Prioritized Synchronization under Mask for Interaction/Control of Partially Observed Discrete Event Systems

Changyan Zhou and Ratnesh Kumar
Department of Electrical & Computer Engineering
Iowa State University, Ames, IA 50014
(czhou, rkumar@iastate.edu)

Abstract
This paper extends the formalism of prioritized synchronous composition (PSC), introduced by Heymann, for modeling interaction/control of discrete event systems to incorporate partial observation. PSC based control helps remove the requirement of control-compatibility of a supervisor. In order to also remove the observation-compatibility requirement of a supervisor, there have been attempts to generalize PSC to account for partial observation. First such attempt was the notion of masked composition (MC), and later the notion of masked-PSC (MPSC) was introduced. Under MPSC the condition for existence of supervisor is normality together with controllability, as opposed to the usual weaker condition of observability together with controllability. The reason being that in the setting of MPSC, observations as well as control are filtered by the interface mask. This motivates the introduction of the notion of prioritized synchronous composition under mask (PSCM). We show that when PSCM is adopted as a mechanism of interaction, not only the control & observation-compatibility requirements are removed of a supervisor, the existence condition is given by achievability that is weaker than controllability and observability combined. (The weaker condition is required since we allow supervisors to be nondeterministic.) This suggests that the notion of PSCM introduced in the paper is an appropriate generalization of PSC to account for partial observation. Both the existence verification and the synthesis of a PSCM-based supervisor is of polynomial complexity.

Keywords: Discrete event systems, supervisory control, prioritized synchronous composition under mask, achievability, partial observation

*The research was supported in part by the National Science Foundation under the grants NSF-ECS-0218207, NSF-ECS-0244732, NSF-EPNES-0323379, and NSF-0424048, and a DoD-EPSCoR grant through the Office of Naval Research under the grant N000140110621.
1 Introduction

Most work on supervisory control of discrete event systems (DESs), such as [20, 19, 13], essentially use strict synchronous composition (SSC) of the plant and supervisor as a mechanism of control. In SSC, it is required that the common events occur synchronously, which is a restriction. For example, there is no a priori reason for a supervisor to synchronously execute all the uncontrollable events that a plant executes.

Heymann [6] proposed a type of interaction, called prioritized synchronous composition (PSC), which relaxed such synchronization requirements. PSC delegates the effects of control limitations of a supervisor from its logic part (implemented as an automaton) to its interface part (implemented as PSC), and thereby, removes the requirement of “completeness” [13] or “$\Sigma$,-compatibility” [12] of a supervisor. In PSC, each system possesses an event priority set specifying a set of events whose execution require its participation. Thus, an event can occur if and only if all the systems having priority over the event can participate in its execution. In this case, the event occurs synchronously in all such systems, and otherwise the event gets blocked from occurring. The systems which do not have priority over the event also participate in its execution if they can, and otherwise the event takes place without the participation of such systems. Thus, a system with no priority over an event cannot block its execution. In other words, at a given state of a system, all executable and all non-priority events are enabled by that system at that state. A key difference between PSC and other types of concurrent compositions such as those in [9, 10] is that in PSC, although a process cannot block events which are outside its priority set, it is able to execute these events synchronously whenever possible. Supervisory control of DESs using PSC has been studied in [8, 1, 4, 7, 21, 16, 18, 17].

PSC models the interaction among systems when all the events are completely observable. When systems interact under partial observation (modeled as non-identity observation masks), their interaction through PSC requires that the systems be observation compatible with respect to their masks [18]. So it is meaningful to generalize the notion of PSC to allow interaction of systems possessing non-identity observation mask so that the systems can interact without needing to ensure that they are control or observation compatible.

An effort to generalize PSC in such a direction was presented in [22], and the generalization was called masked composition (MC). The notion of process-objects was introduced and their MC was defined. In MC, each system is associated with two types of masks: A control mask that identifies events from the control perspective, and an observation mask that identifies events from the observation perspective. One difficulty with that work is the underlying modeling formalism of process objects that contains “virtual transitions” besides “real” ones, and modeling of practical systems as such process objects is not quite clear.

Another generalization of PSC to describe prioritized synchronization of systems via interfaces, masked prioritized synchronous composition (MPSC), was introduced by Kumar-Heymann in [14]. [11] later used MPSC for control with “driven” or “command” or “forcible” events [5, 6, 2, 3]. 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. In MPSC, each system is allowed to interact with its environment via interfaces that are modeled as event mask functions. When two or more systems interact at a common interface, they can synchronize on events that are mapped to common interface events.

MPSC is appropriate for systems interacting via common interfaces. When MPSC is employed for control the condition for existence of a supervisor is \textit{normality together with controllability}, as opposed to the usual weaker condition of \textit{observability together with controllability}. This suggests that MPSC imposes certain stringent interface constraints. In MPSC, not only the observations but also the controls are filtered through the interface mask - An event is enabled as long as an indistinguishable event is enabled. This makes the control more restrictive than that of the usual supervisory control setting where control and observation of events are not inter-dependent. This is the reason for the stronger condition of normality required of a control specification. This serves as a motivation to introduce the notion of prioritized synchronous composition under mask (PSCM) in this paper.

In PSCM, each system has its own event priority set. At any given state, a system enables all priority events that it can execute at that state, together with all non-priority events. An event is enabled in the composition (i.e., globally enabled) if and only if it is enabled by all interacting systems (i.e., locally enabled by all). A globally enabled event that is executable by one of the systems, can occur in the composed system upon “initiation” by a system that can execute it. Then other systems track it by executing an observation indistinguishable event. If the event is unobservable to or if no observationally indistinguishable events are defined in one of the systems, then that system does not participate in tracking. The transition in the composed system is labeled by the initiating event (and not by the tracking event).

We established a link between PSCM and PSC (and thereby SSC) under certain constraints on the priority sets and the mask functions: PSCM of two systems can be alternatively obtained by first augmenting individual systems, and next computing the PSC of the augmented systems. For this to work, the priority sets of the two systems must exhaust the entire event set, and each event must be observable to a system having priority over that event. These constraints are naturally met in supervisory control where a plant has priority over each event and can also “observe” each event. Note while MPSC is always convertible to PSC (through unmasking of the PSC of masked systems), this is not the case with PSCM, showing its generality over PSC. Similarly, while PSC and MPSC are known to be associative, this is not the case with PSCM as is shown by an example.

Through the introduction of PSCM we are able to relax the restriction on control that MPSC imposes. We show that when PSCM is adopted as a mechanism of interaction, not only the control & observation-compatibility requirements are removed of a supervisor, the existence condition is given by \textit{achievability} \cite{15} that is weaker than controllability and observability combined. (The weaker condition is required since we allow supervisors to be nondeterministic, whereas the conditions of controllability and observability combined are required for the existence of a deterministic supervisor.) This suggests that the notion of PSCM, presented in the paper, is an appropriate generalization of PSC to account for
partial observations. Also, both the existence verification and the synthesis of a PSCM-based supervisor can be performed polynomially: The existence verification is linear in the plant size and quadratic in the specification size, whereas the synthesis is linear in the specification size. The results on PSCM-based control presented in the paper have benefited from the past work of our group [15] that laid the foundation of nondeterministic control and introduced the notion of *achievability* as a condition for existence of a nondeterministic supervisor under partial observation.

The rest of the paper is organized as follows. Section 2 gives notation and preliminaries. Sections 3 introduces the concept of prioritized synchronous composition under mask and studies its properties. Section 4 introduces augmentation in the setting of PSCM, and show how in certain cases, PSCM of systems is equivalent to PSC of their augmentations. Section 5 studies PSCM-based supervisory control with no “driven” events. Section 6 presents an illustrative example. Section 7 concludes the paper.

## 2 Notations and Preliminaries

In this paper nondeterministic state machines (NSMs) are used to model discrete event systems. A NSM $G$ is a four tuple: $G := (X, \Sigma, \alpha, X_0)$, where $X$ is its set of states, $\Sigma$ is its set of events, $\alpha : X \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^X$ is its state transition function, and $X_0 \subseteq X$ is its set of initial states. For an event set $\Sigma$, we use $\overline{\Sigma}$ to denote $\Sigma \cup \{\epsilon\}$. A triple $(x, \sigma, x') \in X \times \overline{\Sigma} \times X$ is called a *transition* if $x' \in \alpha(x, \sigma)$; if $\sigma = \epsilon$, the transition is called an $\epsilon$-transition.

Given an event set $\Sigma$, $\Sigma^*$ denotes the set of all finite-length sequences of events from $\Sigma$, including the trace of zero length, denoted $\epsilon$. For $x \in X$, we use $\Sigma(x) := \{\sigma \in \overline{\Sigma} | \alpha(x, \sigma) \neq \emptyset\}$ to denote the set of labels in $\Sigma$ defined at $x$. For $x \in X$, the $\epsilon$-*closure* of $x$, denoted $\epsilon^*(x)$, is the set of states reached by the execution of zero or more $\epsilon$-transitions from state $x$, and is defined recursively as:

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

The $\epsilon$-closure map can be used to extend the definition of transition function from events to traces. This yields $\alpha : X \times \Sigma^* \rightarrow 2^X$, which is defined inductively as:

$$\forall x \in X : [\alpha(x, \epsilon) := \epsilon^*(x) ; \forall s \in \Sigma^*, \sigma \in \overline{\Sigma} : \alpha(x, s\sigma) := \epsilon^*(\alpha(x, s), \sigma)].$$

The *generated* languages of $G$, denoted $L(G)$, is defined as $L(G) := \{s \in \Sigma^* | \alpha(X_0, s) \neq \emptyset\}$. Letting $pr(\cdot)$, denote the prefix closure operation, $L(G) = pr(L(G))$.

One way to model control interaction between plant and supervisor is via the *strict synchronous composition (SSC)* of their state machine representations. The SSC of two state machines $G_1 := (X_1, \Sigma, \alpha_1, X_{01})$ is the NSM $G_1 \parallel G_2 := (X_1 \times X_2, \Sigma, \alpha, X_{01} \times X_{02})$, where for $x_1 \in X_1$, $x_2 \in X_2$, and $\sigma \in \Sigma$:

$$\alpha((x_1, x_2), \sigma) := \begin{cases} \alpha_1(x_1, \sigma) \times \alpha_2(x_2, \sigma) & \text{if } \sigma \in \Sigma \\ (\alpha_1(x_1, \epsilon) \times \{x_2\}) \cup (\{x_1\} \times \alpha_2(x_2, \epsilon)) & \text{if } \sigma = \epsilon \end{cases}$$
It is easy to see that $L(G_1 \| G_2) = L(G_1) \cap L(G_2)$.

