Masked Prioritized Synchronization for Interaction and Control of Discrete Event Systems *

Ratnesh Kumar
Department of Electrical Engineering
University of Kentucky
Lexington, KY 40506-0046

Michael Heymann
Department of Computer Science
Technion, Israel Institute of Technology
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. 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.

Keywords: Discrete event system, supervisory control, masked prioritized synchronization, controllability, normality

*The research of the first author was supported in part by the Center for Manufacturing Systems, University of Kentucky, in part by the National Science Foundation under the Grants NSF-ECS-9409712 and NSF-ECS-9709796, and in part by the Office of Naval Research under the Grant ONR-N00014-96-1-5026. The research of the second author was done while he was a Senior NRC Research Associate at NASA Ames Research Center, Moffett Field, CA, and was also supported in part by the Technion Fund for Promotion of Research. The work was completed while the first author was visiting Dr. Geroge Meyer at the NASA Ames Research Center in the summer of 1996.
1 Introduction

Supervisory control of discrete event systems (DES’s) has been studied using prioritized synchronous composition (PSC) [4] in [5, 16, 10, 11, 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 which 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 is 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, and the need to ensure that interacting systems are control compatible [15, 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 (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 non-specific, 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. When systems interact through non-identity 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 [15, 17]. Thus it is sensible to generalize the notion of PSC to describe prioritized synchronization of systems interacting through non-identity
interface masks. This then will allow us to model interactions of systems without the need to ensure that they are control or observation compatible [15, 17].

An effort to generalize PSC in such a direction was first presented in [15, 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 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 (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, 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]. This however we do not explore in this paper.

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 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 non-projection type control mask associated with the plant.

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 trace of zero length, denoted ∈. For an event set Σ, we use Σ to denote Σ ∪ {∈}. A subset of Σ* is called a language. Given trace s ∈ Σ*, we let |s| denote its length. For a language H ⊆ Σ*, 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, Σ_P, δ_P, x_0^P, X^m_P), where X_P is its set of states, Σ_P is its set of events, δ_P : X_P × (Σ_P ∪ {∈}) → 2^{X_P} is its transition function, x_0^P ∈ X_P is its initial state, and X^m_P ⊆ X_P is its set of marked, or final, states. For any set of states X ⊆ X_P and set of events Σ ⊆ Σ_P, the notation δ_P(X, Σ) is used to denote ∪_{x ∈ X} ∪_{σ ∈ Σ} δ_P(x, σ). P is called deterministic if |δ_P(x, σ)| ≤ 1 for all x ∈ X_P and σ ∈ Σ_P, and |δ_P(x, ∈)| = 0. A triple (x, σ, x’) ∈ X_P × (Σ_P ∪ {∈}) × X_P is called a transition if x’ ∈ δ_P(x, σ); If σ = ∈, the transition is called silent or an inactive transition. Given an interface mask M : Σ_P → Σ ∪ {∈} from the internal events Σ_P to external interface events Σ, the “masked” NSM M(P) := (X_P, Σ, δ_{M(P)}, x_0^P, X^m_P) is obtained by replacing each transition (x, σ, x’) ∈ X_P × Σ_P × X_P of P by the transition (x, M(σ), x’). The interface mask M_P is extended to be defined over traces in Σ_P as follows:

M_P(∈) := ∈; ∀s ∈ Σ_P, σ ∈ Σ_P : M_P(sσ) := M_P(s)M_P(σ).

For x ∈ X_P, the closure of x, denoted ε_P^∗(x), is the set of states reached by the execution of zero or more inactive transitions from the state x, and is defined recursively as:

x ∈ ε_P^∗(x); x’ ∈ ε_P^∗(x) ⇒ δ_P(x’, ∈) ⊆ ε_P^∗(x).

