Process Objects/Masked Composition: An Object Oriented Approach for Modeling and Control of Discrete Event Systems *

Mark A. Shayman
Electrical Engineering Department and
Institute for Systems Research
University of Maryland
College Park, MD 20742
Email: shayman@eng.umd.edu

Ratnesh Kumar
Department of Electrical Engineering
University of Kentucky
Lexington, KY 40506
Email: kumar@engr.uky.edu

Abstract

An object-oriented framework for modeling and supervisory control of discrete event systems is described. Control and observation masks are encapsulated with process logic to form process objects, and a single type of interconnection operator called masked composition is used to build complex process objects out of simpler component process objects. The approach applies to both deterministic and nondeterministic plant models and supervisory design. In addition to the usual benefits of object-oriented design, such as software reusability, it yields conditions under which the existence of a nondeterministic supervisor implies existence of a deterministic supervisor.

1 Introduction

In recent years, there has been growing interest in object-oriented methods for the modeling and design of engineering systems. Among other advantages, such an approach offers the promise of software reusability [2]. An approach to object-oriented design of discrete event systems has been proposed by Fabian and Lemarthon [5, 4, 3]. A basic goal is to develop the capacity to build modules (objects) in which the vast majority of

*This research was supported in part by the National Science Foundation under grants EEC-9402384, ECS-9312587, ECS-9409712, ECS-9626399, and ECS-9709712.
the logic is standardized (inherited from its class), and which can be used off-the-shelf and interconnected in different ways to model and control many different plants to meet different possible specifications. For a particular application, a portion of the logic would be configurable when defining an object as an instance of its class.

Our approach is based on two concepts defined in this paper: process objects and masked composition (PO/MC). Each process object consists of a logic submodule encapsulated with an input interface (observation mask) and output interface (control mask). The interfaces may be modified to reflect new sensor or actuator capabilities without requiring any modification in the logic submodule. When objects are interconnected by masked composition to build complex systems or to impose control, there are no compatibility conditions (such as supervisor completeness) that require off-the-shelf objects to be modified. Our framework easily accommodates specifications that limit concurrency, and the associativity of masked composition makes the approach suitable for modular control.

In addition to providing a basis for object-oriented modeling and control design, our approach leads to some new results on the control of nondeterministic discrete event systems. A greatly expanded version of this paper, which includes all omitted proofs, as well as examples and applications, is available on the World Wide Web [15]. Most of the results in this paper first appeared in abridged form in [14, 12].

2 Process Object Representation

A nondeterministic state machine (with \(\epsilon\)-moves) is represented by a four-tuple \(P := (X_P, \Sigma, \delta_P, X^0_P)\) where \(X_P\) denotes the state space of \(P\), \(\Sigma\) denotes the event set of \(P\), \(\delta_P : X_P \times \Sigma \to 2^{X_P}\), where \(\Sigma := \Sigma \cup \{\epsilon\}\), denotes the nondeterministic transition function of \(P\), and \(X^0_P \subset X_P\) denotes the nonempty set of initial states of \(P\). We use \(\delta^*_P\) to denote the extension of \(\delta\) from the set of events \(\Sigma\) to the set of strings \(\Sigma^*\). Thus \(\delta^*_P(x, s)\) denotes the set of states reachable from state \(x\) executing the sequence of events in the string \(s\) possibly interleaved with \(\epsilon\)-events. Define \(\Sigma_P(x) := \{\sigma \in \Sigma | \delta_P(x, \sigma) \neq \emptyset\}\)—i.e., the set of events (possibly including \(\epsilon\)) defined in state \(x\); and \(\Sigma_P(x) := \Sigma_P(x) \cup \{\epsilon\}\). \(L(P)\) denotes the generated language of \(P\)—i.e., \(L(P) = \{s \in \Sigma^* | \exists x_0 \in X^0_P \text{ s.t. } \delta^*_P(x_0, s) \neq \emptyset\}\). Given a language \(K \subseteq \Sigma^*\), \(\overline{K}\) is used to denote its prefix-closure.

In the standard supervisory control framework [11], the plant \(P\) generates all events. Those events that are observable are received by the supervisor \(S\). Based on the current string of observable events, the supervisor may disable any of the controllable events that the plant could generate in its current state. Specification of the uncontrollable events describes the limitations on the ability of the supervisor to control the plant, while specification of the unobservable events describes the limitations on the ability of the supervisor to observe the plant.