In order to relax the strict synchronization requirement of SSC, Heymann [6] proposed prioritized synchronous composition (PSC). In PSC, each system has an event priority set. An event can occur as long as all systems having the priority over the event can participate. For $i = 1, 2$, consider NSM $G_i = (X_i, \Sigma, \alpha_i, X_{0i})$ with its event priority set $A_i \subseteq \Sigma$. Then the prioritized synchronous composition of $G_1$ and $G_2$ is given by

$$G_{1A_1 \| A_2} G_2 = (X_1 \times X_2, \Sigma, \alpha, X_{01} \times X_{02}),$$

where for $x_1 \in X_1, x_2 \in X_2$ and $\sigma \in \Sigma$,

$$\alpha((x_1, x_2), \sigma) := \begin{cases} 
\alpha_1(x_1, \sigma) \times \alpha_2(x_2, \sigma), & \text{if } \alpha_1(x_1, \sigma) \neq \emptyset, \alpha_2(x_2, \sigma) \neq \emptyset \\
\alpha_1(x_1, \sigma) \times \{x_2\}, & \text{if } \alpha_1(x_1, \sigma) \neq \emptyset, \alpha_2(x_2, \sigma) = \emptyset, \sigma \not\in A_2 \\
\{x_1\} \times \alpha_2(x_2, \sigma), & \text{if } \alpha_2(x_2, \sigma) \neq \emptyset, \alpha_1(x_1, \sigma) = \emptyset, \sigma \not\in A_1 \\
\emptyset, & \text{otherwise}
\end{cases}$$

$$\alpha((x_1, x_2), \epsilon) := (\alpha_1(x_1, \epsilon) \times \{x_2\}) \cup (\{x_1\} \times \alpha_2(x_2, \epsilon)).$$

The event priority set of $G_{1A_1 \| A_2} G_2$ is taken to be $A_1 \cup A_2$. In the special case when the event priority sets of the two systems exhaust the entire event set $\Sigma$, PSC can be transformed to SSC using the method of augmentation introduced in [6].

The events executed by a system are partially observed by other systems owing to the particular event-sensors used. Such a partial observation induces a partition of $\overline{\Sigma}$, with each partition representing a set of observation indistinguishable events. For $\sigma \in \Sigma$, we use $M(\sigma) \subseteq \Sigma$ to denote the set of $\sigma$-indistinguishable events. $\sigma \in \Sigma$ is said to be unobservable if $\sigma \in M(\epsilon)$; $\sigma$ is said to be completely observable if $M(\sigma) = \{\sigma\}$. The set of unobservable events is $M(\epsilon)$, and the set of completely observable events is denoted $\Sigma_o$.

The following definition introduces the notion of control compatibility (requiring uncontrollable events be never disabled), and observation compatibility (requiring state updates on indistinguishable events be identical).

**Definition 1** [15] Let $S = (Y, \Sigma, \beta, Y_0)$ be a nondeterministic state machine, $\Sigma_u \subseteq \Sigma$ be the set of uncontrollable events, and $M$ be the observation mask. Then

1. $S$ is called $\Sigma_u$-compatible if $\forall y \in Y$ and $\forall a \in \Sigma_u$, $\beta(y, a) \neq \emptyset$.
2. $S$ is called $M$-compatible if $\forall y \in Y$ and $\forall a, b \in \Sigma(y)$, if $M(a) = M(b)$, then $\beta(y, a) = \beta(y, b)$.
3. $S$ is called $(\Sigma_u, M)$-compatible if $S$ is $\Sigma_u$-compatible and $M$-compatible.

### 3 Prioritized Synchronization under Mask

In this section, we formalize the notion of prioritized synchronous composition under mask (PSCM) and study its properties. PSCM generalizes the prioritized synchronization
of systems to incorporate partial observation. In PSCM, each system possesses an event priority set and an observation mask. The observation mask is not required to be priority-consistent as in MPSC. In [14], it was shown that MPSC can alternatively be computed by “unmasking” the PSC of “masked” systems, thereby established a link between MPSC and PSC. In this section, we show that when certain constraints are imposed on the priority sets and observation masks, the PSCM of two systems can alternatively be obtained by first “augmenting” each of the systems, and next computing the PSC of augmented systems. Note that although MPSC can always be transformed to PSC via “pre-masking” and “post-unmasking”, PSCM can only be transformed to PSC under certain restrictions, showing the generality of PSCM over PSC.

In PSCM, an event is “locally” enabled at a certain state of a system if it is executable at that state or is a non-priority event. An event is enabled in the composition (i.e., “globally” enabled) if and only if it is enabled by all interacting systems (i.e., locally enabled by all). A globally enabled event that is executable by one of the systems, can occur in the composed system upon “initiation” by a system that can execute it. Then other systems track it by executing an observation indistinguishable event. If the event is unobservable to or if no observationally indistinguishable events are defined in one of the systems, then that system does not participate in tracking. The transition in the composed system is labeled by the initiating event (and not by the tracking event).

Since any executable event is automatically enabled, and since a system cannot block events outside its priority set (meaning they always remain enabled) or an $\epsilon$-transition, the set of enabled “events” at a state $x_i$ of $G_i$ is $\Sigma(x_i) \cup A_i^e \cup \{\epsilon\}$, the set of executable events at $x_i$ together with the non-priority events and all $\epsilon$-transitions. We denote this as,

\[ \Sigma_e(x_i) := \Sigma(x_i) \cup A_i^e \cup \{\epsilon\}. \]

An event is enabled at a state $(x_1, \ldots, x_n)$ of PSCM composed systems $\{G_i, i \leq n\}$ if and only if it is enabled at state $x_i$ of $G_i$. In other words, the set of enabled events at state $(x_1, \ldots, x_n)$ of the composition is given by,

\[ \Sigma_e((x_1, \ldots, x_n)) := \bigcap_{i=1}^{n} \Sigma_e(x_i) = (\bigcap_{i=1}^{n} \Sigma(x_i) \cup A_i^e) \cup \{\epsilon\}. \]

Next we formally define the notion of PSCM.

**Definition 2** For $i = 1, 2$, consider system $G_i = (X_i, \Sigma, \alpha_i, X_{0_i})$, possessing event priority set $A_i$, and observation mask $M_i$. Then the *prioritized synchronous composition under mask (PSCM)* of $G_1$ and $G_2$ is given by

\[ G_1 A_1 \parallel A_2 G_2 = (X_1 \times X_2, \Sigma, \alpha, X_{01} \times X_{02}), \]

where for $x_1 \in X_1, x_2 \in X_2$ and $\sigma \in \Sigma$,
\[
\begin{align*}
\alpha(x_1, x_2, \sigma) & := \\
\alpha_1(x_1, \sigma) \times \alpha_2(x_2, \sigma'), & \text{if } \begin{cases} 
\sigma \in \Sigma_e((x_1, x_2)), \\
\alpha_1(x_1, \sigma) \neq \emptyset, \\
\alpha_2(x_2, \sigma') \neq \emptyset, \\
\sigma' \in M_2(\sigma) \neq M_2(\epsilon) \end{cases} \\
\alpha_1(x_1, \sigma') \times \alpha_2(x_2, \sigma), & \text{if } \begin{cases} 
\sigma \in \Sigma_e((x_1, x_2)), \\
\alpha_1(x_1, \sigma') \neq \emptyset, \\
\alpha_2(x_2, \sigma) \neq \emptyset, \\
\sigma' \in M_1(\sigma) \neq M_1(\epsilon) \end{cases} \\
\{x_1\} \times \alpha_2(x_2, \sigma), & \text{if } \begin{cases} 
\sigma \in \Sigma_e((x_1, x_2)), \\
\alpha_2(x_2, \sigma) \neq \emptyset, \\
\epsilon \in M_1(\sigma) \lor \alpha_1(x_1, M_1(\sigma)) = \emptyset \end{cases} \\
\emptyset, & \text{otherwise}
\end{align*}
\]

\[
\alpha((x_1, x_2), \epsilon) := (\alpha_1(x_1, M_1(\epsilon)) \times \{x_2\}) \cup (\{x_1\} \times \alpha_2(x_2, M_2(\epsilon))).
\]

Note in all clauses, the executable event \(\sigma\) is also enabled in the composition (\(\sigma \in \Sigma_e((x_1, x_2))\)). In the first clause, \(\sigma\) is executable at \(x_1\) (\(\alpha_1(x_1, \sigma) \neq \emptyset\)). \(G_1\) initiates \(\sigma\) by transitioning to a state in \(\alpha_1(x_1, \sigma)\), and \(G_2\) tracks by executing an \(M_2\)-indistinguishable event \(\sigma' \in M_2(\sigma) \neq M_2(\epsilon)\) that is defined at state \(x_2\) (\(\alpha_2(x_2, \sigma') \neq \emptyset\)). The second clause is similar to the first clause, except here \(G_2\) initiates \(\sigma\) and \(G_1\) tracks by executing \(\sigma' \in M_1(\sigma) \neq M_1(\epsilon)\). In the third clause, \(\sigma\) is executable in \(G_1\) and either it is unobservable to \(G_2\) (\(\sigma \in M_2(\epsilon)\)) or there is no \(M_2\)-indistinguishable event defined at state \(x_2\) (\(\alpha_2(x_2, M_2(\sigma)) \neq \emptyset\)). So, \(\sigma\) occurs asynchronously in \(G_1\). (Note that \(G_2\) does not block since \(\sigma\) is enabled by both \(G_1\) and \(G_2\) by virtue of its membership in \(\Sigma_e((x_1, x_2)) = \Sigma_e(x_1) \cap \Sigma_e(x_2)\).) The fourth clause can be understood in a similar way as the third clause. Finally, an \(\epsilon\)-transition in the composition corresponds to an asynchronous execution of a label in \(M_i(\epsilon), i = 1, 2\), in which case only \(G_i\) participates.

The event priority set of \(G_{1A1}^{M_1} \parallel_{A2}^{M_2} G_2\) can be taken to be \(A := A_1 \cup A_2\), whereas the class of observationally indistinguishable events for an event \(\sigma\) in \(G_{1A1}^{M_1} \parallel_{A2}^{M_2} G_2\) is given by \(M(\sigma) := M_1(\sigma) \cap M_2(\sigma)\).