The closure map can also be used to extend the definition of transition function from events to traces. Thus we obtain δ_P : X_P × Σ_P → 2^{X_P}, which is defined inductively as

∀x ∈ X : [δ_P^∗(x, ∈) := ε_P^∗(x); ∀s ∈ Σ_P, σ ∈ Σ_P : δ_P^∗(x, sσ) := ε_P^∗(δ_P(δ_P^∗(x, s), σ))],

where for any X ⊆ X_P, δ_P(X, σ) := ∪_{x ∈ X} δ_P(x, σ). Using this extended transition function, the generated and the marked languages of P, denoted, respectively, L(P) and L_m(P), are defined as:

L(P) := {s ∈ Σ_P | δ_P^∗(x_0^P, s) ≠ ∅}; L_m(P) := {s ∈ Σ_P | δ_P^∗(x_0^P, s) ∩ X^m_P ≠ ∅}.
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 $P_A \parallel_B Q$, is the NSM defined as $P_A \parallel_B Q := (X, \Sigma, \delta, x^0, X^m)$, where $X := X_P \times X_Q$, $x^0 := (x^0_P, x^0_Q)$, $X^m := X^m_P \times X^m_Q$, and the transition function $\delta : X \times (\Sigma \cup \{\epsilon\}) \to 2^X$ is defined as:

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

$$\delta(x, \sigma) := \begin{cases} 
\delta_P(x_p, \sigma) \times \delta_Q(x_q, \sigma) & \text{if } \delta_P(x_p, \sigma) \neq \emptyset, \delta_Q(x_q, \sigma) \neq \emptyset \\
\delta_P(x_p, \sigma) \times \{x_q\} & \text{if } \delta_P(x_p, \sigma) \neq \emptyset, \delta_Q(x_q, \sigma) = \emptyset, \sigma \not\in B \\
\{x_p\} \times \delta_Q(x_q, \sigma) & \text{if } \delta_Q(x_q, \sigma) \neq \emptyset, \delta_P(x_p, \sigma) = \emptyset, \sigma \not\in A \\
\emptyset & \text{otherwise}, 
\end{cases}$$

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

The event priority set of $P_A \parallel_B Q$ 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 $\epsilon$-transitions asynchronously. In the special case when the event priority sets of both systems are the entire event set $\Sigma$, 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 $\Sigma$, a prefix-closed language $H \subseteq \Sigma^*$, a set of events $\hat{\Sigma} \subseteq \Sigma$, and a mask function $M$ defined over $\Sigma$, 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 it is said to be $(H, M)$-normal [13] 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...
Figure 1: $P$ and $Q$ interacting at a common interface

$P$ and $Q$, are connected as shown in Figure 1. $P$ and $Q$ evolve over their “internal” events $\Sigma_P$ and $\Sigma_Q$ respectively, 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 $\Sigma_I$. The interface mask of $P$ is given by $M_P : \Sigma_P \rightarrow \Sigma_I \cup \{\epsilon\}$, and that of $Q$ is given by $M_Q : \Sigma_Q \rightarrow \Sigma_I \cup \{\epsilon\}$. The composed system is denoted by $P \parallel Q$, 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,

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

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 Figure 2 $P$ and $Q$ interact at a pair of interfaces. The interface mask of $P$ for the two interfaces are $M_{P1} : \Sigma_P \rightarrow \Sigma_{I1}$ and $M_{P2} : \Sigma_P \rightarrow \Sigma_{I2}$ respectively, and similarly those for $Q$ are $M_{Q1} : \Sigma_Q \rightarrow \Sigma_{I1}$ and $M_{Q2} : \Sigma_Q \rightarrow \Sigma_{I2}$ respectively. This scenario

Figure 2: $P$ and $Q$ interacting at two interfaces

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:

$$M_P : \Sigma_P \rightarrow \Sigma_{I1} \times \Sigma_{I2}, \quad M_P(\sigma) := (M_{P1}(\sigma), M_{P2}(\sigma)), \forall \sigma \in \Sigma_P;$$

$$M_Q : \Sigma_Q \rightarrow \Sigma_{I1} \times \Sigma_{I2}, \quad M_Q(\sigma) := (M_{Q1}(\sigma), M_{Q2}(\sigma)), \forall \sigma \in \Sigma_Q.$$