A simple method of implementing supervisory control is to interconnect the plant and supervisor by strict synchronous composition (SSC) [8]. For future reference, it is useful to view the interaction of the plant and supervisor via SSC as consisting of three steps: (1) The events enabled by the supervisor are those that it can execute in its current state. (2) If the event \(\sigma\) is possible in the current state of the plant and is enabled by the supervisor, then the plant can generate this event. (3) When the plant generates an event \(\sigma\), the supervisor responds by synchronously executing it.
In the implementation of supervisory control by SSC, the control and observation limitations on the
supervisor associated with the specification of the sets of uncontrollable and unobservable events are not explicitly
modeled. Instead, they are reflected in constraints on the logic (state machine structure) of the supervisor.
The requirement that no uncontrollable events be disabled (so-called \textit{supervisor completeness}) means that if
execution of a string $s$ results in states $x$ and $y$ for $P$ and $S$ respectively, and if the uncontrollable event $\sigma$ is
possible in $x$ for $P$, then it must be possible in $y$ for $S$. The requirement that the supervisor base its control
action only on those events that are actually observed (so-called \textit{observation compatibility}) means that if the
strings $s_1, s_2$ contain the same sequence of observable events and result in states $x_1, x_2$ for $P$ and $y_1, y_2$ for $S$,
and if $\sigma$ is an event possible in $x_1, x_2, y_1$, then $\sigma$ must also be possible in $y_2$.

One of our goals is to provide a framework for object-oriented supervisory control design. The constraints
on supervisor logic associated with supervisor completeness and observation compatibility make the SSC imple-
mation ill-suited for this purpose. Suppose that we wish to define an \textit{object class} called “one-step supervisor.”
It would have the object class “supervisor” as superior class. A one-step supervisor would be a supervisor with
two states $y_0, y_1$ and transitions $\{(y_0, \sigma, y_1)|\sigma \in A\}$, where $A \subseteq \Sigma$ is a configurable parameter. The creation
of an \textit{instance} of this object would require the specification of the parameter $A$. This approach is not viable
because the completeness and observation compatibility requirements make the admissible set for the config-
urable parameter $A$ dependent on the specific structure of the plant object to which the supervisor will be
connected.

The basic problem with the SSC approach is that the supervisor completeness and observation compatibility
conditions represent control and observation limitations \textit{implicitly} as restrictions on the supervisor logic. Our
solution is to separate the control and observation limitations from the logic and \textit{encapsulate} them together
with the logic to form \textit{process objects} that can be interconnected without compatibility constraints.

When SSC is viewed as a three-step procedure, the presence of a transition labeled by $\sigma$ in the current
state of a process serves three functions: (1) it \textit{enables} the other process to generate $\sigma$ if it can do so; (2) it
can \textit{generate} $\sigma$ if the other process enables it; (3) it can \textit{respond} to the event $\sigma$ if it is generated by the other
process. Masked composition (MC) is a generalization of SSC in which the first function is filtered by the
control mask and the third function is filtered by the observation mask.

To be able to define masked composition in such a way as to be associative, it is necessary to distinguish
between two types of transitions in an NSM $P$. The \textit{real} transitions are transitions of the usual type; they
can generate events, enable another process to generate events, and respond by synchronously executing events
generated by another process. In contrast, \textit{virtual} transitions can only enable and respond to events generated
by another process. They cannot generate (initiate) events on their own. (See Example 1.)

\textbf{Definition 1} A \textit{process object} consists of three components $((P, \hat{P}), C, M)$, where

1. $(P, \hat{P})$, the logic component, consists of an NSM $P$ together with a sub-NSM $\hat{P}$. The transitions in $\hat{P}$ are
   referred to as the \textit{real} transitions, while those in $P - \hat{P}$ are referred to as the \textit{virtual} transitions. $(P, \hat{P})$
is referred to as a \textit{logic module}.

2. $C$, the output (actuator) component, is an equivalence relation on $\Sigma$ representing a control mask.
3. \( M_i \), the input (sensor) component, is an equivalence relation on \( \Sigma \) representing an observation mask.

The \textit{generated language} of the process object \( ((P, \hat{P}), C, M) \) is defined to be the generated language \( L(\hat{P}) \) of its real part.

Given \( \sigma \in \Sigma \), we use \( C(\sigma) \) and \( M(\sigma) \) to denote the corresponding equivalence classes containing \( \sigma \). If \( \Sigma' \) is a set of events, then \( C(\Sigma') \) and \( M(\Sigma') \) are unions of the equivalence classes of the events in \( \Sigma' \). An event is said to be \textit{completely uncontrollable} (respectively, \textit{completely unobservable}) if it belongs to the equivalence class of \( \epsilon \) of mask \( C \) (respectively, of mask \( M \)). Two events are said to be \textit{control-equivalent} (respectively, \textit{observation-equivalent} or indistinguishable) if they belong to the same equivalence class of \( C \) (respectively, of \( M \)). An event is said to be \textit{completely observable} (respectively, \textit{completely controllable}) if its \( C \)-equivalence class (respectively, \( M \)-equivalence class) is a singleton. If every equivalence class with the possible exception of the class of \( \epsilon \) is a singleton, then we refer to the mask as a \textit{natural projection}. For any \( A \subseteq \Sigma \), we use \( \pi_A \) to denote the natural projection for which the equivalence class of \( \epsilon \) is \( \{ \epsilon \} \cup (\Sigma - A) \). If \( A = \Sigma \), we use \( I \) in place of \( \pi_\Sigma \) and refer to it as the \textit{identity mask}.