**Remark 1** In the special case when both systems have identity mask (\(Id\)) functions, the PSCM reduces to the PSC. I.e., \(G_{1A1}^{Id} \parallel_{A2}^{Id} G_2 = G_{1A1} \parallel_{A2} G_2\). To see this, when \(M_1 = M_2 = Id\), \(M_1(\sigma) = M_2(\sigma) = \{\sigma\}\) for all \(\sigma \in \Sigma\), the transition function of Definition 2 simplifies to:
Consider clause 1. Then $\alpha_i(x_1, \sigma) \neq \emptyset \iff \sigma \in \Sigma(x_i) \iff \sigma \in \Sigma(x_1) \cap \Sigma(x_2) \subseteq \Sigma_e((x_1, x_2))$.

So the condition of clause 1 is equivalent to $\alpha_i(x_1, \sigma) \neq \emptyset$ as in clause 1 of definition of PSC.

Next consider clause 2. Then $\alpha_1(x_1, \sigma) \neq \emptyset$, $\alpha_2(x_2, \sigma) = \emptyset \Rightarrow \alpha \in \Sigma(x_1) - \Sigma(x_2)$. So for $\sigma \in \Sigma_e((x_1, x_2))$ to hold, $\sigma \in A_2$. So the condition of clause 2 is equivalent to $\alpha_1(x_1, \sigma) \neq \emptyset$, $\alpha_2(x_2, \sigma) = \emptyset$, $\sigma \not\in A_2$ as in clause 2 of definition of PSC. Clause 3 can be analyzed similar to clause 2.

The following example illustrates the concept of PSCM.

**Example 1** Consider $G_1$ and $G_2$ shown in Figure 1, with

$$A_1 = \{a\}, M_1(a) = M_1(b) = \{a, b\}, M_1(c) = \{e, c\}, M_1(d) = \{d\};$$
$$A_2 = \{b\}, M_2(a) = M_2(d) = \{a, d\}, M_2(b) = \{b\}, M_2(c) = \{c\}.$$

$G_{A_1}^{M_1} \parallel G_{A_2}^{M_2}$ is drawn in Figure 3, where for simplicity a state $(x_1, x_2)$ of the composition is written as $x_1x_2$. At state 1A,

$$\Sigma_e(1A) = [(a, b) \cup \{b, c, d\}] \cap [(a, d) \cup \{a, c, d\}] = \{a, c, d\}.$$  

Since for $a \in \Sigma_e(1A)$, $\alpha_1(1, a) = \{2\}$, $a, d \in M_2(a)$, $\alpha_2(A, a) = \{B\}$, and $\alpha_2(A, d) = \{C\}$, by clause 1, we have the transitions $(1A, a, 2B)$ and $(1A, a, 2C)$ in $G_{A_1}^{M_1} \parallel G_{A_2}^{M_2}$. Similarly, for $a \in \Sigma_e(1A)$, $\alpha_2(A, a) = \{B\}$, $a, b \in M_1(a)$, $\alpha_1(1, a) = \{2\}$, and $\alpha_1(1, b) = \{3\}$. By clause 2,
we have the transitions \((1A, a, 2B)\) and \((1A, a, 3B)\) in \(G_{1A_1||A_2}^{M_1||M_2}G_2\). Similarly, for \(d \in \Sigma_e(1A)\), \(\alpha_2(A, d) = \{C\}\), \(d \in M_1(1d)\) and \(\alpha_1(1, d) = \emptyset\). By clause 4, we have the transition \((1A, d, 1C)\) in \(G_{1A_1||A_2}^{M_1||M_2}G_2\). Note that for \(c \in \Sigma_e(1A)\), \(\alpha_1(1, c) = \emptyset\) and \(\alpha_2(A, c) = \emptyset\). Thus, transition on \(c\) at state \(1A\) in \(G_{1A_1||A_2}^{M_1||M_2}G_2\) does not exist.

At state \(2B\),

\[
\Sigma_e(2B) = \{(c) \cup \{b, c, d\}\} \cap \emptyset = \{c, d\}.
\]

Since for \(c \in \Sigma_e(2B)\), \(\alpha_1(2, c) = \{3\}\), \(c \in M_2(c)\) and \(\alpha_2(B, c) = \emptyset\). By clause 3, we have the transition \((2B, c, 3B)\) in \(G_{1A_1||A_2}^{M_1||M_2}G_2\). Also, since \(c \in M_1(c)\), by clause 5, transition \((2B, \epsilon, 3B)\) is defined in \(G_{1A_1||A_2}^{M_1||M_2}G_2\).

When the priority set of the composition is taken to be the union of the two priority sets, and the event-indistinguishability partition is taken to be the intersection of the two event-indistinguishability partitions, then the property of associativity may not be preserved under PSCM. This is in contrast to the operations of PSC and MPSC (both are known to be associative), showing again the generality of PSCM.

**Remark 2** Consider \(G_1\), \(G_2\), and \(G_3\) drawn in Figure 2, with

\[
A_1 = \{a\}, M_1(a) = M_1(b) = \{a, b\}, M_1(c) = \{c\};
\]

\[
A_2 = \{b\}, M_2(a) = M_2(c) = \{a, c\}, M_2(b) = \{b\};
\]

\[
A_3 = \{c\}, M_3(b) = M_3(c) = \{b, c\}, M_3(a) = \{a\}.
\]

The state machines \((G_{1A_1||A_2}^{M_1||M_2}G_2)^{M_1\cap M_2}G_3\) and \((G_{1A_1||A_2}^{M_1||M_2\cap M_3}G_2A_2^{M_2\cap M_3}A_3)\) are also drawn in Figure 2 (details are omitted), from which the non-associativity of PSCM is clear.

As discussed below under certain assumptions, PSCM can be converted to PSC, in which case the property of associativity of PSC carries over to that of PSCM.

### 4 Augmentation for Conversion to PSC/SSC

The main feature of PSCM (when compared to PSC) is that execution of an event enabled in the composition by a system can be tracked by another system by synchronously executing an indistinguishable event. We call such synchronous execution \(M\)-synchronous executions, or simply \(M\)-synchronizations. One purpose of augmentation is to introduce new transitions that let such \(M\)-synchronizations be computed as ordinary synchronizations, where the synchronizing transitions carry the same label. Another purpose is to also capture asynchronous executions also as ordinary synchronous executions by introducing appropriate self-loop transitions in systems that are non-participants, and for unobservable events appropriate \(\epsilon\)-transitions in the systems where unobservable events are executable.

It is clear that augmentation in \(G_j\) for an asynchronous execution of \(G_i\) will be a self-loop, whereas the augmentation for a transition in \(G_i\) that \(G_j\) tracks, will be along side the tracking transition, and also augmentation on \(\epsilon\)-transition in \(G_j\) will be along side an unobservable
event transition executable in $G_j$. Care must be taken so that it is always the case that an augmented transition of $G_j$ synchronizes with an existing transition of $G_i$, i.e., an augmented transition of $G_j$ should not synchronize with an augmented transition of $G_i$, since such a transition is not possible in the original composition.

Letting $\text{Aug}_i(x_i) \subseteq \Sigma$ denote the set of labels in $\Sigma$ with which state $x_i$ in system $G_i$ can be augmented, then so as they are not stray transitions introduced by augmentation one constraint the set of augmented events should satisfy is that they be locally enabled, i.e.,

$$\text{Aug}_i(x_i) \cap \Sigma \subseteq \Sigma_e(x_i) \cap \Sigma = [\Sigma(x_i) \cup A_i^e]$$

$$= [\Sigma(x_i) \cap A_i] \cup [\Sigma - A_i]$$

$$= [\Sigma(x_i) \cap A_i] \cup [A - A_i] \cup [\Sigma - A],$$

where $A := \bigcup_i A_i$ denotes the set of all priority events. So the above provides an upper bound for $\text{Aug}_i(x_i)$.

Consider first an event $\sigma$ in the local priority set $A_i$. As mentioned earlier, for $\sigma$ to be a candidate for augmentation at $x_i$ (i.e., $\sigma \in \text{Aug}(x_i)$), $\sigma$ must be locally enabled. Since $\sigma \in A_i$, this means $\sigma$ must be locally executable (i.e., $\sigma \in \Sigma_i(x_i)$). Now if $\sigma$ is locally executable, an augmentation on $\sigma$ is not required if it is completely observable, i.e., if $\sigma \in \Sigma_{oi}$. Thus

If $\sigma \in \text{Aug}_i(x_i) \cap A_i$, then it should hold that $\sigma \in [A_i \cap \Sigma(x_i) - \Sigma_{oi}]$.

Now if $\sigma \in A_i \cap \Sigma(x_i) - \Sigma_{oi}$ so that an augmentation on $\sigma$ is performed at $x_i$, then for this transition to not synchronize with another augmented transition in another system, it must
be the case that there exists at least one system where no augmentation on \( \sigma \) is performed. This is guaranteed by having \( j \) such that \( \sigma \in A_j \cap \Sigma_{oj} \). Thus

\[
\text{If } \sigma \in \text{Aug}_i(x_i) \cap A_i, \text{ then it should hold that } \exists j, \sigma \in [\Sigma(x_i) - \Sigma_{oi}] \cap [A_j \cap \Sigma_{oj}]. \tag{1}
\]

Next consider an event \( \sigma \) not in local priority set \( A_i \), but in priority set for some system \((\sigma \in A - A_i)\). Then it is locally enabled and so a candidate for augmentation. Also since \( \sigma \) is in priority set of another system \( G_j \), it is guaranteed that \( \sigma \) is globally executable only if it is locally executable in \( G_j \) (i.e., \( \sigma \in \Sigma(x_j) \)). In order to avoid synchronization of augmented transition in one system by augmented transition in another system, it must be the case that no augmentation is performed on the same event in at least one system. This is guaranteed by further having \( \sigma \in \Sigma_{oj} \). Thus

\[
\text{If } \sigma \in \text{Aug}_i(x_i) \cap [A - A_i], \text{ then it should hold that } \exists j, \sigma \in [A_j \cap \Sigma_{oj}]. \tag{2}
\]

Finally consider an event \( \sigma \) in priority set of none of the systems \((\sigma \in \Sigma - A)\). Then \( \sigma \) is locally enabled in every system, and so a candidate for augmentation in each system. However, it is possible that \( \sigma \) is not locally executable in any one of them, i.e., it is possible that \( \sigma \in \Sigma_e((x_1, \cdots, x_n)) - \cup_i \Sigma(x_i) \). Then augmentation on \( \sigma \) will allow synchronization of transition that are augmented transitions in all systems. This suggests that augmentation cannot be performed on events in \( \Sigma - A \). In other words,

\[
\sigma \in \text{Aug}(x_i) \cap [\Sigma - A] \text{ should be impossible.} \tag{3}
\]

Combining 1, 2, and 3 it can be obtain that,

\[
\text{If } \sigma \in \text{Aug}_i(x_i) \cap \Sigma, \text{ then it should hold that } \exists j, \sigma \in [(A_i \cap \Sigma(x_i) - \Sigma_{oi}) \cup (A - A_i)] \cap [A_j \cap \Sigma_{oj}]. \tag{4}
\]

In other words, in order to perform augmentation on \( \sigma \in \Sigma \) at state \( x_i \) of \( G_i \), either (i) \( \sigma \) is a priority event of \( G_i \), is executable at \( x_i \), and is not completely observable by \( G_i \), or (ii) \( \sigma \) is a non-priority event of \( G_i \) but a priority event of some system. In either case there must exist a system for which \( \sigma \) is a priority event and is completely observable. Allowing for the possibility of augmentation on any priority event, we make the following requirement an assumption.

