TECHNICAL RESEARCH REPORT

A New Framework for Supervisory Control of Discrete Event Systems

by M.A. Shayman, R. Kumar

T.R. 95-72

Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

A New Framework for Supervisory Control of Discrete Event Systems [1]

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 40506-0046
Email: kumar@engr.uky.edu

August 9, 1995

[1]This research was supported in part by the Center for Robotics and Manufacturing, University of Kentucky, and in part by the National Science Foundation under grants CDR-8803012, EEC-94-02384, ECS-9312587 and ECS-9409712.

Abstract

We propose a new framework for supervisory control design for discrete event systems. Some of the features of the proposed approach are: (i) By associating control and observation capabilities and limitations with the plant as well as the supervisor, it models *reactive* systems, and also treats plant and supervisory processes in a symmetric way. (ii) By introducing a single general interconnection operation, called *masked composition*, it permits open-loop as well as closed-loop control. (iii) By viewing the uncontrollability of events as corresponding to a projection-type control mask, and considering more general nonprojection-type control as well as observation masks, it treats the controllability and observability of events in a unified way. (iv) It applies to both deterministic and nondeterministic plant models and supervisory design. The sublanguages of a given language that are realizable under control are closed under union. Hence, the supremal realizable sublanguage always exists. In addition, it yields conditions under which existence of a nondeterministic supervisor implies existence of a deterministic supervisor. (v) By encapsulating control and observation masks with process logic to form *process objects*, and using a single type of interconnection operator to build complex process objects out of simpler component process objects, it provides a foundation for an *object-oriented* approach to discrete event control.

Keywords: discrete event system, supervisory control, nondeterministic systems, object-oriented systems, partial observation

1 Introduction

The standard supervisory control framework is ill-suited for modeling *reactive systems.* In traditional supervisory control [27, 20], the plant generates all events. The plant is a "closed system"; the only intervention that is possible is from the supervisor disabling some controllable events. In models with "command events" [1] or "driven events" [10], the supervisor can initiate some events, but the controlled system of the plant and supervisor is again a closed system. In a reactive system, the controlled plant is not a closed system. It interacts with an environment that can initiate events that may be observed by the plant, and whose occurrence may or may not require the participation of the plant. For example, the radar screen of a fighter aircraft may indicate the presence of another aircraft, but may not be able to identify it as friend or foe. Thus, the *plant*–as opposed to the supervisor–observes an event initiated by the environment. Furthermore, it has partial observation since it cannot distinguish between the two events corresponding to friend and foe. To model reactive systems, not only the supervisor, but the plant as well, must have control and observation capabilities and limitations. This is one of the features of the proposed new approach.

The standard supervisory control framework of Ramadge-Wonham [27] permits the control input from the supervisor to be changed only when an event is observed. In other words, non-constant open-loop control is not permitted. Requiring the control input to be constant between observed events is analogous to requiring that zero-order hold be used in sampled data systems. However, it is more realistic in that setting since the sampling rate can be chosen to be high. In general, the frequency of observable events is not a design parameter. Consider the problem of allocating a shared communication channel between two users under the assumption that the controller cannot observe the messages submitted by the users. Suppose that the controller can disable the access of either user, and the specification is that there be no interleaving of the messages from the two users. Since there are no observable events, the Ramadge-Wonham framework would require that the control input be forever constant. Consequently, one user would be permanently denied access to the channel. In contrast, by introducing the interconnection operation called *masked composition*, our framework permits an "open-loop" transition in the supervisor that allows the access to be changed from one process to the other. Masked composition generalizes many of the interconnection operators including strict synchronous composition [12], prioritized synchronous composition [10], etc., while maintaining the desired property of associativity. It can also be used to model the suppression ("hiding") of events.

In nearly all the work on supervisory control, the event set is partitioned into a set of controllable events and a set of uncontrollable events. The supervisor can disable any subset of controllable events. Equivalently, this can be viewed as having a projection-type *control mask.* While Golaszewski-Ramadge [9] have introduced a framework in which arbitrary sets of events may be disabled, only limited results can be obtained in this very general setting. Similarly, in the work of Holloway-Krogh on controlled Petri nets [13], arbitrary control patterns can be applied by appropriately selecting the connectivity between the control

input places and the transitions of the net. Although observation has been studied using non-projection masks [3], the same has apparently not been done for control. Yet it is entirely natural to have groups of events which must be simultaneously disabled or not disabled at all. For example, a machine may perform two operations and be subject only to on-off control. Thus, to disable one operation requires shutting the machine off, thereby disabling the other operation. Also, in the work of Stiver-Antsaklis [33] on hybrid control systems, the command from a discrete event supervisor to a discrete event plant (that has been extracted out of a continuous-time plant by way of aggregation) is able to enable/disable a group of events simultaneously. Our framework includes arbitrary control masks, as well as arbitrary observation masks.