To clarify why it is necessary to distinguish between real and virtual transitions in the logic component of a process object, we give an informal “preview” of the interconnection operator of \textit{masked composition}; the formal definition together with an illustrative example is given in §3. Masked composition is a generalized synchronization of processes that interact through their control and observation interfaces. Let \( \left( (P_i, \hat{P}_i), C_i, M_i \right), \ i = 1, \ldots, n \) denote a collection of process objects defined over a common event set \( \Sigma \). Their masked composition is a process object \( ((P, \hat{P}), C, M) \). The real transitions—i.e., those in \( \hat{P} \)—are determined by a 3-step synchronization protocol:

1. **Enablement:** Each constituent process \( P_i \) “broadcasts” the set \( \Sigma_{P_i}(x_i) \) of events (together with the null event \( \epsilon \)) that are possible in its current state. This broadcast is “filtered” by its control mask \( C_i \), so the environment of \( P_i \) actually receives the set \( C_i(\Sigma_{P_i}(x_i)) \) consisting of all events that are either completely uncontrollable to the process or are control-equivalent to an event that is possible in its current state.

   The \textit{enabled} event set \( \Sigma_{P}(x), \ x = (x_1, \ldots, x_n) \), for the composite process is the intersection of these sets—i.e., \( \cap_{i=1}^{n} C_i(\Sigma_{P_i}(x_i)) \).

2. **Generation:** A constituent process \( P_i \) can \textit{generate} an event \( \sigma \) provided \( \sigma \in \Sigma_{P_i}(x_i) \cap \Sigma_{P}(x) \)—i.e., \( \sigma \) is enabled in \( P \) and \( P_i \) can execute a \textit{real} transition on \( \sigma \) in its current state.

3. **Response:** An event \( \sigma \) generated by \( P_i \) is broadcast to each of the other constituent processes \( P_j \). \( P_j \) receives this broadcast filtered by its observation mask \( M_j \). If \( M_j(\sigma) \cap \Sigma_{P_j}(x_j) \neq \emptyset \), \( P_j \) \textit{responds} by synchronously executing an event \( \sigma' \) from this set. If the set contains more than one element, the choice of \( \sigma' \) is nondeterministic; if the set is empty, \( P_j \) does not participate in the transition. In any case, the transition in the composite system is labeled only by the generated event \( \sigma \), not by the response event \( \sigma' \).

There is a subtle feature in the definition of masked composition that is omitted from this informal description, namely that \( P_j \) can execute a completely unobservable event possible in its current state as a response event.
when no event has been generated by the remaining constituent processes. The interpretation is that $P_j$ cannot distinguish such an event from the null event $\epsilon$.

We can now justify the need for distinguishing between real and virtual transitions.

**Example 1** Let $\Sigma = \{a, a_1, a_2\}$. Let $P_i = \hat{P}_i$, $i = 1, 2$ denote a deterministic process that can execute $a_i$ and then deadlock. Suppose that $C_i(a_i) = C_i(a)$, $i = 1, 2$. Then the enabled event set in the initial state of the masked composition of the two processes is

$$C_1(\{a_1\}) \cap C_2(\{a_2\}) = \{a\}.$$

Since neither constituent process can generate the event $a$ in its initial state, no events can be generated in the initial state of the composite system—i.e., the composite system contains no real transitions. However, if the two constituent processes are composed with a process $P_0 = \hat{P}_0$ that can execute $a$ and deadlock, then the resulting composite system has a real transition on $a$ in its initial state generated by the constituent process $\hat{P}_0$. We need a virtual transition on $a$ in the composition of $P_1$ and $P_2$ to represent the fact that these processes can enable $a$ if interconnected to a process that is able to generate $a$. (Another way to view this is that without virtual transitions, masked composition could not be associative.) Thus, even if we start with constituent processes that have no virtual transitions—i.e., for which $P_i = \hat{P}_i$—virtual transitions can arise when the processes are interconnected.

Next we define the *augmentation* of an NSM $P$, denoted $P^{CM}$. The purpose of augmentation is to permit us to represent the masked composition operation in terms of the strict synchronous composition (SSC) operation. $P^{CM}$ is obtained from $P$ by adding those transitions to $P$ in each state which are required to obtain a new NSM that satisfies the following properties:

**P1**: If $\sigma$ is defined in state $x$, then every event in $C(\sigma)$ must be defined in $x$.

**P2**: Every event in $C(\epsilon)$ must be defined in every state $x$.

**P3**: If $\sigma_1$ and $\sigma_2$ are defined in $x$ and $M(\sigma_1) = M(\sigma_2) \neq M(\epsilon)$, then $\sigma_1$ and $\sigma_2$ must have the same set of successor states.