**Assumption 1** For \( i \leq n \), consider NSM \( G_i \) possessing event priority set \( A_i \) and observation mask \( M_i \). Suppose \( \forall \sigma \in A = \cup_i A_i, \exists j \text{ such that } \sigma \in A_j \cap \Sigma_{oj} \).

Under Assumption 1, all events in \( A = \cup_i A_i \) are candidates for augmentation. Moreover the label \( \epsilon \) can be used for augmentation whenever an unobservable event is locally defined.

**Algorithm 1** For \( i \leq n \), consider NSM \( G_i \) possessing event priority set \( A_i \) and observation mask \( M_i \). The following algorithm computes augmented \( G_i \), \( G_i^{\text{Aug}} := (X_i, \Sigma, \alpha_i^{\text{Aug}}, X_{0i}) \).
1. For each state $x_i$ of $G_i$,

$$\text{Aug}_i(x_i) := \begin{cases} 
[A_i \cap \Sigma(x_i) - \Sigma_{o1}] \cup [A - A_i] & \text{if } \Sigma(x_i) \cap A \cap M_i(\varepsilon) = \emptyset \\
[A_i \cap \Sigma(x_i) - \Sigma_{o1}] \cup [A - A_i] \cup \{\varepsilon\} & \text{otherwise}
\end{cases}$$

2. For $\sigma \in \text{Aug}_i(x_i) - M_i(\varepsilon)$, if $\alpha_i(x_i, M_i(\sigma)) \neq \emptyset$, add transitions on $\sigma$ from $x_i$ to all states in the set $\alpha_i(x_i, M_i(\sigma))$, otherwise add self-loop on $\sigma$ at $x_i$;

3. For $\sigma \in \text{Aug}_i(x_i) \cap M_i(\varepsilon)$, add self-loop on $\sigma$ at $x_i$. Further if $\alpha_i(x_i, \sigma) \neq \emptyset$, add transitions on $\varepsilon$ from $x_i$ to all states in this set.

In other words, for augmentation on observable $\sigma$, we add $\sigma$-transitions along side all existing transitions on $\sigma$-indistinguishable events, and if none such transitions exist, we simply add a self-loop on $\sigma$. On the other hand (when $\sigma$ is unobservable), we augment by adding a self-loop on $\sigma$, together with $\varepsilon$-transitions along side any existing $\sigma$-transitions.

The following example illustrates Algorithm 1.

**Example 2** Consider $G_1$ and $G_2$ shown in Figure 3, with

$$A_1 = \{d, e\}, M_1(a) = M_1(b) = \{a, b\}, M_1(c) = \{c\}, M_1(d) = \{d\}, M_1(\varepsilon) = \{\varepsilon\};$$

$$A_2 = \{a, c\}, M_2(a) = \{a\}, M_2(c) = \{c\}, M_2(b) = M_2(d) = M_2(\varepsilon) = \{\varepsilon, b, d, e\}.$$ 

![Figure 3: G_1 (first), G_2 (second), G_1^{Aug_1} (third), G_2^{Aug_2} (fourth) and G_1 M_1 \parallel G_2 (fifth)]](image)

Then $A = A_1 \cup A_2 = \{a, c, d, e\}, \Sigma_{o1} = \{c, d, e\}$ and $\Sigma_{o2} = \{a, c\}$. For state $1$ in $G_1$, since

$$\Sigma(1) \cap A \cap M_1(\varepsilon) = \{a, b\} \cap \{a, c, d, e\} \cap \emptyset = \emptyset,$$

$$\text{Aug}_1(1) = [A_1 \cap \Sigma(1) - \Sigma_{o1}] \cup [A - A_1] = \{(d, e) \cap \{a, b\} \cap \{c, d, e\}\} \cup \{(a, c, d, e) \cap \{d, e\}\} = \{a, c\}.$$ 

For $a \in \text{Aug}_1(1)$, since $a \not\in M_1(\varepsilon)$ and $\alpha_1(1, M_1(a)) = \alpha_1(1, \{a, b\}) = \{2, 3\}$, by step 2 of Algorithm 1, transition $(1, a, 3)$ is added in $G_1$ for augmentation. For $c \in \text{Aug}_1(1)$, since
c \not\in M_1(e) and \alpha_1(1, M_1(c)) = \emptyset, by step 2 of Algorithm 1, transition (1, c, 1) is added in \( G_1 \) for augmentation. Similarly, one can compute \( \text{Aug}_1(2) = \{a, c\} = \text{Aug}_1(3) \). The state machine \( G_1^{\text{Aug}_1} \) is drawn in Figure 3.

For state \( A \) in \( G_2 \), since

\[
\Sigma(A) \cap A \cap M_2(e) = \{a, e\} \cap \{a, c, d, e\} \cap \{\epsilon, b, d, e\} = \{\epsilon\} \neq \emptyset,
\]

\[
\text{Aug}_2(A) = [A_2 \cap \Sigma(A) - \Sigma_{\text{o2}}] \cup [A - A_2] \cup \{\epsilon\} = [(\{a, c\} \cap \{a, e\} - \{a, c\}] \cup [\{a, c, d, e\} - \{a, c\}] \cup \{\epsilon\} = \{d, e, \epsilon\}.
\]

For \( d \in \text{Aug}_2(A) \), since \( d \in M_2(e) \) and \( \alpha_2(A, d) = \emptyset \), by step 3 of Algorithm 1, transition \((A, d, A)\) is added in \( G_2 \) for augmentation. For \( e \in \text{Aug}_2(A) \), since \( e \in M_2(e) \) and \( \alpha_2(A, e) = \{C\} \), by step 3 of Algorithm 1, transitions \((A, e, A)\) and \((A, e, C)\) are added in \( G_2 \) for augmentation. Similarly, one can compute \( \text{Aug}_2(B) = \{d, e\} = \text{Aug}_1(C) \). The state machine \( G_2^{\text{Aug}_2} \) is drawn in Figure 3. Finally, the state machine \( G_1^{M_1} \parallel G_2^{M_2} \) is also drawn in Figure 3.

Since we do not augment with events in \([\Sigma - A]\), through the augmentation we are able to simulate only those asynchronous or \( M \)-synchronous executions as synchronous executions that occur on events outside \([\Sigma - A]\), i.e., on events in \( A \). So after the augmentation, the priority sets of both the systems can only be enlarged to the set \( A \), where all events will occur synchronously. This is summarized in the following theorem.

**Theorem 1** Under the Assumption 1, \( G_1^{M_1} \parallel G_2^{M_2} = G_1^{\text{Aug}_1 M_1} \parallel G_2^{\text{Aug}_2 M_2} \).

**Proof:** Since the state sets, the event sets, and the initial states of the two NSM’s are all identical, we only need to show that they also have the identical set of transitions. For notational convenience, the set of events defined (resp., enabled) at a state \( x_i \) of \( G_i^{\text{Aug}_i} \) is denoted by \( \Sigma_{\text{Aug}}(x_i) \) (resp., \( \Sigma_{\text{Aug}, e}(x_i) \)). Since the priority set of \( G_i^{\text{Aug}_i} \) is \( A \), it follows that

\[
\Sigma_{\text{Aug}, e}(x_i) = \Sigma_{\text{Aug}}(x_i) \cup A^c \cup \{\epsilon\}.
\]

By Algorithm 1, events defined at \( x_i \) of \( G_i^{\text{Aug}_i} \) are given by,

\[
\begin{align*}
\Sigma_{\text{Aug}}(x_i) \\
= \Sigma(x_i) \cup \text{Aug}_i(x_i) \\
= \begin{cases} 
\Sigma(x_i) \cup [A \cap \Sigma(x_i) - \Sigma_{\text{o1}}] \cup [A - A_1] & \text{if } \Sigma(x_i) \cap A \cap M(\epsilon) = \emptyset \\
\Sigma(x_i) \cup [A \cap \Sigma(x_i) - \Sigma_{\text{o1}}] \cup [A - A_1] \cup \{\epsilon\} & \text{otherwise}
\end{cases}
\end{align*}
\]

\( (a) \)

\[
\begin{align*}
\Sigma(x_i) \cup [A - A_1] & \text{ if } \Sigma(x_i) \cap A \cap M(\epsilon) = \emptyset \\
\Sigma(x_i) \cup [A - A_1] \cup \{\epsilon\} & \text{ otherwise}
\end{align*}
\]
Equality (a) holds since $A_i \cap \Sigma(x_i) \subseteq \Sigma(x_i)$, which implies $A_i \cup \Sigma(x_i) - \Sigma_{o2} \subseteq \Sigma(x_i)$. So, we have

$$\begin{align*}
\Sigma_{Aug,e}((x_1, x_2)) \\
= \Sigma_{Aug,e}(x_1) \cap \Sigma_{Aug,e}(x_2) \\
= ([\Sigma_{Aug}(x_1) \cup A_i^c] \cup \{\epsilon\}) \cap ([\Sigma_{Aug}(x_2) \cup A_i^c] \cup \{\epsilon\}) \\
= ([\Sigma(x_1) \cup [A - A_i] \cup A_i^c \cap [\Sigma(x_2) \cup [A - A_2] \cup A_i^c]) \cup \{\epsilon\} \\
= ([\Sigma(x_1) \cup A_i^c \cap [\Sigma(x_2) \cup A_2^c]) \cup \{\epsilon\} \\
= \Sigma_e((x_1, x_2))
\end{align*}$$

Equality (b) holds since

$$[A - A_i] \cup A_i^c = [A - A_i] \cup [\Sigma - A] = \Sigma - A_i = A_i^c.$$ 

It follows that the set of enabled events at $(x_1, x_2)$ in $G_1^{M_1} \parallel_{A_2} G_2$ is the same as the set of enabled events at $(x_1, x_2)$ in $G_1^{Aug_i} \parallel_{A} G_2^{Aug_2}$. Since $G_i$ is a subautomaton of $G_i^{Aug_i}$, it follows that each transition of the left NSM is also a transition of the right NSM.