**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 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 $a_i$’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

![Figure 3: Interacting pumping and controlling stations](image-url)
$a$ and $b$ respectively, which are priority events of $S$, $a_i$'s and $b_i$'s are controllable events. However, they are only non-uniquely 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 at a common interface with events $\Sigma_I$ 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 $P \|_B Q := (X, \Sigma, \delta, x^0, X^m)$, where $X := X_P \times X_Q$, $\Sigma := \Sigma_P \times \Sigma_Q$, $x^0 := (x^0_P, x^0_Q)$, $X^m := X^m_P \times X^m_Q$, and $\delta : X \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^X$ is defined as follows:

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

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

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

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

Intuitively, $P$ and $Q$ interact by either executing synchronously events $\sigma_p$ and $\sigma_q$, 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 event of the other system that has the same observation as 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 $\epsilon$-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 Figure 4(a)-(b). We have $\Sigma_P = \{a, b, c\}, \Sigma_Q = \{e, f, g\}, A = \{a, b\}, B = \{e, f\}$. 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, $P \|_B Q$ is
shown in Figure 4(c). The transitions labeled \((b, e), (b, f), (a, e), (a, f)\) are as in the first case of Definition 1, those labeled \((c, e), (b, e)\) are as in the second case of the definition, that labeled \((e, g)\) is as in the third case of the definition, and finally that labeled \(\epsilon\) 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

\[
P_A|_B Q = P_{M_P^{-1}M_P(A) M_Q^{-1}M_Q(B)} Q,
\]

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 \(\bar{A} = M_P^{-1}M_P(A)\) and \(\bar{B} := M_Q^{-1}M_Q(B)\) 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 \(\Sigma_I\)** and denoted \((P_A|_B Q)|\Sigma_I\), 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 \(\Sigma_I\).

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

\(\forall (x, (\sigma_p, \sigma_q), x') \in X \times (\Sigma_P \times \Sigma_Q) \times X\), replace it by \((x, \sigma_p, x')\)
It is easily seen that the generated (resp. marked) language of \((P_A\parallel_B Q)\Sigma_P\) 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 3** The MPSC of the pumping station \(G\) and the controlling station \(S\) of Example 1 is shown in Figure 5. 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. \(S\)).

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 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 Figure 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 \(\Sigma_I\) 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 \(P_A\parallel_B Q := (X, \Sigma, \delta, x^0, X^m)\) 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 $M_P(P) \Box_{P(A)} \parallel M_Q(B) M_Q(Q)$

3. Replace each transition in $M_P(P) \Box_{P(A)} \parallel M_Q(B) M_Q(Q)$ to obtain NSM $R$ as follows:

(a) $\forall ((x_p, x_q), \sigma, (x'_p, x'_q)) \in X \times \Sigma_p \times X$, 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

$$x'_p \in \delta_P(x_p, \sigma_p), x_q = x'_q, \delta_Q(x_q, M_Q^{-1}(\sigma)), M_Q^{-1}(\sigma) \cap B = \emptyset, M_P(\sigma_p) = \sigma.$$  

iii. $((x_p, x_q), (\sigma, \epsilon), (x'_p, x'_q)) \in X \times (\{\epsilon\} \times \Sigma_q) \times X$ if

$$x_p = x'_p, \delta_P(x_p, M^{-1}(\sigma)), M^{-1}(\sigma) \cap A = \emptyset, x'_q \in \delta_Q(x_q, \sigma_q), M_Q(\sigma_q) = \sigma.$$  

(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 $P \parallel_B Q = R$.