**P4**: If $\sigma$ is defined in $x$ and $M(\sigma) = M(\epsilon)$, then every successor state under $\sigma$ must also be a successor state under $\epsilon$, and there must be a self-loop on $\sigma$ at $x$.

If $P^{CM} = P$, then we say that $P$ is a $(C, M)$-invariant process. Thus, $P$ is a $(C, M)$-invariant process if $P$ itself satisfies properties P1-P4.

In general, $P^{CM}$ can be nondeterministic even if $P$ is deterministic. If $P$ is deterministic, $P^{CM}$ will be deterministic if and only if every transition labeled by an event in $M(\epsilon)$ is a self-loop, and whenever $M(\sigma) = M(\sigma')$ and $\delta_P(x, \sigma), \delta_P(x, \sigma')$ are both nonempty, then $\delta_P(x, \sigma) = \delta_P(x, \sigma')$.

The definition of the augmentation of an NSM is extended in a trivial way to define the augmentation of a logic module: Given a logic module $(P, \hat{P})$, its augmentation with respect to the control and observation masks $(C, M)$ is the logic module

$$(P, \hat{P})^{CM} := (P^{CM}, \hat{P}).$$
We will obtain a characterization of the set of languages that can be generated by \((C, M)\)-invariant processes for given control and observation masks. This leads us to introduce the notion of \((C, M)\)-closed languages.

**Definition 2** Let \(K\) be a language over \(\Sigma\) and let \((C, M)\) be given control and observation masks. We say that \(K\) is a \((C, M)\)-closed language if the following conditions are satisfied:

CM1 If \(\sigma, \sigma' \in \Sigma\) with \(\sigma' \in C(\sigma)\) and \(s\sigma \in K\), then \(s\sigma' \in K\).

CM2 If \(\sigma, \sigma' \in \Sigma\) with \(\sigma' \in [C(\varepsilon) \cup C(\sigma)] \cap M(\varepsilon)\) and \(s\sigma t \in K\), then \(s(\sigma')^*\sigma t \subseteq K\).

CM3 If \(\sigma, \sigma' \in \Sigma\) with \(\sigma' \in [C(\varepsilon) \cup C(\sigma)] \cap M(\sigma)\) and \(s\sigma t \in K\), then \(s\sigma' t \in K\).

**Remark 1** A useful easily-derived property of a \((C, M)\)-closed language is the following:

CM4 If \(\sigma \in M(\varepsilon)\) and \(s\sigma t \in K\), then \(s\sigma^* t \subseteq K\).

The following lemma is an immediate consequence of the definitions of augmentation, \((C, M)\)-invariant process, and \((C, M)\)-closed language.

**Lemma 1** If \(P\) is a \((C, M)\)-invariant process, then \(L(P)\) is a \((C, M)\)-closed language.

Given a language \(K \subseteq \Sigma^*\), we use \(\overline{CM}(K)\) to denote the collection of all prefix-closed \((C, M)\)-closed superlanguages of \(K\). Then it is clear from Definition 2 that \(\overline{CM}(K)\) is nonempty and closed under arbitrary intersections and arbitrary unions. In particular, this implies that it contains a unique infimal element which we denote by \(K^{CM}\) and refer to as the \((C, M)\)-closure of \(K\).

**Lemma 2** Given a nonempty language \(K\) and control and observation masks \((C, M)\), there exists a \((C, M)\)-invariant process \(P\) such that \(L(P) = K^{CM}\). If \(K\) is regular, then \(P\) can be chosen to be finite-state, i.e., \(K^{CM}\) is also regular.

The constructive proof of this result is omitted; it is available in [15].

The results of Lemmas 1 and 2 can be combined to obtain the following representation theorem that describes exactly the class of languages that can be generated by \((C, M)\)-invariant processes. It provides the foundation for the supervisory control results in §4.

**Theorem 1** Let \(K\) be a nonempty prefix-closed language and let \((C, M)\) be given control and observation masks. There exists a \((C, M)\)-invariant process \(P\) such that \(L(P) = K^{CM}\). If \(K\) is regular, then \(P\) can be chosen to be finite-state.

We now consider the class of languages that can be generated by deterministic \((C, M)\)-invariant processes.

**Definition 3** A process object \(((\hat{P}, \hat{P}), C, M)\) is called a deterministic process object if \(P^{CM}\) is a deterministic state machine.

**Definition 4** A language \(K\) is called a deterministic \((C, M)\)-closed language if for each \(s \in \Sigma^*\), \(\sigma \in \Sigma\) such that \(s\sigma \in K\), it satisfies conditions CM1, CM2, CM3, and the following additional condition:
CMD4 If $\sigma, \sigma' \in \Sigma$ with $\sigma' \in M(\sigma)$ and $s\sigma, s\sigma' \in K$, then $s\sigma' \in K$; i.e., $s\sigma$ and $s\sigma'$ are Nerode equivalent relative to $K$.

