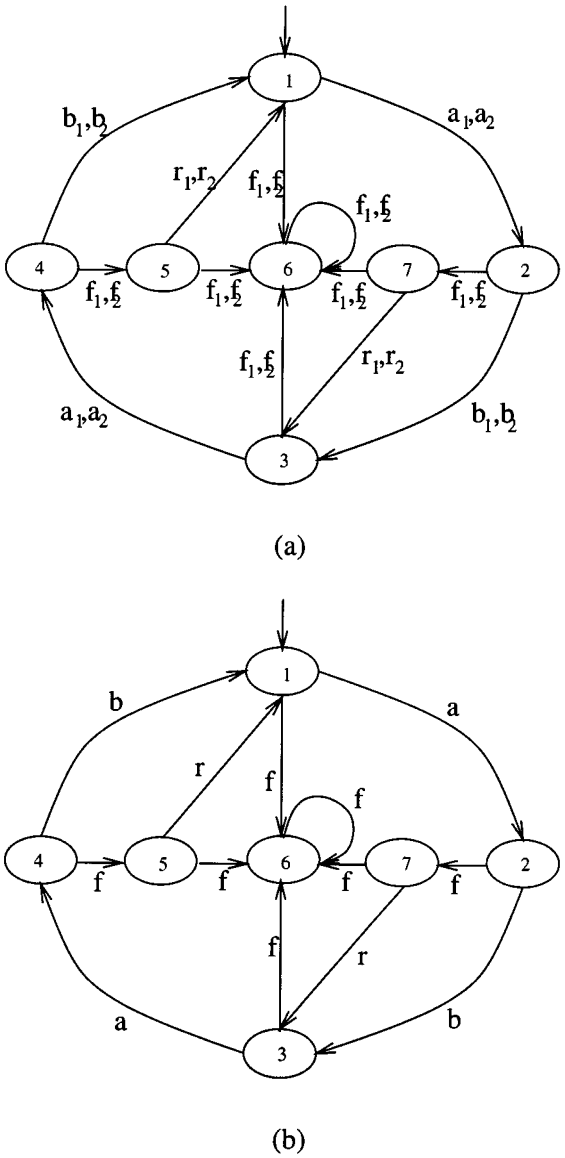

Fig. 10. Generator for K^{Σ_u, M_G} and supervisor S.

and (Σ_G^*, M_G)-normal superlanguage of the specification language, where $\Sigma_u := A - M_G^{-1}(M_S(B) - \{\epsilon\})$. Next replace each observable event label $\sigma_g \in \Sigma_G$ of any transition in $\overline{S}$ by an event label $\sigma_s \in \Sigma_S$ such that $M_S(\sigma_s) = M_G(\sigma_g) \neq \epsilon$, and delete all transitions of $\overline{S}$ on unobservable events.

In case the specification language does not satisfy either controllability or normality condition, a *maximally permissive* supervisor can be obtained by replacing the specification language by its supremal prefix-closed $(L(G), \Sigma_u)$-controllable and $(L(G), M_G)$-normal sublanguage, which can be computed using the existing algorithms (see, for example, [9, Section 4.2.2]).

V. Conclusion

In this paper, we introduced the notion of masked prioritized synchronous composition to model the mechanism of interaction of discrete event systems that interact with the environment through interfaces. This extends the formalism of prioritized synchronous composition, which assumes the identity interface mask function. This extension is particularly useful in supervisory control, where the limited control and observation capabilities of a supervisor are captured in the *external interconnection mechanism* of MPSC rather than the *internal state logic* of the supervisor.

We established a link between MPSC and PSC by showing that MPSC of two systems can be computed using PSC, by applying a "premasking" and a "postunmasking" operation. We also showed that whenever three or more systems interact at a common interface, their MPSC possesses the desired property of associativity. This is specially useful in context of supervisory control, where the plant and supervisor are distributed consisting of several interacting components.

We also studied the problem of obtaining a supervisor that controls a given discrete event plant by the MPSC based interaction when there are no driven events, so that the behavior of the composed system, when projected on the events of the plant, equals a given specification language. The familiar conditions of controllability and normality were found to be necessary and sufficient for the existence of a supervisor. A recent paper [8] studies the more general case of supervisory control when the set of driven events is nonempty.

References

[1] S. Balemi, "Input/output discrete event processes and communication delays," *Discrete Event Dynam. Syst.*, pp. 41–85, 1994.

[2] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya, "Supervisory control of discrete event processes with partial observation," *IEEE Trans. Automat. Contr.*, vol. 33, no. 3, pp. 249–260, 1988.

[3] M. Fabian, "On object oriented nondeterministic supervisory control," Ph.D. dissertation, Control Engineering Laboratory, Chalmers Univ., Goteberg, 1995.

[4] M. Heymann, "Concurrency and discrete event control," *IEEE Contr. Syst. Mag.*, vol. 10, no. 4, pp. 103–112, 1990.

[5] M. Heymann and G. Meyer, "Algebra of discrete event processes," NASA Ames Research Center, Moffett Field, CA, Tech. Rep. NASA 102 848, June 1991.

[6] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*. Reading, MA: Addison-Wesley, 1979.

[7] K. Inan, "An algebraic approach to supervisory control," *Math. Contr., Signals Syst.*, vol. 5, pp. 151–164, 1992.