**Proof:** Since the two NSM’s $P \parallel_B Q$ and $R$ have identical states $X := X_P \times X_Q$, identical events $\Sigma := (\Sigma_P \times \Sigma_Q) \cup \Sigma_P \cup \Sigma_Q$, identical initial state $x^0 := (x^0_P, x^0_Q)$, and identical final states $X^m := X^m_P \times X^m_Q$, 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 $P \parallel_B Q$.

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 3(a)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 $P \parallel_B Q$.

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 3(a)ii or 3(b)i. of Algorithm 1. In the first case from clause 3(a)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
3(b)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

$$((x_p, x_q), \epsilon, (x'_p, x'_q)) \in X \times \{\epsilon\} \times X \times \{\epsilon\} \times X$$

of $R$ is also a transition of $P_{A\|B} \ Q$.

Finally, consider a transition $((x_p, x_q), \epsilon, (x'_p, x'_q)) \in X \times \{\epsilon\} \times X$ as in clause 3(b)\(i\). 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 $((x_p, x_q), (\sigma_p, \sigma_q), (x'_p, x'_q)) \in X \times \{\epsilon\} \times X \times \{\epsilon\} \times X$ of $P_{A\|B} \ Q$ as in the first clause of Definition 1. Then from this clause $x'_p \in \delta_p(x_p, \sigma_p)$ and either $\delta_p(x_q, M_p^{-1}(\sigma_p))$, $M_p^{-1}(\sigma_p)) \cap B = \emptyset$ 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 $x'_p \in \delta_p(x_p, \sigma)$, and $\delta_p(x_q, \sigma) = \sigma Q = \sigma Q(\sigma) = \emptyset$. Also since $\sigma Q(\sigma) \cap B = \emptyset$, $\sigma \notin M_Q(B) \supset M_Q(B)$. 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 $M_Q(\sigma)$. So by applying clause 3(a)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 $((x_p, x_q), (\sigma_p, \epsilon), (x'_p, x'_q)) \in X \times \{\epsilon\} \times X$ 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 either $\delta_p(x_q, M_p^{-1}(\sigma_p))$, $M_p^{-1}(\sigma_p)) \cap B = \emptyset$ 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 $x'_p \in \delta_p(x_p, \sigma)$, and $\delta_p(x_q, \sigma) = \sigma Q = \sigma Q(\sigma) = \emptyset$. Also since $\sigma Q(\sigma) \cap B = \emptyset$, $\sigma \notin M_Q(B) \supset M_Q(B)$. 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 $M_Q(\sigma)$. So by applying clause 3(b)i. 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 $((x_p, x_q), \sigma_p, (x'_p, x'_q))$ is a transition of $M_Q(\sigma)$. So by applying the clause 3(b)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 $((x_p, x_q), (\epsilon, \sigma_q), (x'_p, x'_q)) \in X \times \{\epsilon\} \times X$ of $P_{A\|B} \ Q$ is also a transition of $R$.

Finally, consider a transition $((x_p, x_q), (\epsilon, \sigma_p, \sigma_q)) \in X \times \{\epsilon\} \times X \times \{\epsilon\}$ of $P_{A\|B} \ Q$. Then from the last clause in Definition 1, $x'_p \in \delta_p(x_p, \epsilon)$, which implies $x'_p \in \delta_p(x_p, \epsilon)$. 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 $M_Q(\sigma)$. So by applying the clause 3(b)i. of Algorithm 1 we conclude that $((x_p, x_q), (\epsilon, \sigma_p, \sigma_q))$ 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 “pre-masking” and a “post-unmasking” operation. This fact is not explored in this paper any further. A consequence of Theorem 1, however, is that $P_{A\|B} \ Q \Sigma I = M_P(P_{A\|B}) M_Q(Q)$, 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 NSM’s $P, Q, R$ that evolve over a common event set $\Sigma$ along with their respective priority sets $A, B, C \subseteq \Sigma$ the following holds:

$$(P_A \parallel_B Q)_{A \cup B} \parallel_C R = P_A \parallel_B Q \parallel_C (Q_B \parallel_C R).$$

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 NSM’s $P, Q, R$ 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 Figure 6. The interface masks of $P, Q, R$ are given by $M_P, M_Q, M_R$ respectively.

