Proceedings of the 36th
Conference on Decision & Control
San Diego, California USA • December 1997

Masked prioritized synchronization for interaction & control of discrete event systems *

Ratnesh Kumar
Dept. of Electrical Eng.
University of Kentucky
Lexington, KY 40506

Michael Heymann
Dept. of Computer Sc.
Technion, Israel Inst. of Tech.
Haifa 32000, Israel

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. 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 show 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 controllability together with normality of the given specification serve as conditions for the existence of a supervisor.

1 Introduction

Supervisory control of discrete event systems (DES's) has been studied using prioritized synchronous composition (PSC) [3] in [4, 9, 7, 8, 1, 2]. In PSC, each system component possesses an event priority set specifying the set of events whose execution in the environment requires its participation. 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. 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 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 (resp. driven) events can only be blocked by the plant (resp. 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 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. Thus it is sensible to generalize the notion of PSC of systems (in which the interface mask is the identity function) to describe prioritized synchronization of systems interacting through non-identity interface masks.

An effort to generalize PSC in such a direction was first presented in [10], 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". One of the limitations of this formalism is the difficulty to model with it a wide range of realistic physical systems and their interaction.

In the present paper we introduce a very 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. 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.

Since a system may have multiple interfaces, the mask functions are not unique to the system, but rather to the particular interface of the system. When two or more systems interact via 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, it is necessary 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. (Otherwise an event in the environment could synchronize with both a priority event and a non-priority event, leading to ambiguity with respect to the requirement for system participation in the synchronous execution.)

2 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 research of the first author was supported in part by the Center for Manufacturing Systems, University of Kentucky, and in part by the National Science Foundation under the Grants NSF-ECS-9409712. The research of the second author was done while he was a Senior NRC Research Associate at NASA Ames Resear ch Cenrer, Moffett Field, CA, and was also supported in part by the Technion Fun d for Promotion of Research. The work was completed while the first author was visiting NASA Ames Research Center in the summer of 1996.

0-7803-3970-8/97 $10.00 © 1997 IEEE

the trace of zero length, denoted ϵ. 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 (NSM's) are used to model discrete event systems. A 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, $\delta_P : X_P \times (\Sigma_P \cup \{\epsilon\}) \to 2^{X_P}$ 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. 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. The *generated* and the *marked* languages of P are denoted $L(P)$ and $L_m(P)$, respectively. Given an interface mask $M : \Sigma_P \to \Sigma$ 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,σ,x') of P by the transition $(x, M(\sigma), x')$.

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 is 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 $\delta : X \times \Sigma \cup \{\epsilon\} \to 2^X$ is defined as:

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

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

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

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* [6] if $pr(K)\hat{\Sigma} \cap H \subseteq pr(K)$, and it is said to be (H, M)-*normal* [6] if $M^{-1}M(pr(K)) \cap H \subseteq pr(K)$. We further recall that controllability (resp. normality) is preserved under language union. Consequently, the supremal controllable (resp. the supremal normal) sublanguage of a given language exists. Similarly, controllability (resp. normality) of prefix-closed languages is preserved under language intersection, whence the infimal prefix-closed and controllable (resp. the infimal prefix-closed and normal) superlanguage of any given language exists.

3 Masked Prioritized Synchronization

In this section we formalize the notion of masked prioritized synchronous composition (MPSC) of two systems as discussed in the introduction. Two systems, modeled as NSM's P and Q, are connected as shown in Figure 1. P and Q

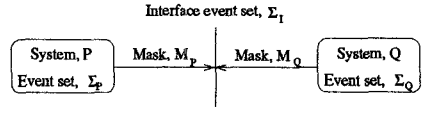

Figure 1: P and Q interacting via a common interface

evolve over their "internal" events Σ_P and Σ_Q respectively, and their event priority sets are $A \subseteq \Sigma_P$ and $B \subseteq \Sigma_Q$ respectively. The systems interact via a common interface consisting of the interface (or "external") events Σ_I. The interface mask of P is given by $M_P : \Sigma_P \to \Sigma_I$, and that of Q is given by $M_Q : \Sigma_Q \to \Sigma_I$. 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,

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

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.*

Example 1 Consider for example a pumping station G consisting of two identical pumps P_1 and P_2 and a synchronizer R sharing a common interface, as shown in Figure 2. The