A discrete event plant can have unmodeled dynamics. Thus, knowledge of the current state and next event may not uniquely determine the successor state. In addition, there may be state transitions that occur without being accompanied by the occurrence of a modeled event. Such plants can be modeled by *nondeterministic state machines with ϵ-moves* (NSM's) [14]. If there are *driven events*–i.e., supervisor commands that are not always executable by the plant–then control design based on language models is inadequate and design based directly on the NSM models or on trajectory models is needed [10, 11]. In [31, 21, 22], necessary and sufficient conditions for the solution to the supervisory control problem are obtained under the restrictions that (1) the control and observation masks are natural projections and (2) either every controllable event is observable [31] or only deterministic supervisors are permitted [22]. Using the new framework, we are able to solve the supervisory control problem for nondeterministic plants without either of these restrictions. Moreover, when the supervisor is not constrained to be deterministic, then the sublanguages of a given language that are realizable as controlled behavior are closed under union; consequently, the supremal realizable sublanguage always exists. In addition, we obtain conditions under which the existence of a nondeterministic supervisor meeting a specification implies the existence of a deterministic supervisor meeting the same specification.

It is highly desirable to develop an object-oriented approach to supervisory control of discrete event systems. Such an approach would offer the promise of software reusability [4]. Efforts to develop such an approach for continuous variable control systems have already begun [16]. The basic goal is to develop the capacity to build modules (objects) in which the vast majority of the logic is standardized (inherited from its class), and which can be used off-the-shelf and interconnected in different ways to 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.

In our approach, 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. In contrast to the approach in [8, 7], our framework can accomodate specifications that limit concurrency, and

the associativity of masked composition makes our approach suitable for modular control.

The organization of the remainder of the paper is as follows: In §2, we describe the encapsulation of systems together with their control and observation interfaces as *process objects*. The special case of *deterministic process objects* is also examined. In §3, the operation of *masked composition* is defined. §4 contains the main results concerning supervisory control theory in the framework of process objects and masked composition. The special case of *deterministic supervisors* is also described. Finally, §5 contains several examples illustrating the importance of both nondeterministic supervision and open-loop control, and the application of the process object/masked composition paradigm to reactive systems and object-oriented design.

Most of the results in this paper first appeared in abridged form in [32, 30].

2 Process Object Representation

A *nondeterministic state machine* (with ϵ-moves) is represented by a four-tuple $P := (X_P, \Sigma, \delta_P, X_P^0)$ where X_P denotes the state space of P, Σ denotes the event set of P, $\delta_P : X_P \times \overline{\Sigma} \rightarrow 2^{X_P}$, where $\overline{\Sigma} := \Sigma \cup \{\epsilon\}$, denotes the nondeterministic transition function of P, and $X_P^0 \subset X_P$ denotes the nonempty set of initial states of P. [1]

We use the symbol x for denoting a state, and σ for denoting an event including the hidden event denoted as ϵ. Thus $\delta_P(x, \sigma)$ denotes the set of states reachable from state x executing the event $\sigma \in \overline{\Sigma}$. A triple (x, σ, x') is called a transition if $x' \in \delta_P(x, \sigma)$; it is said to be a silent transition if $\sigma = \epsilon$. A *deterministic state machine* is a nondeterministic state machine satisfying the following conditions: (i) X_P^0 contains exactly one element; (ii) $\delta_P(x, \epsilon) = \emptyset, \forall x$; (iii) $\delta_P(x, \sigma)$ is either empty or contains one element for each $\sigma \in \Sigma$.