![Diagram of three systems interacting at a common interface](image)

**Figure 6: $P, Q, R$ interacting at 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 7. In Figure 7(a) composition of $P, Q$ is first obtained and next this is composed with $R$, whereas in Figure 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 $P_A \parallel_B Q$, the first (resp. second) component of which applies to transitions with an event label in $\Sigma_P$ (resp. $\Sigma_Q$). Note that whenever a transition in $P_A \parallel_B Q$ 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 $P_A \parallel_B Q$ interacts with $R$.

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

$$[P_A \parallel_B Q]_{(A \cup B)} \parallel_C R = P_A \parallel_{(B \cup C)} [Q_B \parallel_C R].$$
Proof: Let \( X := X_P \times X_Q \times X_R \), \( \Sigma := \Sigma_P \times \Sigma_Q \times \Sigma_R \) 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 Figure 7(a)-(b) has identical transition 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 Figure 7(a)-(b) has identical transition function and is given by:

\[
\forall x = (x_p, x_q, x_r) \in X_P \times X_Q \times X_R, \sigma = (\sigma_p, \sigma_q, \sigma_r) \in \Sigma_P \times \Sigma_Q \times \Sigma_R:
\]

\[
\delta(x, \sigma) :=\begin{cases} 
\delta_p(x_p, \sigma_p) \times \delta_q(x_q, \sigma_q) \times \delta_r(x_r, \sigma_r) & \text{if } M_P(\sigma_p) = M_Q(\sigma_q) = M_R(\sigma_r) \neq \epsilon, \\
\delta_p(x_p, \sigma_p) & \text{if } \sigma := M_p(\sigma_p) = M_Q(\sigma_q) \neq \epsilon, \\
\delta_q(x_q, \sigma_q) \times \{x_r\} & \text{if } \sigma := M_P(\sigma_p) = M_R(\sigma_r) \neq \epsilon, \\
\delta_r(x_r, \sigma_r) & \text{if } \sigma := M_Q(\sigma_q) = M_R(\sigma_r) \neq \epsilon, \\
\emptyset & \text{otherwise.}
\end{cases}
\]

\[
\delta(x, (\sigma_p, \sigma_q, \epsilon)) :=\begin{cases} 
\emptyset & \text{otherwise.}
\end{cases}
\]

\[
\delta(x, (\epsilon, \sigma_q, \sigma_r)) :=\begin{cases} 
\{x_p\} \times \delta_q(x_q, \sigma_q) \times \delta_r(x_r, \sigma_r) & \text{if } \sigma := M_Q(\sigma_q) = M_R(\sigma_r) \neq \epsilon, \\
\emptyset & \text{otherwise.}
\end{cases}
\]

\[
\delta(x, (\sigma_p, \epsilon, \sigma_r)) :=\begin{cases} 
\delta_p(x_p, \sigma_p) \times \{x_q\} \times \delta_r(x_r, \sigma_r) & \text{if } \sigma := M_P(\sigma_p) = M_R(\sigma_r) \neq \epsilon, \\
\emptyset & \text{otherwise.}
\end{cases}
\]

\[
\delta(x, (\sigma_p, \epsilon, \epsilon)) :=\begin{cases} 
\delta_p(x_p, \sigma_p) \times \{x_q\} \times \{x_r\} & \text{if } \delta_p(x_p, \sigma_p) \neq \emptyset, [M_P(\sigma_p) = \epsilon] \\
\emptyset & \text{otherwise.}
\end{cases}
\]