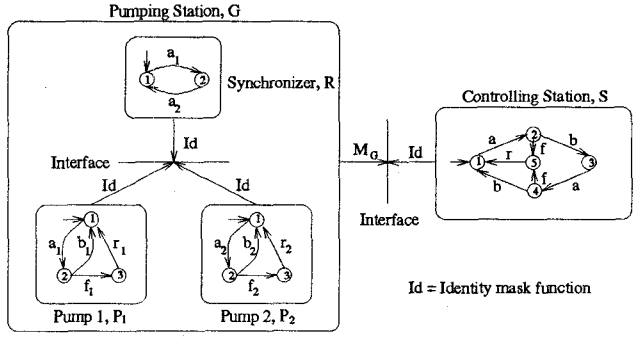

Figure 2: Interacting pumping and controlling stations

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_i's to a, b_i's to b, f_i's to f, and r_i's to r, at the interface with the controlling station. Clearly, M_G is priority consistent. Since a_i's and b_i's are priority events of G, and are masked to a and b respectively, which are priority events of S, a_i's and b_i's are controllable events. However, they are only partially controllable since both a_i's (resp. b_i's) are enabled in G when a (resp. b) is enabled in S. On the other hand, f_i's are uncontrollable events since they are in the priority set of G, and are masked to f which is a non-priority event of S. Similarly, r_i's are the driven events since they are non-priority events of G, and are masked to r which is a priority event of S.

Definition 1 Consider systems P and Q interacting via a common interface with events Σ_I as shown in Figure 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 $X := X_P \times X_Q$, $\Sigma := [\Sigma_P \times \Sigma_Q] \cup \Sigma_P \cup \Sigma_Q$, $x^0 := (x_P^0, x_Q^0)$, $X^m := X_P^m \times X_Q^m$, and $\delta : X \times (\Sigma \cup \{\epsilon\}) \to 2^X$ is defined as follows:

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

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

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

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

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

The event priority set of $P \ {}_A\|_B \ Q$ is given by

$$A \oplus B := [A \times \Sigma_Q] \cup [\Sigma_P \times B] \cup A \cup B.$$

Intuitively, P and Q interact by either 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, alternatively, they execute individual events (without participation of the other system) whenever the event is either unobservable at the interface, or no corresponding event of the other system (with respect to interface identity) 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.

Remark 1 The "external" behavior of the MPSC of P and Q observed at the interface, denoted $(P \ {}_A\|_B \ Q)\uparrow\Sigma_I$, can be obtained by replacing each transition 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. $\forall (x, \sigma_p, x') \in X \times \Sigma_P \times X$, replace it by $(x, M_P(\sigma_p), x')$
3. $\forall (x, \sigma_q, x') \in X \times \Sigma_Q \times X$, 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 $(P \ {}_A\|_B \ Q)\uparrow\Sigma_P$, can be obtained by "erasing" each Σ_Q-event label from all 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, σ_p, x')
2. $\forall (x, \sigma_q, x') \in X \times \Sigma_Q \times X$, replace it by (x, ϵ, x')

It is easily seen that the generated (resp. marked) language of $(P \ {}_A\|_B \ Q)\uparrow\Sigma_P$ is contained in the generated (resp. 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 2 The MPSC of the pumping station G and the controlling station S of Example 1 is shown in Figure 3. In each state number 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 (resp. second) is the event label of the corresponding transition in G (resp. S).

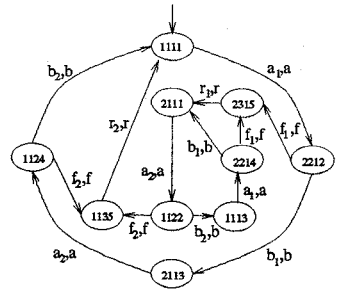

Figure 3: MPSC of pumping and controlling stations

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 it is masked to a which is in the priority set 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 Figure 3.

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 via a common interface with events Σ_I as shown in Figure 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" NSM's $M_P(P)$ 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 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. $((x_p,x_q),\sigma_p,(x'_p,x'_q)) \in X \times \Sigma_P \times X$ if $x'_p \in \delta_P(x_p,\sigma_p), x_q = x'_q$, $\delta_Q(x_q, M_Q^{-1}(\sigma))$, $M_Q^{-1}(\sigma) \cap B = \emptyset$, $M_P(\sigma_p) = \sigma$.
      iii. $((x_p,x_q),\sigma_q,(x'_p,x'_q)) \in X \times \Sigma_Q \times X$ 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) $\forall((x_p,x_q),\epsilon,(x'_p,x'_q)) \in X \times \{\epsilon\} \times X$, replace it by transitions
      i. $((x_p,x_q),\sigma_p,(x'_p,x'_q)) \in X \times (\Sigma_P \cup \{\epsilon\}) \times X$ if $x'_p \in \delta_P(x_p,\sigma_p)$, $x_q = x'_q, M_P(\sigma_p) = \epsilon$.
      ii. $((x_p,x_q),\sigma_q,(x'_p,x'_q)) \in X \times (\Sigma_Q \cup \{\epsilon\}) \times X$ if $x_p = x'_p, x'_q \in \delta_Q(x_q,\sigma_q)$, $M_Q(\sigma_q) = \epsilon$.