[8] S. Jiang and R. Kumar, "Supervisory control of nondeterministic discrete event systems with driven events via masked prioritized synchronization," in *Proc. 1999 IEEE Conf. Decision and Control*, Pheonix, AZ, Dec. 1999.

[9] R. Kumar and V. K. Garg, *Modeling and Control of Logical Discrete Event Systems*. Boston, MA: Kluwer Academic , 1995.

[10] R. Kumar and M. A. Shayman, "Non-blocking supervisory control of nondeterministic systems via prioritized synchronization," *IEEE Trans. Automat. Contr.*, vol. 41, pp. 1160–1175, Aug. 1996.

[11] ——, "Supervisory control of real-time systems using prioritized synchronization," in *Hybrid Systems III*. New York: Springer-Verlag, 1996, vol. 1066.

[12] "Centralized and decentralized supervisory control of nondeterministic systems under partial observation," *SIAM J. Contr. Optim.*, vol. 35, no. 2, pp. 363–383, March 1997.

[13] F. Lin and W. M. Wonham, "On observability of discrete-event systems," *Inform. Sci.*, vol. 44, no. 3, pp. 173–198, 1988.

[14] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Contr. Optim.*, vol. 25, no. 1, pp. 206–230, 1987.

[15] M. Shayman and R. Kumar, "Supervisory control of nondeterministic systems with driven events via prioritized synchronization and trajectory models," *SIAM J. Contr. Optim.*, vol. 33, no. 2, pp. 469–497, Mar. 1995.

[16] "A new framework for supervisory control," in *Proc. 1995 Amer. Control Conf.*, Seattle, WA, June 1995, pp. 1341–1345.

[17] "Process objects/masked composition: An object oriented approach for modeling and control of discrete event systems," *IEEE Trans. Automat. Contr.*, vol. 44, no. 10, pp. 1864–1869, 1999.

[18] K. C. Wong and W. M. Wonham, "Hierarchical control of discrete event systems," *Discrete Event Dynam. Syst.*, vol. 6, pp. 241–273, 1996.

[19] H. Zhong and W. M. Wonham, "On the consistency of hierarchical supervision in discrete-event systems," *IEEE Trans. Automat. Contr.*, vol. 35, pp. 1125–1134, Oct. 1990.

Ratnesh Kumar received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 1987 and the M.S. and the Ph.D. degrees in electrical and computer engineering from the University of Texas at Austin in 1989 and 1991, respectively.

In 1991, he joined the Faculty of the University of Kentucky, Lexington, where he is an Associate Professor of electrical engineering. He has held visiting positions at the Institute of Systems Research, University of Maryland, College Park; the Applied Research Laboratory, Pennsylvania State University; and the NASA Ames Research Center. His primary research interest is discrete event systems and their applications to network protocols, manufacturing, formal verification, and intelligent control systems. He is a coauthor of *Modeling and Control of Logical Discrete Event Systems* (Norwell, MA: Kluwer Academic, 1995). He is an Associate Editor of the *SIAM Journal on Control and Optimization.*

Prof. Kumar was a recipient of the Microelectronics and Computer Development (MCD) Fellowship from the University of Texas at Austin. He received the Lalit Narain Das Memorial Gold Medal for the Best EE Student and the Ratan Swarup Memorial Gold Medal for the Best All-Round Student from the Indian Institute of Technology. He received the NSF Research Initiation Award, NASA-ASEE summer faculty fellowship award, and ARL Sabbatical Fellowship. He serves on the program committee for the IEEE Control Systems Society, the International Workshop on Discrete Event Systems, the International Symposium on Intelligent Control. He is an Associate Editor of the IEEE Transactions on Robotics and Automation and the IEEE Control Systems Society.

Michael Heymann received the B.Sc. and M.Sc. degrees from the Technion—Israel Institute of Technology, Haifa, in 1960 and 1962, respectively, and the Ph.D. degree from the University of Oklahoma, Norman, in 1965, all in chemical engineering.

During 1965–1966, he was on the Faculty of the University of Oklahoma. From 1966 to 1968, he was with Mobil Research and Development Corporation, engaged in research in process control and systems theory. From 1968 to 1970, he was with the Ben-Gurion University of the Negev, Beer-Sheva, Israel, where he founded and headed the Department of Chemical Engineering. Since 1970, he has been with the Technion, where he is currently Professor of computer science holding the Carl Fechheimer Chair in Electrical Engineering and Director of the Center for Intelligent Systems. He was previously with the Department of Applied Mathematics, of which he was Chairman, and the Department of Electrical Engineering. He held visiting positions at various institutes, including the University of Toronto, the University of Florida, the University of Eindhoven, Concordia University, CSIR, Yale University, the University of Bremen, the University of Newcastle, and the University of Kaiserslautern. During 1983–1984, 1988–1989, and 1995–1996, as well as during many summers, he was with NASA-Ames Research Center as an NRC-Senior Research Associate. Since 1997, he has been with NASA-Ames Research Center through a grant with the San Jose State Foundation. His research has covered topics in the areas of linear system theory, differential games, optimization, and adaptive control. His current interests are primarily in the areas of discrete event systems, hybrid systems, the theory of concurrent processes, and various formal aspects of human–machine interaction. He has been on the editorial boards of the *SIAM Journal of Control and Optimization* and *Systems and Control Letters*.