Figure 7: Two ways of associating composition of \( P, Q, R \)
\[
\delta(x, (\epsilon, \sigma_q, \epsilon)) := \begin{cases} 
\{x_p\} \times \delta_Q(x_q, \sigma_q) \times \{x_r\} & \text{if } \delta_Q(x_q, \sigma_q) \neq \emptyset, [[M_Q(\sigma_q) = \epsilon] \\
\emptyset & \text{otherwise}
\end{cases}
\]

\[
\delta(x, (\epsilon, \epsilon, \sigma_r)) := \begin{cases} 
\{x_p\} \times \{x_q\} \times \delta_R(x_r, \sigma_r) & \text{if } \delta_R(x_r, \sigma_r) \neq \emptyset, [[M_R(\sigma_r) = \epsilon] \\
\emptyset & \text{otherwise}
\end{cases}
\]

Finally the event priority set of the composition of the three systems in either of the two configurations of Figure 7(a)-(b) equals

\[
[A \oplus B] \oplus C = A \oplus [B \oplus C] = [A \times \Sigma_Q \times \Sigma_R] \cup [\Sigma_P \times B \times \Sigma_R] \cup [\Sigma_P \times \Sigma_Q \times C]
\]

**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 Figure 6.

## 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 under the restriction that the set of driven events is empty. 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 \( \Sigma_I \) 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 \( S := (X_S, \Sigma_S, \delta_S, x_S^0, X_S^n) \). 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 be identified via the interface masks with some event that is in the priority set of the supervisor, i.e., \( \Sigma_G - A \subseteq M_G^{-1}(M_S(B)) \), where \( M_S(B) := M_S(B) - \{\epsilon\} \). In other words, the set of plant events is the union of controllable events: \( A \cap M_G^{-1}(M_S(B)) \), the uncontrollable events: \( A - M_G^{-1}(M_S(B)) := \Sigma_u \), and the driven events: \( M_G^{-1}(M_S(B)) - A = \Sigma_G - A \). 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 non-priority event of the plant is identified...
with no priority event of the supervisor or is masked to $\epsilon$. The above requirement together with the assumption that there are no driven events implies that $\Sigma_G = A$.

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. Similarly Table 2 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.

<table>
<thead>
<tr>
<th>plant event</th>
<th>identified with supervisor event</th>
<th>event type</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority</td>
<td>priority</td>
<td>controllable &amp; observable</td>
</tr>
<tr>
<td>priority</td>
<td>non-priority</td>
<td>uncontrollable &amp; observable</td>
</tr>
<tr>
<td>priority</td>
<td>no event (epsilon)</td>
<td>uncontrollable &amp; unobservable</td>
</tr>
<tr>
<td>non-priority</td>
<td>priority</td>
<td>driven &amp; observable</td>
</tr>
<tr>
<td>non-priority</td>
<td>non-priority</td>
<td>non-existent</td>
</tr>
<tr>
<td>non-priority</td>
<td>no event (epsilon)</td>
<td>non-existent</td>
</tr>
</tbody>
</table>

Table 1: Controllability & observability property of plant events

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 Figure 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 \((\Sigma_G^*, M_G)\)-normality.

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

1. \(K\) is \((\Sigma_G^*, M_G)\)-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 \((\Sigma_G^*, M_G)\)-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 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 \(s := \bar{s}\sigma, s' := \bar{s}\sigma'\), where \(\sigma, \sigma' \in \Sigma_G\). Then we have three possible case: (i) \(M_G(\bar{s}\sigma) = M_G(\bar{s}\sigma')\) and \(M_G(\sigma) = M_G(\sigma')\), or (ii) \(M_G(s) = M_G(\bar{s}\sigma')\) and \(M_G(\sigma') = \epsilon\), or (iii) \(M_G(\bar{s}\sigma) = M_G(s')\) and \(M_G(\sigma) = \epsilon\). We only analyze the first case; the others can be analyzed similarly. In the first case since \(M_G(\bar{s}\sigma) = M_G(\bar{s}\sigma')\), and since from the hypothesis of the second assertion \(\pi\sigma t = st \in pr(K)\), it follows from the induction hypothesis that \(\bar{s}\sigma t \in pr(K)\). Set \(u := \bar{s}\sigma, v := t\). Then since
\(M_G(\sigma) = M_G(\sigma')\), and since \(u\sigma v = \pi'\sigma t \in pr(K)\), it follows from the third assertion that \(u\sigma' v = \pi'\sigma' t = s't \in pr(K)\), completing the induction step.