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

We next investigate the associativity of MPSC. It is known from [4], Theorem 13.4 (see also [7] where a detailed proof was given) that PSC is associative, i.e., given NSM's 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 via a common interface. Consider for example three NSM's P,Q,R with respective event priority set $A \subseteq \Sigma_P, B \subseteq \Sigma_Q, C \subseteq \Sigma_R$ interacting via a common interface as shown in Figure 4. The interface masks of P,Q,R are given by M_P, M_Q, M_R respectively.

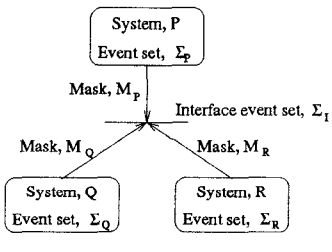

Figure 4: P,Q,R interacting via a common interface

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 Figure 5. In Figure 5(a) composition of P,Q is first obtained and next this is composed with R, whereas in Figure 5(b) composition of Q,R is first obtained which is then composed with P. We use the mask function pair

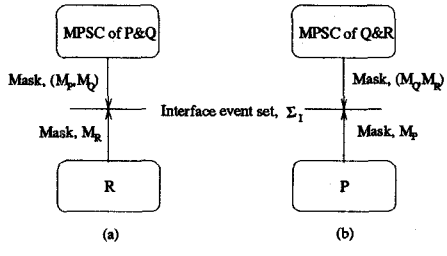

Figure 5: Two ways of associating composition of P,Q,R

(M_P, M_Q) to denote the mask function of the composition $P\ {}_A\|_B\ Q$, the first (resp. second) component of which applies to transitions with an event label in Σ_P (resp. 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 via 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

$$[P\ {}_A\|_B\ Q]\ {}_{(A\oplus B)}\|_C\ R = P\ {}_A\|_{(B\oplus C)}\ [Q\ {}_B\|_C\ R].$$

4 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 Figure 1. The plant is modeled by a 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 of enabling/disabling events 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 is either in the priority set of the plant or it is identified via the interface masks with some event that is in the priority set of the supervisor, i.e., $M_G(\Sigma_G - A) \subseteq \overline{M}_S(B) := M_S(B) - \{\epsilon\}$. 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. This rules out the possibility that a non-priority event of the plant is identified with no event of the supervisor or is masked to ϵ. Table 1 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. A similar reasoning is used to obtain Table 2

plant event	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 1: Properties of plant events

summarizing the properties of the events of the supervisor.

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

Table 2: Properties of supervisor events

The control objective is given by a specification $K \subseteq \Sigma_G^*$ 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.

Example 3 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 most one pump is running at any given time.* This desired specification is shown in Figure 6. The

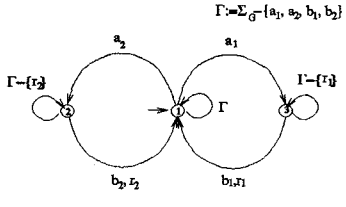

Figure 6: Design specification for the pumping station

event priority set and the interface mask function of a controlling station to be designed enforcing such a specification are as given in Example 2.

We are interested in obtaining a necessary and sufficient condition for the existence of a supervisor for the supervisory control problem described above. Define the set of *uncontrollable* events $\Sigma_u := A - M_G^{-1}M_S(B)$ to be the priority events of the plant that are not identified with any priority events of the supervisor. Note that Σ_u includes any priority events of the plant that are unobservable under M_G. 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 state two preliminary results about controllability and normality.

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 first lemma states that $K^{\Sigma_u, M_G} \cap L(G)$ equals the infimal prefix-closed $(L(G), \Sigma_u)$-controllable and $(L(G), M_G)$-normal superlanguage of K.

Lemma 1 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.

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

Lemma 2 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. $\forall u, v \in \Sigma_G^*, \sigma, \sigma' \in (\Sigma_G \cup \{\epsilon\}) : [u\sigma v \in pr(K), M_G(\sigma) = M_G(\sigma')] \Rightarrow [u\sigma' v \in pr(K)]$.

Remark 2 Let S be a trim deterministic state machine that accepts a (Σ_G^*, M_G)-normal language $K \subseteq \Sigma_G^*$. Then it follows from the third assertion of Lemma 2 that 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.

Theorem 3 Consider a plant G with priority set $A \subseteq \Sigma_G$ and priority consistent interface mask $M_G : \Sigma_G \to \Sigma_I$ such that $\epsilon \notin M_G(\Sigma_G - A)$. 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 such that $B \supseteq M_S^{-1}M_G(\Sigma_G - A)$. 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 $\Sigma_u := A - M_G^{-1}(M_S(B) - \{\epsilon\})$.