Next we prove the converse that each transition of the right NSM is also a transition of the left NSM. Since we do not augment an event outside $A_i$, it holds that a transition of the right NSM on an event outside $A$ is also a transition of the left NSM. Consider next a transition on an event $\sigma \in A$. Then without loss of generality, by Assumption 1, we can assume $\sigma \in A_2 \cap \Sigma_{o2}$. Then it suffices to prove that a transition on an event $\sigma \in A_2 \cap \Sigma_{o2}$ of the right NSM is also a transition of the left NSM.

By Definition 2, for a transition on $\sigma$ to occur in the composed system, $\sigma$ must be enabled by both systems, i.e., $\sigma \in \Sigma_{Aug,e}((x_1, x_2))$. Then

$$\begin{align*}
\sigma \in [A_2 \cap \Sigma_{o2}] \cap \Sigma_{Aug,e}((x_1, x_2)) \\
\Rightarrow \sigma \in [A_2 \cap \Sigma_{o2}] \cap ([\Sigma(x_1) \cup A_i^c] \cap [\Sigma(x_2) \cup A_2^c]) \cup \{\epsilon\} \\
\Rightarrow \sigma \in [\Sigma(x_1) \cup A_i^c] \cap [\Sigma(x_2) \cup A_2 \cap \Sigma_{o2}].
\end{align*}$$

Since $\sigma \in \Sigma(x_2)$, exists a transition $(x_2, x', \sigma)$ in $G_2$. Also since

$$\begin{align*}
\sigma & \in \Sigma(x_1) \cup A_i^c \\
& = [\Sigma(x_1) \cap \Sigma_{o1}] \cup [\Sigma(x_1) - \Sigma_{o1}] \cup A_i^c \\
& = [\Sigma(x_1) \cap \Sigma_{o1}] \cup ([\Sigma(x_1) - \Sigma_{o1}] \cup A_i^c) \cup M_1(\epsilon) \cup [((\Sigma(x_1) - \Sigma_{o1}) \cup A_i^c) - M_1(\epsilon)],
\end{align*}$$

the following cases exist.

1. $\sigma \in \Sigma(x_1) \cap \Sigma_{o1}$.

In this case, since $\sigma$ is observable and already defined at $x_1$, there is no augmentation on $\sigma$ at $x_1$ in $G_1$. Since there is no augmentation on $\sigma$ in $G_2$ at any of its states, including $x_2$, there is no newly introduced transition on $\sigma$ in the right NSM.
2. \( \sigma \in ([\Sigma(x_1) - \Sigma_{o_1}] \cup A_1^c) - M_1(\epsilon) \).

(a) If \( \alpha_1(x_1, M_1(\sigma)) \neq \emptyset \), then by step 2 of Algorithm 1, for \( x_1' \in \alpha_1(x_1, \sigma_1) \) with \( \sigma_1 \in M(\sigma), \)

if transition \((x_1, \sigma_1, x_1')\) defined in \( G_1 \),
then transitions \((x_1, \sigma_1, x_1'), (x_1, \sigma, x_1')\) defined in \( G_1^{Aug1} \).

Then exists synchronous transition \(((x_1, x_2), \sigma, (x_1', x_2'))\) in \( G_1^{Aug1}_{A_1} \parallel G_2^{Aug2}_{A_2}. \) By clause 1 of Definition 2, it also holds that the transition \(((x_1, x_2), \sigma_1, (x_1', x_2'))\) is in \( G_1^{M_1}_{A_1} \parallel G_2^{M_2}_{A_2}. \)

(b) If \( \alpha_1(x_1, M_1(\sigma)) = \emptyset \), then by step 2 of Algorithm 1,

transition \((x_1, \sigma, x_1')\) is defined in \( G_1^{Aug1} \).

Then exists synchronous transition \(((x_1, x_2), \sigma, (x_1', x_2'))\) in \( G_1^{Aug1}_{A_1} \parallel G_2^{Aug2}_{A_2}. \) It also holds that this transition is in \( G_1^{M_1}_{A_1} \parallel G_2^{M_2}_{A_2} \) by clause 4 of Definition 2.

3. \( \sigma \in ([\Sigma(x_1) - \Sigma_{o_1}] \cup A_1^c) \cap M_1(\epsilon) \).

(a) If \( \sigma \in \Sigma(x_1) \), then \( \sigma \in \Sigma(x_1) \cap M_1(\epsilon) \). Further since \( \sigma \in A, \sigma \in \Sigma(x_1) \cap A \cap M_1(\epsilon), \) and so by step 1 of Algorithm 1, \( \epsilon \in Aug_1(x_1) \). By step 3 of Algorithm 1,

if transition \((x_1, \sigma, x_1')\) defined in \( G_1 \),
then transitions \((x_1, \sigma, x_1'), (x_1, \epsilon, x_1'), (x_1, \sigma, x_1)\) defined in \( G_1^{Aug1} \).

Then exist synchronous transitions \(((x_1, x_2), \sigma, (x_1, x_2'))\) and \(((x_1, x_2), \sigma, (x_1', x_2'))\), and asynchronous transitions \(((x_1, x_2'), \epsilon, (x_1', x_2'))\) and \(((x_1, x_2), \epsilon, (x_1', x_2))\) in \( G_1^{Aug1}_{A_1} \parallel G_2^{Aug2}_{A_2}. \) It also holds that by clause 4 of Definition 2, the transition \(((x_1, x_2), \sigma, (x_1', x_2'))\) is in \( G_1^{M_1}_{A_1} \parallel G_2^{M_2}_{A_2} \), whereas by clause 2 of Definition 2, the transition \(((x_1, x_2), \sigma, (x_1', x_2'))\) is in \( G_1^{M_1}_{A_1} \parallel G_2^{M_2}_{A_2} \), and finally, by clause 5 of Definition 2, transitions \(((x_1, x_2'), \epsilon, (x_1', x_2'))\) and \(((x_1, x_2), \epsilon, (x_1', x_2))\) are in \( G_1^{A_1}_{A_1} \parallel G_2^{A_2}_{A_2} \).

(b) If \( \sigma \notin \Sigma(x_1) \), then \( \sigma \in A_1^c \cap M_1(\epsilon) \subseteq Aug_1(x_1) \cap M_1(\epsilon), \) and so by step 3 of Algorithm 1,

\((x_1, \sigma, x_1)\) is defined in \( G_1^{Aug1} \).

Then exists synchronous transition \(((x_1, x_2), \sigma, (x_1, x_2'))\) in \( G_1^{Aug1}_{A_1} \parallel G_2^{Aug2}_{A_2}. \) It also holds that this transition is in \( G_1^{M_1}_{A_1} \parallel G_2^{M_2}_{A_2} \) by clause 4 of Definition 2.

This completes the proof.

In the special case, when the event priority sets exhaust the entire event set, i.e., when \( A = \Sigma, \) all \( M \)-synchronous executions can be simulated as ordinary synchronous executions. Then no \( M \)-synchronizations are needed and so all masks can be treated as identity mask. We state this formally below.
Assumption 2 For \( i \leq n \), consider NSM \( G_i \) possessing event priority set \( A_i \) such that \( A = \bigcup_i A_i = \Sigma \).

Theorem 2 Under the Assumptions 1 and 2,

\[
G_{A_1}^{M_1} \parallel A_2 \parallel G_2 = G_{A_1}^{Aug_1, M_1} \parallel \Sigma \parallel G_2^{Aug_2} = G_{A_1}^{Aug_1, Id} \parallel \Sigma \parallel G_2^{Aug_2}.
\]

Proof: By Theorem 1, we have \( G_{A_1}^{M_1} \parallel A_2 \parallel G_2 = G_{A_1}^{Aug_1, M_1} \parallel \Sigma \parallel G_2^{Aug_2} \). Thus, it suffices to prove that

\[
G_{A_1}^{Aug_1, M_1} \parallel \Sigma \parallel G_2^{Aug_2} = G_{A_1}^{Aug_1, Id} \parallel \Sigma \parallel G_2^{Aug_2}. \tag{5}
\]

Since the state sets, the event sets, and the initial states of the two NSM’s are all identical, we only need to show that they also have the identical set of transitions. Note that since the mask functions of the right NSM of (5) are identity masks, no \( M \)-synchronizations are allowed in the right NSM of (5). Consequently, the right NSM of (5) is a subautomaton of the left NSM of (5). Hence, each transition of the right NSM of (5) is also a transition of the left NSM of (5). It remains to prove the converse that each transition of the left NSM of (5) is also a transition of the right NSM of (5). The analysis carried out in proof of Theorem 1 is also valid here since Theorem 1 applies under one less assumption.

We consider transitions on an event \( \sigma \in A \) in left NSM. Then as in proof of Theorem 1, we can assume \( \sigma \in A_2 \cap \Sigma_2 \). Then as in the proof of Theorem 1, we again have the cases 1, 2(a), 2(b), 3(a), and 3(b). There is no new transition introduced in case 1. In case 2(a), each \( M \)-synchronization by a transition on \( \sigma_1 \) in \( G_{A_1}^{Aug_1, M_1} \parallel \Sigma \parallel G_2^{Aug_2} \) is duplicated by an ordinary synchronization on \( \sigma \) in \( G_{A_1}^{Aug_1, M_1} \parallel \Sigma \parallel G_2^{Aug_2} \). In cases 2(b), 3(a), and 3(b), each asynchronous execution on \( \sigma \) in \( G_{A_1}^{Aug_1, M_1} \parallel \Sigma \parallel G_2^{Aug_2} \) is duplicated by an ordinary synchronous execution on \( \sigma \) in \( G_{A_1}^{Aug_1, M_1} \parallel \Sigma \parallel G_2^{Aug_2} \), whereas each asynchronous execution on \( \epsilon \) in \( G_{A_1}^{Aug_1, M_1} \parallel \Sigma \parallel G_2^{Aug_2} \) is duplicated by an appropriate \( \epsilon \)-transition in \( G_{A_1}^{Aug_1, M_1} \parallel \Sigma \parallel G_2^{Aug_2} \). Thus, by replacing non-identity masks \( M_1 \) and \( M_2 \) by the identity mask, only certain duplicate transitions on \( \sigma \in A \) are avoided, but no \( \sigma \)-transition of the left NSM is removed in the right NSM.