The following results relate the concept of deterministic $(C, M)$-closed language to controllability and observability. Recall from [10] that given a prefix-closed language $H$, $K$ is said to be $(H, C)$-controllable if

$$s\sigma \in K, \sigma' \in C(\sigma), s\sigma' \in H \Rightarrow s\sigma' \in K,$$

where the traditional definition of controllability has been naturally extended to include nonprojection-type control masks. It follows that $K$ is $(\Sigma^*, C)$ controllable if and only if condition CM1 holds.

$K$ is said to be $(H, M)$-observable [9] if

$$s, s' \in K, M(s') = M(s), s' \sigma \in H \Rightarrow s' \sigma \in K.$$

**Lemma 3** $K$ is $(\Sigma^*, M)$-observable if and only if it satisfies the condition CMD4.

The result of Lemma 3 can be used to prove the following expected characterization of deterministic $(C, M)$-closed languages.

**Theorem 2** $K$ is a deterministic $(C, M)$-closed language if and only if it is both $(\Sigma^*, C)$-controllable and $(\Sigma^*, M)$-observable.

**Lemma 4** If $P$ is a deterministic $(C, M)$-invariant process, then $L(P)$ is a deterministic $(C, M)$-closed language.

Given a language $K \subseteq \Sigma^*$, we use $\overline{CMD}(K)$ to denote the collection of all prefix-closed deterministic $(C, M)$-closed superlanguages of $K$. Since controllability and observability of prefix-closed languages are preserved under arbitrary intersections, $\overline{CMD}(K)$ is closed under arbitrary intersections. However, since observability of prefix-closed languages is not preserved under arbitrary unions, it follows that $\overline{CMD}(K)$ is not closed under arbitrary unions. This is in contrast to the $(C, M)$-closed languages. We use $K^{CMD}$ to denote the infimal prefix-closed deterministic $(C, M)$-closed superlanguage of $K$, and refer to it as the deterministic $(C, M)$-closure of $K$.

**Lemma 5** Given a nonempty language $K$ and control and observation masks $(C, M)$, there exists a deterministic $(C, M)$-invariant process $P$ such that $L(P) = K^{CMD}$. If $K$ is regular, then $P$ can be chosen to be finite-state, i.e., $K^{CMD}$ is also regular.

The results of Lemmas 4 and 5 can be combined to obtain the following representation theorem that describes exactly the class of languages that can be generated by deterministic $(C, M)$-invariant processes. It provides the foundation for the deterministic supervisory control results in §4.

**Theorem 3** Let $K$ be a nonempty prefix-closed language and let $(C, M)$ be given control and observation masks. There exists a deterministic $(C, M)$-invariant process $P$ such that $L(P) = K$ if and only if $K$ is a deterministic $(C, M)$-closed language. If in addition $K$ is regular, then $P$ can be chosen to be finite-state.
Remark 2 Whenever the observation mask refines the control mask, i.e., $M(\sigma) \subseteq C(\sigma)$ for each $\sigma \in \Sigma$, then CM3 is equivalent to CMD4; consequently, every $(C, M)$-closed language is also a deterministic $(C, M)$-closed language. In particular, this holds when both $C$ and $M$ are projection-type masks and $M(\epsilon) \subseteq C(\epsilon)$—i.e., when every completely controllable event is completely observable. While in general we have

$$K^{CM} \subseteq K^{CMD},$$

in this special case the two closures coincide.

3 Masked Composition of Process Objects

This section defines the operation of masked composition which is the interconnection mechanism of process objects that we use for achieving control as well as for interaction among the components of a plant. We begin by defining a synchronous product of logic modules. Recall that in the SSC of NSM’s, an event $\sigma \in \Sigma$ is enabled in the current state provided it is enabled in the current state of each constituent process. If this is the case and $\sigma$ occurs, it is synchronously executed by each process. An $\epsilon$-transition is possible in the current state of the SSC provided it is possible in the current state of at least one constituent process. If $\epsilon$ is possible in the current state of only one process, it is executed by only that constituent process; the state of the other constituent process remains unchanged. On the other hand, if $\epsilon$ is possible in the current state of each process, then it may be executed either synchronously by each process, or asynchronously by one of the processes.

Definition 5 Let $(P_i, \hat{P}_i)$ ($i = 1, 2$) be logic modules over $\Sigma$ with $P_i = (X_{P_i}, \Sigma, \delta_{P_i}, X_{P_i}^0)$, and let $\delta_{P_i}$ denote the transition function for the sub-NSM $\hat{P}_i$. The synchronous product of the logic modules is denoted by

$$(P, \hat{P}) := (P_1, \hat{P}_1)\parallel (P_2, \hat{P}_2)$$

and is defined as follows:

- $P := P_1\parallel P_2$, the standard strict synchronous composition (SSC) of $P_1, P_2$.