Remark 3 Theorem 3 provides a necessary and sufficient condition for the existence of a deterministic supervisor 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 [6, Sections 3.2.3, 4.2.3]).

The proof of Theorem 3 (omitted here for brevity) is constructive, and provides a technique to obtain a supervisor whenever one exists: First obtain a minimal deterministic state machine $\overline{S}$ that generates K^{Σ_u, M_G}, where $\Sigma_u := A - M_G^{-1}(M_S(B) - \{\epsilon\})$; and 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 [6, Section 4.2.2]).

Example 4 We now return to Example 3 where a specification for the pumping station G was formulated. This specification, shown in Figure 6, when intersected with the generated language of the pumping station imposes the language $K \subseteq L(G)$ as shown in Figure 7. In this figure each

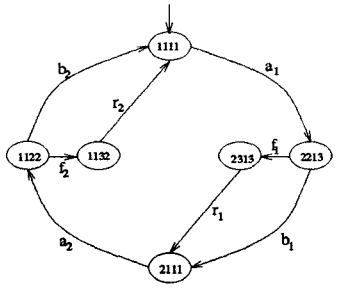

Figure 7: Specification $K \subseteq L(G)$ for pumping station G

state has four components, the first component denotes the state of the synchronizer (Figure 2), the second that of the pump 1 (Figure 2), the third that of the pump 2 (Figure 2), and the last that of the specification (Figure 6).

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_i's to a, b_i's to b, f_i's to f, and r_i's to r. We use Lemma 1 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 Figure 8(a). Then it is easy to see that the syn-

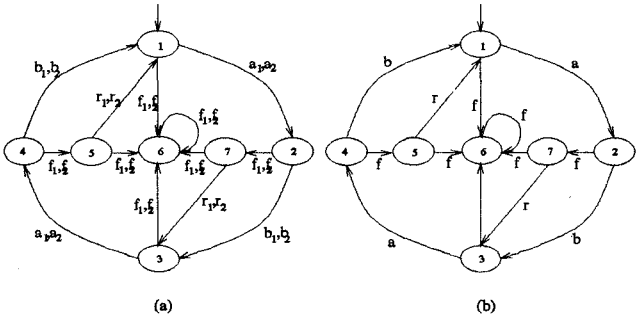

Figure 8: Generator for K^{Σ_u, M_G} and supervisor S

chronous composition of this with G yields the same state machine as the generator for K shown in Figure 7. This establishes that K is $(L(G), \Sigma_u)$-controllable and $(L(G), M_G)$-normal.

Using the procedure outlined in Remark 3 we arrive at the supervisor shown in Figure 8(b) that enforces K as the projected behavior on the event set Σ_G of the composed system $G \,{}_A\|_B\, S$.

5 Conclusion

In this paper we introduced the notion of masked prioritized synchronous composition (MPSC), to model the mechanism of interaction of discrete event systems that interact with the environment through interfaces. This 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 "pre-masking" and a "post-unmasking" operation. We also showed that whenever three or more systems interact at a common interface, their MPSC possesses the desired property of associativity.

We studied the problem of obtaining a supervisor that controls a given discrete event plant by the MPSC based interaction, 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.

References

[1] S. Balemi. Input/output discrete event processes and communication delays. *Discrete Event Dynamical Systems: Theory and Applications*, pages 41–85, 1994.

[2] M. Fabian. *On Object Oriented Nondeterministic Supervisory Control.* PhD thesis, Control Engineering Laboratory, Chalmers University, Goteberg, 1995.

[3] M. Heymann. Concurrency and discrete event control. *IEEE Control Systems Magazine*, 10(4):103–112, 1990.

[4] M. Heymann and G. Meyer. Algebra of discrete event processes. Technical Report NASA 102848, NASA Ames Research Center, Moffett Field, CA, June 1991.

[5] K. Inan. An algebraic approach to supervisory control. *Mathematics of Control, Signals and Systems*, 5:151–164, 1992.

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

[7] R. Kumar and M. A. Shayman. Non-blocking supervisory control of nondeterministic systems via prioritized synchronization. *IEEE Transactions on Automatic Control*, 41(8):1160–1175, August 1996.

[8] R. Kumar and M. A. Shayman. Centralized and decentralized supervisory control of nondeterministic systems under partial observation. *SIAM Journal of Control and Optimization*, 35(2):363–383, March 1997.

[9] M. Shayman and R. Kumar. Supervisory control of nondeterministic systems with driven events via prioritized synchronization and trajectory models. *SIAM Journal of Control and Optimization*, 33(2):469–497, March 1995.

[10] M. A. Shayman and R. Kumar. A new framework for supervisory control. In *Proceedings of 1995 American Control Conference*, pages 1341–1345, Seattle, WA, June 1995.