Finally, due to Assumption 2, \( A = \Sigma \), and so the above statements are true of all transitions on all events \( \sigma \in \Sigma \). This completes the proof.

We showed that under Assumptions 1 and 2, PSCM of two systems having non-identity masks can be computed first by augmenting each of the systems, then computing PSCM of the augmented systems possessing identity masks. Further, by Remark 1, when two systems interact through identity masks, their PSCM reduces to simply their PSC. Also note that when the event priority set of each system is the entire event set \( \Sigma \), PSC reduces to SSC. This leads to the following corollary.

Corollary 1 Under the Assumptions 1 and 2,

\[
G_{A_1}^{M_1} \parallel A_2 \parallel G_2 = G_{A_1}^{Aug_1, M_1} \parallel \Sigma \parallel G_2^{Aug_2} = G_{A_1}^{Aug_1, Id} \parallel \Sigma \parallel G_2^{Aug_2} = G_{A_1}^{Aug_1} \parallel \Sigma \parallel G_2^{Aug_2} = G_{A_1}^{Aug_1} \parallel G_2^{Aug_2}.
\]

The result in the above corollary can be read as follows. Under Assumptions 1 and 2,

\[
PSCM((G_1, A_1, M_1), (G_2, A_2, M_2)) = PSC((G_1^{Aug_1}, A), (G_2^{Aug_2}, A)) = SSC(G_1^{Aug_1}, G_2^{Aug_2}).
\]
Note that Assumptions 1 and 2 automatically hold when one of systems can observe every event completely (has identity mask) and has priority over every event (priority set = \(\Sigma\)). This yields the following corollary, which is useful in supervisory control setting, as a plant can “observe” every event and has priority over every event.

**Corollary 2**

\[
G_{1\Sigma}^I \parallel_{A_2} M G_2 = G_{1\Sigma}^{Aug_1} \parallel \Sigma G_2^{Aug_2} = G_{1\Sigma} \parallel \Sigma G_2^{Aug_2} = G_1 \parallel G_2^{Aug_2}.
\]

**Proof:** Since priority set of \(G_1\) is \(\Sigma\), and its observation mask is \(Id\), it follows that \(A_1 = \Sigma\) and \(\Sigma_{o1} = \Sigma\). Thus \(A_1 \cap \Sigma_{o1} = \Sigma\), which implies that \(\sigma \in A_1 \cap \Sigma_{o1}\) for any \(\sigma \in \Sigma\). Thus, Assumption 1 holds. Also since \(A_1 \cup A_2 = \Sigma \cup A_2 = \Sigma\), Assumption 2 also holds. Then the first equality in assertion of the corollary follows from Corollary 1.

To show the second equality of the assertion, it suffices to show that \(G_{1}^{Aug_1} = G_1\). Since each event is observable in \(G_1\), \(M_1(\varepsilon) = \emptyset\). So for each state \(x_1\) in \(G_1\), \(\Sigma(x_1) \cap M_1(\varepsilon) = \emptyset\). Thus

\[
Aug_1(x_1) = [A - A_1] \cup [A_1 \cap \Sigma(x_1) - \Sigma_{o1}] \\
= [\Sigma - \Sigma] \cup [\Sigma \cap \Sigma(x_1) - \Sigma] \\
= \emptyset.
\]

Since \(Aug_1(x_1) = \emptyset\) for any state \(x_1\) in \(G_1\), no augmentation takes place in \(G_1\), i.e., \(G_{1}^{Aug_1} = G_1\). Finally the last equality in the assertion of the corollary follows from the equivalence of PSC and SSC when each priority set is the entire event set \(\Sigma\).

Note that no assumptions are needed in Corollary 2.

## 5 Control of Discrete Event Systems via PSCM

In this section, we extend the theory of supervisory control under partial observation to the present setting where control is exercised by means of interaction via PSCM (under the restriction that the event priority set of the supervisor is a subset of the event priority set of the plant, i.e., there are no “driven” events). The plant is modeled by a NSM \(G = (X, \Sigma, \alpha, X_0)\) having event priority set \(\Sigma\) and identity observation mask. The supervisor is modeled as another NSM \(S = (Y, \Sigma, \beta, Y_0)\) having event priority set \(\Sigma_c = \Sigma - \Sigma_u\), the set of controllable events, and an observation mask \(M\). We let \(\Sigma_o\) denote the set of completely observable events of a supervisor. The control specification is generated by a state machine \(R = (Q, \Sigma, \delta, Q_0)\). The control task is to design a supervisor \(S\) such that the behavior of the controlled plant equals the specification language, i.e., \(L(G_{\Sigma}^{I} \parallel_{\Sigma_c} M S) = L(R)\). We show that both the existence and synthesis of a supervisor can be determined polynomially.

In [15], SSC based control under partial observation was studied, allowing the supervisor to be nondeterministic. Due to the use of SSC, supervisor was required to be \((\Sigma_u, M)\)-compatible. It was shown that a necessary and sufficient condition for the existence of a \((\Sigma_u, M)\)-compatible supervisor such that the behavior of the controlled system equals the specification language is *achievability.*
Definition 3 [15]

1. $K \subseteq \Sigma^*$ is said to be $\Sigma_u$-controllable with respect to $\Sigma^*$ if $\forall s \in \text{pr}(K)$ and $\forall a \in \Sigma_u$, $sa \in \text{pr}(K)$.

2. $K \subseteq \Sigma^*$ is said to be $M$-recognizable with respect to $\Sigma^*$ if $\forall s, t \in \Sigma^*$ and $\forall a \in \Sigma$ with $M(a) = M(e)$, $sat \in \text{pr}(K) \Rightarrow sa^*t \subseteq \text{pr}(K)$.

3. $K \subseteq \Sigma^*$ is said to be $(\Sigma_u, M)$-achievable with respect to $\Sigma^*$ ($(\Sigma^*, \Sigma_u, M)$-achievable for short) if $K$ is $\Sigma_u$-controllable and $M$-recognizable with respect to $\Sigma^*$, and $\forall a \in \Sigma$, $b \in \Sigma_u$ with $M(a) = M(b)$, $sat \in \text{pr}(K) \Rightarrow \{sbt\} \subseteq \text{pr}(K)$.

4. $K \subseteq L(G)$ is said to be $(L(G), \Sigma_u, M)$-achievable if exists $(\Sigma^*, \Sigma_u, M)$-achievable $K'$ such that $\text{pr}(K) = \text{pr}(K') \cap L(G)$.

It was shown in [15] that $(L(G), \Sigma_u, M)$-achievability of prefix-closed language is preserved under intersection and so the infimal $(L(G), \Sigma_u, M)$-achievable superlanguage of a language $K \subseteq L(G)$, denoted $\inf \overline{PA}(L(G))(K)$, exists. Further it holds that,

$$
\inf \overline{PA}(L(G))(K) = \inf \overline{PA}_{\Sigma^*}(K) \cap L(G),
$$

and so $\inf \overline{PA}(L(G))(K)$ can be computed by first computing $\inf \overline{PA}_{\Sigma^*}(K)$. The following algorithm was given in [15, Algorithm 1] to compute a generator $\hat{R}$ of the language $\inf \overline{PA}_{\Sigma^*}(K)$.

Algorithm 2 [15] Let $R = (Q, \Sigma, \delta, Q_0)$ be a state machine with $L(R) = \text{pr}(K)$.

A1 For each transition $(q, b, q_1)$ with either $b \in M(\epsilon)$ or $\exists (q, b', q_2)$ such that $M(b) = M(b') \neq M(\epsilon)$, replace $(q, b, q_1)$ by a pair of transitions $(q, \epsilon, q_1')$ and $(q_1', b, q_1)$, where $q_1'$ is a newly added state. We denote the resulting state machine by $R' = (Q', \Sigma, \delta', Q_0)$.

A2 For every transition $(q, b, q')$ of $R'$ with $b \in M(\epsilon)$, add transitions $(q, b, q)$ and $(q, \epsilon, q')$.

A3 For every state $q \in Q'$ and every event $b \in \Sigma_u \cap M(\epsilon)$, if $b$ is not defined at $q$ then let $\delta(q, b) = \delta(q, \epsilon)$.

A4 For every state $q \in Q'$, every event $b \in \Sigma_u - M(\epsilon)$, and every transition $(q, a, q')$ in $R'$ with $M(a) = M(b)$, add a transition $(q, b, q')$.

A5 (i) For every state $q \in Q'$ and every event $b \in \Sigma_u - M(\epsilon)$, if no such an event $a$ with $M(a) = M(b)$ is defined at $q$ then add a transition $(q, b, \text{dump})$, where $\text{dump}$ is an added state, (ii) Add self-loops at the dump state on all uncontrollable events.

Let the state machine constructed by Algorithm 2 be denoted $\hat{R} = (Q' \cup \{\text{dump}\}, \Sigma, \hat{\delta}, Q_0)$. Then from [15, Theorem 6], $L(\hat{R}) = \inf \overline{PA}_{\Sigma^*}(L(R))$. 

18
Next we aim to derive a necessary condition for the existence of a supervisor such that $L(G^d_{\Sigma} \| M_{\Sigma} S) = L(R)$. Since $G$ possesses priority set $\Sigma$ and identity mask, from Corollary 2, $G^d_{\Sigma} \| M_{\Sigma} S = G_{\Sigma} \| M_{\Sigma} S^A$. Let the augmented supervisor be given by, $S^A = (Y, \Sigma, \beta^A, Y_0)$. We first observe that $L(S^A)$ is $(\Sigma^*, \Sigma_u, M)$-achievable and the PSCM-based controlled behavior $L(G^d_{\Sigma} \| M_{\Sigma} S)$ is $(L(G), \Sigma_u, M)$-achievable.

Lemma 1 Consider plant $G$ with priority set $\Sigma$ and identity observation mask, a supervisor $S$ with priority set $\Sigma_c$ and observation mask $M$. Then

1. $L(S^A)$ is $(\Sigma^*, \Sigma_u, M)$-achievable.
2. $L(G^d_{\Sigma} \| M_{\Sigma} S)$ is $(L(G), \Sigma_u, M)$-achievable.