\[ \Box \]

**Remark 4** Let \(S\) be a trim [6] deterministic state machine that accepts a \((\Sigma_G, M_G)\)-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 \(K^{\Sigma_u, M_G}\) denote the infimal prefix-closed \((\Sigma_G^*, \Sigma_u)\)-controllable and \((\Sigma_G^*, M_G)\)-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 \((\Sigma_G^*, \Sigma_u)\)-controllability of \(K^{\Sigma_u, M_G}\), \(s\sigma \in K^{\Sigma_u, M_G}\). 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 \((\Sigma_G^*, M_G)\)-normality of \(K^{\Sigma_u, M_G}\), \(t \in K^{\Sigma_u, M_G}\). 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 (i) \(\sigma \in \Sigma_u\), or (ii) there exists \(t \in pr(K)\) such that \(M_G(t) = M_G(s\sigma)\).

(Otherwise the proper sublanguage of \(K^{\Sigma_u, M_G}\), \(K^{\Sigma_u, M_G} - \{s\sigma\}\Sigma_G^*\), obtained by disabling \(\sigma\) after \(s\) is a prefix-closed, \((\Sigma_G^*, \Sigma_u)\)-controllable and \((\Sigma_G^*, M_G)\)-normal superlanguage of \(K\), a contradiction.) Since we also have \(s\sigma \in L(G)\), \((L(G), \Sigma_u)\)-controllability of \(K\) in case (i) (resp. \((L(G), M_G)\)-normality of \(K\) in case (iii)) 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 \(M_G : \Sigma_G \rightarrow \Sigma_I\). Let \(K \subseteq L(G)\) be a prefix-closed nonempty desired language, \(\Sigma_S\) be the event set of the supervisor, \(B \subseteq \Sigma_S\) its event priority set, and \(M_S : \Sigma_S \rightarrow \Sigma_I\) its priority consistent interface mask. Then there exists a deterministic supervisor \(S\) such that \(L((G_A|_B S)|\Sigma_G) = K\) if and only if \(K\) is \((L(G), \Sigma_u)\)-controllable and \((L(G), M_G)\)-normal, where \(\Sigma_u := A - M_G^{-1}(M_S(B) - \{\epsilon\})\).
Moreover, since (superlanguage of) states are reached by “tracking” the two traces implies (c.f. Definition 1) the two traces have the same successor state, and transitions on unobservable events on any state whenever (c.f. Definition 1) exists (c.f. Definition 1) are self-loops.

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

\[ \mathcal{S} := (X_S, \Sigma_G, \delta^1, x^0, X_S) \]

that generates \( K^{\Sigma_u, M_G} \), the infimal prefix-closed (\( \Sigma^*_G, \Sigma_u \))-controllable and (\( \Sigma_G, M_G \))-normal superlanguage of \( K \). Then as explained in Remark 4 since \( K^{\Sigma_u, M_G} \) is (\( \Sigma^*_G, L(G) \))-normal, \( \mathcal{S} \) 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

\[ S := (X_S, \Sigma_S, \delta_S, x^0_S, X_S) \]

by modifying each transition \( (x, \sigma, x') \in X_S \times \Sigma_S \times X_S \) of \( \mathcal{S} \) as follows:

1. if \( M_G(\sigma) \neq \varepsilon \), then replace it by a transition \( (x, \sigma, x') \in X_S \times \Sigma_S \times X_S \) such that \( M_S(\sigma) = M_G(\sigma) \)

2. if \( M_G(\sigma) = \varepsilon \), then delete this transition

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

\[ L((G \mid_B S)\uparrow \Sigma_G) = L(G) \cap K^{\Sigma_u, M_G}. \quad (2) \]

As before let \( G \mid_B S := (X, \Sigma, \delta, x^0, X^m) \) and \((G \mid_B S)\uparrow \Sigma_G := (X, \Sigma_G, \delta^1, x^0, X^m)\).