- The transition function of $\hat{P}$ is defined by

$$\forall \sigma \in \Sigma: \delta_{\hat{P}}((x_1, x_2), \sigma) = [\delta_{P_1}(x_1, \sigma) \times \delta_{P_2}(x_2, \sigma)] \cup [\delta_{P_1}(x_1, \sigma) \times \delta_{P_2}(x_2, \sigma)]$$

$$\delta_{\hat{P}}((x_1, x_2), \epsilon) = [\delta_{P_1}(x_1, \epsilon) \times \{x_2\}] \cup [(x_1) \times \delta_{P_2}(x_2, \epsilon)] \cup$$

$$[\delta_{P_1}(x_1, \epsilon) \times \delta_{P_2}(x_2, \epsilon)]$$

Proposition 1 The synchronous product of logic modules is associative:

$$(P_1, \hat{P}_1)\parallel (P_2, \hat{P}_2)\parallel (P_3, \hat{P}_3) = (P_1, \hat{P}_1)\parallel ((P_2, \hat{P}_2)\parallel (P_3, \hat{P}_3))$$

Definition 6 Given process objects $((P_0, \hat{P}_0), C_0, M_0)$ and $((P_1, \hat{P}_1), C_1, M_1)$, their masked composition (MC), denoted $((P_0, \hat{P}_0), C_0, M_0) \parallel ((P_1, \hat{P}_1), C_1, M_1)$, is defined to be the process object

$$((P_0, \hat{P}_0)^{C_0M_0}\parallel (P_1, \hat{P}_1)^{C_1M_1}, C_0 \land C_1, M_0 \land M_1),$$
where $\|$ denotes the synchronous product of logic modules, and $\wedge$ denotes the conjunction of equivalence relations—i.e., the equivalence relation whose equivalence classes are the intersections of the equivalence classes from the original equivalence relations.

Thus the NSM $P := P_0^{C_0 \cdot M_0} P_1^{C_1 \cdot M_1}$ of the composition is obtained by SSC of the augmentations of the constituent processes. It contains both the real and virtual transitions. The real transitions in $P$—i.e., the transitions in $\hat{P}$—are those transitions in $P$ that can actually be generated by real transitions in the constituent processes. The real transitions can be interpreted as arising via the 3-step synchronization protocol described in §2. It is clear that the definitions of augmentation and masked composition extend in an obvious way to the more general situation where the control and observation masks are allowed to be state-dependent.

**Example 2** Let $\Sigma = \{a, b\}$, and let $P_0 = \hat{P}_0$, $P_1 = \hat{P}_1$ be as shown in Figure 1. Suppose that the $C_0$ equivalence classes are $\{\epsilon, b\}$, $\{a\}$; $M_0 = I$; the $C_1$ equivalence classes are $\{\epsilon, a\}$, $\{b\}$; the $M_1$ equivalence classes are $\{\epsilon\}$, $\{a, b\}$. The augmented processes $P_0^{C_0 \cdot M_0}$, $P_1^{C_1 \cdot M_1}$ are shown in Figure 1. From Definition 6, it is straightforward to verify that the logic component of the masked composition of $((P_0, P_0), C_0, M_0)$ and $((P_1, P_1), C_1, M_1)$ is the NSM $P$ depicted in Figure 1, and that every transition in $P$ is real—i.e., $\hat{P} = P$. Note that if $P_0$ generates $a$, then $P_1$ synchronously executes $b$ since it cannot distinguish between $a, b$. On the other hand, if $P_1$ generates $b$, there is no observation-equivalent event that $P_0$ can execute, so it does not participate in this transition. Since $P_0$ remains in its initial state, it is then able to generate the event $a$. Consequently, the trace $ba$ can be executed by the composite system, while the trace $ab$ cannot be executed.

**Remark 3** The prioritized synchronous composition (PSC) [6, 13] of processes $P_0$ and $P_1$ with priority sets $A$ and $B$ corresponds to the special case of masked composition where $M_0 = I$, $M_1 = I$—i.e., in each process, every observation equivalence class is a singleton (all events are completely observable to each process), and the control masks $C_0$, $C_1$ are the natural projections $\pi_A$, $\pi_B$ respectively. The full synchronous composition operator in concurrency theory [7, p. 68] corresponds to the special case of PSC in which the alphabet (event set) of each process coincides with its priority set. The effect of the interleaving operator [7, p. 119] can also be obtained using masked composition as illustrated in Example 3.

Finally, the traditional supervisory control model [10, 9, 1] corresponds to the special case where $C_0 = M_0 = I$ (all events are completely controllable and completely observable to the plant), $C_1 = \pi_{\Sigma_c}$, and $M_1 = \pi_{\Sigma_o}$, where $\Sigma_c$, $\Sigma_o$ denote the set of controllable events and set of observable events respectively.