Proof:

1. For $y \in Y$, let $Aug(y)$ be the set of labels used for augmenting transitions in $S$ to obtain $S^A$. Then from Algorithm 1,

   \[
   \forall y \in Y, \quad Aug(y) := \begin{cases} 
   [\Sigma_c \cap \Sigma(y) - \Sigma_o] \cup \Sigma_u & \text{if } \Sigma(y) \cap M(\epsilon) = \emptyset \\
   [\Sigma_c \cap \Sigma(y) - \Sigma_o] \cup \Sigma_u \cup \{\epsilon\} & \text{otherwise}
   \end{cases}
   \]

   Note that the set of events defined at state $y$ of $S^A$ is given by, $\Sigma(y) \cup Aug(y)$.

   We first show that $L(S^A)$ is $\Sigma_u$-controllable with respect to $\Sigma^*$. Pick $s \in L(S^A)$, $a \in \Sigma_u$. Let $y \in Y$ be a state reached by $s$ in $S^A$. Then since $\Sigma_u \subseteq Aug(y)$, it follows that $sa \in L(S^A)$.

   Next we show that $L(S^A)$ is $M$-recognizable with respect to $\Sigma^*$. Pick $s, t \in \Sigma^*$, $a \in M(\epsilon)$ such that $sat \in L(S^A)$. Let $y_s, y_{sa}$ be states such that $y_s$ is reached by executing $s$ at $Y_0$, $y_{sa}$ is reached by executing $a$ at $y_s$ in $S^A$, and $t$ is defined at $y_{sa}$. Then $a \in \Sigma(y_s) \cup Aug(y_s)$. We claim that $a \in Aug(y_s)$ even when $a \in \Sigma(y_s)$. Clearly this is the case when $a \in \Sigma_u$ since $\Sigma_u \subseteq Aug(y_s)$. On the other hand, if $a \in \Sigma_c$, then it also holds that $a \in \Sigma_c - \Sigma_o$ ($a \in M(\epsilon)$ implies $a \notin \Sigma_o$). This together with $a \in \Sigma(y_s)$ implies that $a \in \Sigma(y_s) \cap \Sigma_c - \Sigma_o \subseteq Aug(y_s)$. Since $a \in Aug(y_s) \cap M(\epsilon)$, from step 3 of Algorithm 1, transitions $(y_s, a, y_s)$ and $(y_s, \epsilon, y_{sa})$ are added in $S$ for augmentation. It follows that $sa^t \subseteq L(S^A)$.

   Finally, we show that $L(S^A)$ is $(\Sigma_u, M)$-achievable with respect to $\Sigma^*$. Pick $s, t \in \Sigma^*$, $a \in \Sigma$, $b \in \Sigma_u$ with $M(a) = M(b)$ such that $sat \in L(S^A)$. Then as before, $a \in Aug(y_s)$. Since $b \in \Sigma_u$ and $\Sigma_u \subseteq Aug(y_s)$, $b \in Aug(y_s)$. If $b \in M(a) \neq M(\epsilon)$, then from step 2 of Algorithm 1 a $b$-transition is added along side each $a$-transition in $S^A$. On the other hand if $b \in M(a) = M(\epsilon)$, then from step 3 of Algorithm 1, a self-loop transition $(y_s, b, y_s)$ is added in $S$ and also an $\epsilon$-transition is added along side each $a$-transition. In either case, it follows that that $sbt \in L(S^A)$. Thus, $L(S^A)$ is $(\Sigma^*, \Sigma_u, M)$-achievable, as desired. This completes the proof of the first part of Lemma 1.
2. Now we prove the second part of Lemma 1. By Corollary 2,

\[ L(G^{Id}_{\Sigma_c} M, S) = L(G^{S_{Aug}}) = L(G) \cap L(S_{Aug}). \]

By Lemma 1, \( L(S_{Aug}) \) is \((L(G), \Sigma_u, M)\)-achievable. So from last part of Definition 3, \( L(G) \cap L(S_{Aug}) = L(G^{Id}_{\Sigma_c} M, S) \) is \((L(G), \Sigma_u, M)\)-achievable.

From the second part of Lemma 1, if a specification language equals the language of a PSCM controlled plant, then it must be \((L(G), \Sigma_u, M)\)-achievable. Thus, \((L(G), \Sigma_u, M)\)-achievability serves as a necessary condition for the existence of a supervisor. Next we aim to show the converse that if a specification language is \((L(G), \Sigma_u, M)\)-achievable, then it is enforceable by a PSCM-based control.

We propose the following algorithm for constructing a supervisor starting from a specification state machine.

**Algorithm 3** Consider a specification state machine \( R = (Q, \Sigma, \delta, Q_0) \).

1. For every transition \((q, b, q_1)\) with either \( M(b) = M(\epsilon) \) or \( \exists (q, b', q_2) \) such that \( M(b) = M(b') \neq M(\epsilon) \), replace \((q, b, q_1)\) by a pair of transitions \((q, \epsilon, q_{1}')\) and \((q_{1}', b, q_1)\), where \( q_{1}' \) is a newly added state. Note that this step is same as step A1 of Algorithm 2, and so we denote the resulting state machine as \( R' = (Q', \Sigma, \delta', Q_0) \).

2. For every state \( q \in Q' \), every event \( b \in \Sigma_u - M(\Sigma(q)) - M(\epsilon) \), add a transition \((q, b, dump)\), where \( dump \) is an added state. Denote the resulting state machine as \( \bar{R} := (Q' \cup \{dump\}, \Sigma, \bar{\delta}, Q_0) \).

The following example illustrates Algorithm 3.

**Example 3** Consider \( R \) shown in Figure 4, with

\[ \Sigma = \{a, b, c, d\}, \Sigma_u = \{b, c, d\}, M(a) = M(b) = \{a, b\} \neq M(\epsilon), M(c) = \{c, \epsilon\}, M(d) = \{d\}. \]

![Figure 4: R (left), R' (middle), and \( \bar{R} \) (right)](image-url)
Since at state 1, \( M(a) = M(b) \), by step 1 of Algorithm 3, replace transition \((1, a, 2)\) (resp. \((1, b, 3)\)) by transitions \((1, \epsilon, 2')\) and \((2', a, 2)\) (resp. \((1, \epsilon, 3')\) and \((3', b, 3)\)). Next since \( M(c) = M(\epsilon) \), by step 1 of Algorithm 3, replace \((1, c, 4)\) by \((1, \epsilon, 4')\) and \((4', c, 4)\). The resulting state machine \( R' \) is shown in Figure 4.

At state 1, since \( \Sigma_u - M(\Sigma(1)) - M(\epsilon) = \{b\} \), by step 2 of Algorithm 3, transition \((1, b, dump)\) is added in \( R' \), where \( dump \) is an newly added state. At state 2', \( \Sigma_u - M(\Sigma(2')) - M(\epsilon) = \{d\} \), so transition \((2', d, dump)\) is added in \( R' \) to obtain \( \bar{R} \). Similarly, one can compute the set of transitions on \( \Sigma_u - M(\Sigma(\cdot)) - M(\epsilon) \) for other states. The details are omitted here. The resulting state machine \( \bar{R} \) is shown in Figure 4.

We claim that when \( L(R) \) is \( (L(G), \Sigma_u, M) \)-achievable, \( \bar{R} \) constructed in Algorithm 3 can be used a supervisor, i.e., \( L(G^{1\bar{d}}||_{\Sigma_u} \bar{R}) = L(R) \). Note that \( L(G^{1\bar{d}}||_{\Sigma_u} \bar{R}) = L(G) \cap L(\bar{R}^{Aug}) \).

In the following we first prove that \( L(\bar{R}^{Aug}) \) equals \( \inf \bar{F}_{\bar{A}}(L(R)) \). The proof is based on showing the equivalence of the languages \( L(\bar{R}^{Aug}) \) and \( L(\bar{R}) \), where the construction of \( \bar{R} \) is stated above in Algorithm 2.

**Lemma 2** Given \( R \), the set of controllable events \( \Sigma_c \) and a mask \( M \), \( L(\bar{R}^{Aug}) = L(\bar{R}) \), where \( \bar{R} \) is computed by Algorithm 3, \( \bar{R}^{Aug} \) is computed by Algorithm 1, and \( \bar{R} \) is computed by Algorithm 2.

**Proof:** In order to facilitate the proof, the computation of \( \bar{R}^{Aug} \) is shown below by combining Algorithms 1 and 3.

1. **B1** For every transition \((q, b, q_1)\) with either \( M(b) = M(\epsilon) \) or \( \exists (q, b', q_2) \) such that \( M(b) = M(b') \neq M(\epsilon) \), replace \((q, b, q_1)\) by a pair of transitions \((q, \epsilon, q'_1)\) and \((q'_1, b, q_1)\), where \( q'_1 \) is a newly added state. The result of this step is the state machine, \( R' = (Q', \Sigma, \delta', Q_0) \).

2. **B2** For every state \( q \), every event \( b \in \Sigma_u - M(\Sigma(q)) - M(\epsilon) \), add a transition \((q, b, dump)\), where \( dump \) is an added state. The result of this step is the state machine, \( \bar{R} := (Q' \cup \{dump\}, \Sigma, \bar{\delta}, Q_0) \).

3. **B3** For \( q \in Q' \cup \{dump\} \) and \( \sigma \in Aug(q) - M(\epsilon) \), (i) if \( \bar{\delta}(q, M(\sigma)) \neq \emptyset \), add transitions on \( \sigma \) from \( q \) to all states in the set \( \bar{\delta}(q, M(\sigma)) \), (ii) otherwise add a self-loop on \( \sigma \) at \( q \).

4. **B4** For \( q \in Q' \cup \{dump\} \) and \( \sigma \in Aug(q) \cap M(\epsilon) \), (i) add self-loop on \( \sigma \) at \( q \). (ii) Further if \( \bar{\delta}(q, \sigma) \neq \emptyset \), add transitions on \( \epsilon \) from \( q \) to all states in the set \( \bar{\delta}(q, \sigma) \). The resulting state machines is \( \bar{R}^{Aug} \).

Next we compare the construction of \( \bar{R}^{Aug} \) and that of \( R \). Note that the state set of both \( \bar{R}^{Aug} \) and \( R \) are the same, namely, \( Q' \cup \{dump\} \). So we next compare the transitions in the two state machines.