We first prove the forward containment in Equation (2). Since \( L((G \mid_B S)\uparrow \Sigma_G) \subseteq L(G) \) it suffices to show that \( L((G \mid_B S)\uparrow \Sigma_G) \subseteq K^{\Sigma_u, M_G} \). First consider a trace \( s \) that \( G \mid_B S \) 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 $s \in L((G \mid (A \mid B \mid S) \mid \Sigma_G))$ that $s \in K_{\Sigma_u, MG}$. Clearly, this holds for length zero since $K_{\Sigma_u, MG}$ is nonempty and prefix-closed, which establishes the base step. For the induction hypothesis let $s = \overline{\sigma} \sigma \overline{g}$. If $\sigma g$ is uncontrollable or unobservable, then by definition of $K_{\Sigma_u, MG}$ and induction hypothesis (which implies $\overline{\sigma} \in K_{\Sigma_u, MG} = L(\overline{S})$), it follows that $s \in K_{\Sigma_u, MG}$. On the other hand if $\sigma g$ 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 $\sigma$ in $\overline{S}$, then $\sigma g$ is defined at $x$, which of course implies that $s = \overline{\sigma} \sigma g \in K_{\Sigma_u, MG}$ (recall that $L(\overline{S}) = K_{\Sigma_u, MG}$). This proves the induction hypothesis.

To complete the proof of the sufficiency part we next need to show the reverse containment in Equation (2). Pick $u_g \in K_{\Sigma_u, MG} \cap L(G) = L(\overline{S}) \cap L(G)$. 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 $x_g \in \delta_G(x_s^0, u_g)$. 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 $(x_g, x_s) \in \delta^1(x_s^0, u_g)$, which implies $u_g \in L((G \mid (A \mid B \mid S) \mid \Sigma_G))$ as desired.

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

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

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

**Remark 5** Lemma 3 provides a modular way of constructing $K_{\Sigma_u, MG}$. 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 $(K_{\Sigma_u})_{MG} = M_G^{-1}M_G(K_{\Sigma_u})$ by replacing the event label $\sigma$ of any transition in the generator of $K_{\Sigma_u}$ 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 $K_{\Sigma_u}$.

**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 Figure 8, when intersected with the generated language of the pumping station imposes the language $K \subseteq L(G)$ as shown in Figure 9. In this figure each state has four components, the first component denotes the state of the synchronizer (Figure 3), the second that of the pump 1 (Figure 3), the third that of the pump 2 (Figure 3), and the last that of the specification (Figure 8).

Since $K$ is prefix-closed and nonempty, from Theorem 3 there exists a deterministic supervisor $S$ such that $L((G_A\|B)\uparrow \Sigma_G) = K$ 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 2 to verify $(L(G), \Sigma_u)$-controllability and $(L(G), M_G)$-normality of $K$. The generator for $K^{\Sigma_u, M_G}$ is shown in Figure 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 Figure 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 Figure 10(b) that enforces $K$ as the projected behavior on the event set $\Sigma_G$ 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 \( \mathcal{S} \) that generates \( K^{\Sigma_u,M_G} \), the infimal prefix-closed \( (\Sigma_G, \Sigma_u) \)-controllable and \( (\Sigma_G, M_G) \)-normal superlanguage of the specification language, 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 \( \mathcal{S} \) 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 \( \mathcal{S} \) 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]).

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 extends the formalism of prioritized synchronous composition (PSC) 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 “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. 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