**Example 3** In this example, we demonstrate how interleaving composition [7, p. 119] can be modeled using masked composition. Suppose $P_0 = \hat{P}_0$ is a process that can execute $ab$ and deadlock, while $P_1 = \hat{P}_1$ is a process that can execute $a$ and deadlock. We first subscript each event with the index of the process in which it occurs to obtain the modified processes $P_0, P_1$ which have disjoint alphabets $\Sigma_0 = \{a_0, b_0\}$, $\Sigma_1 = \{a_1\}$. Now take the masked composition using $C_1 = M_1 = \pi_{\Sigma_1}$. The resulting process $P = \hat{P}$ has generated language $L(P) = pr\{a_0b_0a_1, a_0a_1b_0, a_1a_0b_0\}$ and hence represents pure interleaving of the constituent processes. If we
do not want to distinguish events that differ only in their subscripts when we compose with other processes, we simply arrange for the masks of the other processes to identify such events.

The next theorem establishes the associativity of masked composition; it follows from associativity of synchronous product and intersection.

**Theorem 4** Masked composition is associative.

The next result describes conditions under which a masked composition contains no virtual transitions. A sufficient condition is that for each event \( \sigma \), there is some process for which \( \sigma \) is completely controllable and completely observable, and for which every transition labeled by \( \sigma \) is real.

**Theorem 5** Let \(((P_i, \hat{P}_i), C_i, M_i) \ (i = 0, \ldots, n)\) be process objects, and let

\[
((P, \hat{P}), C, M) := ((P_0, \hat{P}_0), C_0, M_0) \parallel \cdots \parallel ((P_n, \hat{P}_n), C_n, M_n).
\]

Suppose that \( \forall \sigma \in \Sigma : \) there exists \( i \) such that

(A1) \( [C_i(\sigma) = \{\sigma\}] \land [M_i(\sigma) = \{\sigma\}] \).

(A2) Every transition on \( \sigma \) in \( P_i \) is also a transition in \( \hat{P}_i \).

Then \( \hat{P} = P = P_0^{C_0,M_0} \parallel \cdots \parallel P_n^{C_n,M_n} \).

A consequence of Theorem 5 is the following corollary which states that if a system is obtained by interconnecting a finite collection of process objects, then the language of the system can be obtained by intersecting the languages of the augmented processes provided each event is completely controllable and completely observable to at least one process which has no virtual transitions labeled by that event. This corollary can be regarded as a generalization of the language intersection result for the PSC of two processes when each event belongs to the priority set of at least one process [13, Proposition 4].

**Corollary 1** Under Assumptions A1 and A2, the generated language of the composed system is given by

\[
L(\hat{P}) = L(P) = \bigcap_{i=0}^n L(P_i^{C_i,M_i}).
\]

**Remark 4** The result of Theorem 5 can be specialized to the situation in the Ramadge-Wonham theory [11]. In that framework, every event is generated by the plant and hence is completely controllable and completely observable to the plant—i.e., if we let the process object with index zero represent the plant, the process object with index one the supervisor, and the composed process object the controlled system, then \( C_0 = I \) and \( M_0 = I \).

Also, if \(((P, \hat{P}), C, M) = ((P_0, P_0), I, I) \parallel ((P_1, P_1), C_1, M_1)\), then \( C = I \), \( M = I \), and \( \hat{P} = P = P_0^{C_0,M_0} \parallel P_1^{C_1,M_1} \). In particular, \( L(\hat{P}) = L(P_0) \cap L(P_1^{C_1,M_1}) \).

### 4 Supervisory Control

Given a process object \(((P_0, P_0), C_0, M_0)\) representing a plant, control and observation masks \((C_1, M_1)\) for a supervisor to be constructed, and a language \( K \), we begin by finding necessary and sufficient conditions for
there to exist a process object \(((P_1, P_1), C_1, M_1)\), representing a supervisor, such that the generated language of the masked composition of plant and supervisor is equal to \(K\). Let \(((P, \hat{P}), C, M)\) denote the process object representing the composed system. Then the requirement is that \(L(\hat{P}) = K\). We focus on the special case where each event is assumed to be both completely controllable and completely observable to either the plant or the supervisor—i.e., Assumption A1 is satisfied—so that the language intersection result of Corollary 1 holds. Thus under this assumption the synthesis problem is to construct \(P_1\) such that

\[
L(P_0^{C_0M_0}) \cap L(P_1^{C_1M_1}) = K. \tag{1}
\]

Hence only sublanguages of \(L(P_0^{C_0M_0})\) can be obtained as the generated language of the controlled system.