- Steps A1 and B1 introduce the transitions in the same way.
• Step A2 introduces self-loop transition on an unobservable event $\sigma$ at a state $q \in Q'$ where $\sigma$ is defined, and also $\epsilon$-transitions along side all $\sigma$-transitions at $q$. Since $\sigma \in \Sigma(q) - \Sigma_o$, and

$$Aug(q) = [\Sigma(q) \cap \Sigma_c - \Sigma_o] \cup \Sigma_u \cup \{\epsilon\} \supseteq [\Sigma(q) \cap \Sigma_c - \Sigma_o] \cup [\Sigma(q) \cap \Sigma_u - \Sigma_o] = [\Sigma(q) - \Sigma_o],$$

it follows that $\sigma \in Aug(q)$. Also, $\sigma \in M(\epsilon)$ and so the same set of transitions as introduced by step A2 are introduced by steps B4(i) and B4(ii).

• At a state $q \in Q'$, step A3 introduces transitions on an uncontrollable and unobservable event along side each $\epsilon$-transition at $q$ (including a self-loop at $q$). Only a self-loop transition on such an event is added at $q$ by step B4(i). (Recall that $\Sigma_u \subseteq Aug(\cdot)$.) As we will see this is the only difference between $\hat{R}$ and $\hat{R}^{Aug}$. For $b \in \Sigma_u \cap M(\epsilon)$ if $(q, \epsilon, q')$ is a transition in $\hat{R}'$, then step A3 adds transitions $(q, b, q)$ and $(q, b, q')$, whereas only the transition $(q, b, q)$ is added by step B4(i). This however keeps the languages $L(\hat{R})$ and $L(\hat{R}^{Aug})$ to be the same since the extra transition $(q, b, q')$ of $\hat{R}^{Aug}$ can be simulated as the self-loop transition $(q, b, q)$ followed by the $\epsilon$-transition $(q, \epsilon, q')$ of $\hat{R}$ without affecting the event-trace generated (trace $b$ equals trace $bc$).

• At a state $q \in Q'$, step A4 introduces transitions on an event $\sigma \in \Sigma_u - M(\epsilon) \cap M(\Sigma(q))$ along side all transitions in $M(\sigma) \cap \Sigma(q)$. The same set of transitions are introduced on the event $\sigma \in \Sigma_u - M(\epsilon) \cap M(\Sigma(q)) \subseteq Aug(q) - M(\epsilon)$ by step B3(i).

• For $q \in Q'$, steps A5(i) and B2 introduce, in the same way, uncontrollable transitions that are not unobservable and for which no indistinguishable events are defined at $q$.

• Step A5(ii) introduces self-loops on uncontrollable events at the dump state. Since in $\hat{R}$, no transitions are defined at the dump state, $Aug(dump) = \Sigma_u$. At the dump state, self-loops on events in $Aug(dump) - M(\epsilon) = \Sigma_u - M(\epsilon)$ are introduced by step B3(ii), whereas self-loops on events in $Aug(dump) \cap M(\epsilon) = \Sigma_u \cap M(\epsilon)$ are introduced by step B4(i).

It can be seen that except for one case, the transitions in $\hat{R}$ are the same as those in $\hat{R}^{Aug}$, and even in this exceptional case, the difference is such that the language is preserved, proving the assertion that $L(\hat{R}) = L(\hat{R}^{Aug})$.

The following corollary immediately follows.

**Corollary 3** Given $R$, the set of controllable events $\Sigma_c$ and a mask $M$, if $L(R) \subseteq L(G)$ is $(L(G), \Sigma_u, M)$-achievable, then $L(G^{Id}_{\Sigma_c} M^{\epsilon}_\Sigma \hat{R}) = L(R)$, where $\hat{R}$ is computed by Algorithm 3.

**Proof:** From Corollary 2, $L(G^{Id}_{\Sigma_c} M^{\epsilon}_\Sigma \hat{R}) = L(G) \cap L(\hat{R}^{Aug})$. By Lemma 2, $L(\hat{R}^{Aug}) = L(\hat{R})$, where $L(\hat{R}) = \inf \overline{PA}_{\Sigma_c}(L(R))$ [15, Theorem 6]. It follows that

$$\inf \overline{PA}_{L(G)}(L(R)) = L(G) \cap \inf \overline{PA}_{\Sigma_c}(L(R))$$
and so the result follows.

Since \( L(R) \) is given to be \((L(G), \Sigma, M)\)-achievable, it holds that \( L(R) = \inf FALG(L(R)) \), and so the result follows.

By Lemma 1 and Corollary 3, we obtain the following necessary and sufficient condition for the existence of a PSCM-based supervisor enforcing a given specification.

**Theorem 3** Given \( G \), \( R \), the set of controllable events \( \Sigma \), and a mask \( M \), there exists a supervisor \( S \) such that \( L(G^d / M \Sigma S) = L(R) \) if and only if \( L(R) \subseteq L(G) \) and \( L(R) \) is \((L(G), \Sigma, M)\)-achievable.

**Proof:** \((\Rightarrow)\) From Lemma 1, \( L(G^d / M \Sigma S) = (L(G), \Sigma, M)\)-achievable, and since \( L(R) = L(G^d / M \Sigma S) \), \( L(R) \) is also \((L(G), \Sigma, M)\)-achievable. Further from Corollary 2, \( L(G^d / M \Sigma S) = L(G^{\Sigma} S^d) = L(G) \cap L(S^d) \), it follows that \( L(R) = L(G^d / M \Sigma S) \subseteq L(G) \).

\((\Leftarrow)\) Since \( L(R) \subseteq L(G) \) is \((L(G), \Sigma, M)\)-achievable, using Algorithm 3 we construct NSM \( \tilde{R} \). Then from Corollary 3, \( L(G^d / M \Sigma \tilde{R}) = L(R) \), i.e., \( \tilde{R} \) is the desired supervisor.

**Remark 3** It follows from Theorem 3 that the existing tests for achievability, which is of complexity \( O(|G| |R|^2) \) [15, Remark 5], can be applied to verify the existence of a supervisor. Further if the existence condition is satisfied, a supervisor can be obtained by applying Algorithm 3 to a generator \( R \) of the specification language, implying the synthesis of a supervisor is of complexity \( O(|G|) \). Note that in general the state machine obtained by Algorithm 3 is not \((\Sigma, M)\)-compatible. Thus, by using the control mechanism of PSCM, the requirement of \((\Sigma, M)\)-compatibility of a supervisor has been removed, as desired.

6 An Illustrative Example

We illustrate Theorem 3 through a manufacturing example, taken from [15]. The manufacturing system consists of one robot, two workstations and two storage-stations. The robot moves among the workstations and storage-stations on a guide rail. Initially, the robot departs from workstation 1 (event \( a \)). Then it picks up a part either from storage-station 1 (event \( b_1 \)) or storage-station 2 (event \( b_2 \)), and delivers the part to workstation 2 for processing (event \( c \)). After the processing, robot returns the part to a storage-station. After returning parts, robot goes back to workstation 1 and can repeat the whole process. A state machine model \( G \) of the system is drawn in Figure 5.

Not returning the part to its original storage-station is not desirable. The specification \( R \), also shown in Figure 5, gives the desired behavior. According to the specification, after processing, the robot returns the part to its original storage-station. The rest of the behavior is the same as the one feasible in the system.

We require that the part must be delivered to workstation 2 for processing, i.e., the event \( c \) is uncontrollable. Also, only events \( a \) and \( c \) are completely observable. Events \( b_1 \) and \( b_2 \) are observationally indistinguishable. Thus, we have \( \Sigma = \{a, b_1, b_2, c\} \), \( \Sigma_c = \{a, b_1, b_2\} \),
and the observation mask $M$ is given by, $M(a) = \{a\}$, $M(b_1) = M(b_2) = \{b_1, b_2\}$ and $M(c) = \{c\}$. It can be verified that $L(R)$ is not observable, i.e., a deterministic supervisor does not exist. However, $L(R)$ is $(L(G), \Sigma_u, M)$-achievable and so from Theorem 3, a PSCM-based supervisor does exist. (Thanks to PSCM-based control formalism developed here that allows supervisor to be nondeterministic.)

We use Algorithm 3 to construct $\tilde{R}$ that acts as a supervisor.

At state $q_1$ of $R$, $M(b_1) = M(b_2)$, by step 1 of Algorithm 3, we replace transition $(q_1, b_1, q_2)$ (resp., $(q_1, b_2, q_3)$) by a pair of transitions $(q_1, \epsilon, q_1')$ and $(q_1', b_1, q_2)$ (resp., $(q_1, \epsilon, q_2')$ and $(q_2', b_2, q_3)$). Next by step 2 of Algorithm 3, since $\Sigma_u - M(\Sigma(q)) - M(\epsilon) = \{c\}$ for state $q \in \{q_0, q_1, q_1', q_4, q_5, q_6\}$, we add transitions on $c$ from states $q_0, q_1, q_1', q_4, q_5, q_6$ and $q_0$ to the newly added “dump” state.

We obtain the state machine $\tilde{R}$ drawn in Figure 5. One can verify that $L(G_{\Sigma_u}^{Id}|_{\Sigma_{\tilde{R}}} M \tilde{R}) = L(R)$, i.e., $\tilde{R}$ serves as a desired supervisor. It should be noted that $\tilde{R}$ is not $(\Sigma_u, M)$-compatible. As an example, the uncontrollable event $c$ is undefined at the dump state.

7 Conclusion

In this paper we introduced the notion of prioritized synchronous composition under mask (PSCM) to model interaction/control of discrete event systems under partial observation. This extends the formalism of prioritized synchronous composition (PSC) which assumes identity observation masks. We established a link between PSCM and PSC (and thereby SSC) under certain constraints on the priority sets and the mask functions: PSCM of two systems can be alternatively obtained by first augmenting individual systems, and next computing the PSC of the augmented systems. For this to work, the priority sets of the two
systems must exhaust the entire event set, and each event must be observable to a system having priority over the event.

We showed that when PSCM is adopted as a mechanism of control, not only the control & observation-compatibility requirements are removed of a supervisor, the existence condition is given by achievability that is weaker than controllability and observability combined. (The weaker condition is required since the supervisor is allowed to be nondeterministic.) This suggests that the notion of PSCM, introduced in the paper is an appropriate generalization of PSC to account for partial observation. We also showed that both the existence verification and synthesis of a PSCM-based supervisor is polynomially solvable. The existence is linear in the size of the plant and quadratic in the size of the specification, whereas the synthesis is linear in the size of the specification.

References