**Definition 7** Given a plant \(((P_0, P_0), C_0, M_0)\), control and observation masks \((C_1, M_1)\), and a language \(K \subseteq L(P_0^{C_0M_0})\), \(K\) is said to be \(((P_0, P_0), C_0, M_0)\)-relatively \((C_1, M_1)\)-closed if it satisfies the following condition:

\[
L(P_0^{C_0M_0}) \cap K^{C_1M_1} = \overline{K}. \tag{2}
\]

When the plant process object is clear from the context, we will simply refer to such a language as being relatively \((C_1, M_1)\)-closed. It is clear from this definition that \(K\) is relatively \((C_1, M_1)\)-closed if and only if \(\overline{K}\) is relatively \((C_1, M_1)\)-closed.

**Theorem 6** Consider a plant \(((P_0, P_0), C_0, M_0)\), control and observation masks \((C_1, M_1)\), a language \(K \subseteq L(P_0^{C_0M_0})\), and suppose that Assumption A1 holds. Then there exists an NSM \(P_1\) such that the generated language of the controlled system is \(K\) if and only if \(K = \overline{K} \neq \emptyset\) is relatively \((C_1, M_1)\)-closed. If in addition \(K\) is regular, then \(P_1\) can be chosen to be finite-state.

**Remark 5** Suppose the given specification language \(K\) does not satisfy the relative \((C_1, M_1)\)-closure condition (2). It is easy to show that there is always a supremal relatively \((C_1, M_1)\)-closed sublanguage \(K^\uparrow\) of \(K\). If \(K^\uparrow \neq \emptyset\), then the supervisor process object can be chosen so that the generated language of the controlled system is \(K^\uparrow\), and hence contained in the specification language \(K\).

In the remainder of this section, we consider supervisory control under the additional restriction that the supervisor process object be deterministic. This means that \(P_1^{C_1M_1}\) must be a deterministic state machine. (Recall Definition 3.) We continue to assume that each event is both completely controllable and completely observable to either the plant or the supervisor.

**Definition 8** Given a plant \(((P_0, P_0), C_0, M_0)\), control and observation masks \((C_1, M_1)\), and a language \(K \subseteq L(P_0^{C_0M_0})\), \(K\) is said to be \(((P_0, P_0), C_0, M_0)\)-relatively deterministic \((C_1, M_1)\)-closed language if it satisfies the following condition:

\[
L(P_0^{C_0M_0}) \cap K^{C_1M_1D} = \overline{K}. \tag{3}
\]

When the plant process object is clear from the context, we will simply refer to such a language as being relatively deterministic \((C_1, M_1)\)-closed. It is clear from this definition that \(K\) is relatively deterministic \((C_1, M_1)\)-closed if and only if \(\overline{K}\) is relatively deterministic \((C_1, M_1)\)-closed. The following theorem states that relatively
deterministic \((C_1, M_1)\)-closure is necessary and sufficient for the existence of a deterministic supervisor achieving a desired closed-loop language, and relates relatively deterministic \((C_1, M_1)\)-closure to controllability and observability with respect to the language of the augmented plant.

**Theorem 7** Consider a plant \(((P_0, P_0), C_0, M_0)\), control and observation masks \((C_1, M_1)\), and a nonempty prefix-closed language \(K \subseteq L(P_0^{C_0 M_0})\). Suppose Assumption A1 holds. Then the following are equivalent:

1. There exists a deterministic supervisor \(((P_1, P_1), C_1, M_1)\) such that the closed-loop generated language is \(K\).

2. \(K\) is a relatively deterministic \((C_1, M_1)\)-closed language.

3. \(K\) is \((L(P_0^{C_0 M_0}), C_1)\)-controllable and \((L(P_0^{C_0 M_0}), M_1)\)-observable.

If in addition \(K\) is regular, then \(P_1\) can be chosen to be finite-state.

Since the class of deterministic \((C, M)\)-closed languages is not generally closed under union, there need not be a supremal relatively deterministic \((C_1, M_1)\)-closed sublanguage of a given language \(K\).

The next result shows that under certain conditions, the existence of a supervisor achieving a prescribed generated language for the controlled system is equivalent to the existence of a deterministic supervisor giving that language. This result follows from Theorems 6, 7 and Remark 2.

**Theorem 8** Consider a plant \(((P_0, P_0), C_0, M_0)\), control and observation masks \((C_1, M_1)\), and a nonempty prefix-closed language \(K \subseteq L(P_0^{C_0 M_0})\). Suppose Assumption A1 holds. If \(M(\sigma) \subseteq C(\sigma)\) for each \(\sigma \in \Sigma\), then there exists a supervisor \(((P_1, P_1), C_1, M_1)\) such that the closed-loop generated language is \(K\) if and only if there exists a deterministic supervisor \(((P_1, P_1), C_1, M_1)\) such that the generated language of the controlled system is \(K\).

**Remark 6** Theorem 8 generalizes a previous result of Shayman-Kumar [13, Theorem 5]. Although stated using trajectory models rather than NSM’s, the earlier result is essentially the special case of Theorem 8 in which all masks are natural projections and the observation mask of the plant is trivial—i.e., \(M_0 = I\).

**References**