We use δ_P^* to denote the extension of δ from the set of events $\overline{\Sigma}$ to the set of strings Σ^*. Symbols s, t, etc., are used for denoting strings, including the zero length string denoted with a slight abuse of notation as ϵ. 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 ϵ-events. Define $\Sigma_P(x) := \{\sigma \in \overline{\Sigma} | \, \delta_P(x, \sigma) \neq \emptyset\}$–i.e., the set of events (possibly including ϵ) defined in state x; and $\overline{\Sigma}_P(x) := \Sigma_P(x) \cup \{\epsilon\}$. $L(P)$ denotes the *generated language* of P–i.e., $L(P) = \{s \in \Sigma^* | \, \exists x_0 \in X_P^0 \text{ s.t. } \delta_P^*(x_0, s) \neq \emptyset\}$. Given a language $K \subseteq \Sigma^*$, $\overline{K}$ and prK are used to denote its prefix-closure; K is said to be prefix-closed if $\overline{K} = K$. The Nerode equivalence class of $s \in \Sigma^*$ relative to K is denoted by $[s]_K$. If the language is obvious from the context, we freely omit the subscript and use $[s]$ to denote the Nerode class.

In the standard supervisory control framework [27], 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

[1] We permit the NSM P to have a nonempty set of initial states, rather than requiring that there be only one initial state. This is important only if we constrain P to contain no ϵ-transitions. A similar assumption is made by Inan [15].

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) [19]. 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 σ 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 σ, 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 *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 σ 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 *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 σ is an event possible in x_1, x_2, y_1, then σ 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 implementation ill-suited for this purpose. Suppose that we wish to define an *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 *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 configurable 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 *implicitly* as restrictions on the supervisor logic. Our solution is to separate the control and observation limitations from the logic and *encapsulate* them together with the logic to form *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 σ in the current state of a process serves three functions: (1) it *enables* the other process to generate σ if it can do so; (2) it can *generate* σ if the other process enables it; (3) it can

respond to the event σ 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 *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, *virtual* transitions can only enable and respond to events generated by another process. They cannot generate (initiate) events on their own. (See Example 1.)

Definition 1 A *process object* consists of four 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 *real* transitions, while those in $P - \hat{P}$ are referred to as the *virtual* transitions. $(P, \hat{P})$ is referred to as a *logic module.*

2. C, the output (actuator) component, is an equivalence relation on $\overline{\Sigma}$ representing a control mask.

3. M, the input (sensor) component, is an equivalence relation on $\overline{\Sigma}$ representing an observation mask.

The *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 $\sigma \in \overline{\Sigma}$, we use $C(\sigma)$ and $M(\sigma)$ to denote the corresponding equivalence classes containing σ. An event is said to be *completely uncontrollable* (respectively, *completely unobservable*) if it belongs to the equivalence class of ϵ of mask C (respectively, of mask M). Two events are said to be *control-equivalent* (respectively, *observation-equivalent* or indistinguishable) if they belong to the same equivalence class of C (respectively, of M). An event is said to be *completely controllable* (respectively, *completely observable*) if its C-equivalence class (respectively, M-equivalence class) is a singleton. If every equivalence class with the possible exception of the class of ϵ is a singleton, then we refer to the mask as a *natural projection.* For any $A \subseteq \Sigma$, we use π_A to denote the natural projection for which the equivalence class of ϵ is $\{\epsilon\} \cup (\Sigma - A)$. If $A = \Sigma$, we use I in place of π_Σ and refer to it as the *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 *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 $((P_i, \hat{P}_i), C_i, M_i)$, $i = 1, \ldots, n$ denote a collection of process objects defined over a common event set Σ. 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 $\overline{\Sigma}_{P_i}(x_i)$ of events (together with the null event ϵ) 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 $C_i(\overline{\Sigma}_{P_i}(x_i))$ 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 *enabled* event set $\Sigma_P(x)$, $x = (x_1, \ldots, x_n)$, for the composite process is the intersection of these sets–i.e., $\cap_{i=1}^{n} C_i(\overline{\Sigma}_{P_i}(x_i))$.

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

3. **Response:** An event σ 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 *responds* by synchronously executing an event σ' from this set. If the set contains more than one element, the choice of σ' 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 σ, not by the response event σ'. [2]

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. [3] 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. The purpose of augmentation is to permit us to represent the masked composition operation in terms of the strict synchronous

[2] 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 ϵ.

[3] Another way to view this is that without virtual transitions, masked composition could not be associative.

composition (SSC) operation. In order to do this, it is necessary to add transitions to P in each state to obtain a new NSM, denoted P^{CM}, that satisfies the following properties:

P1: If σ 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 σ_1 and σ_2 are defined in x and $M(\sigma_1) = M(\sigma_2) \neq M(\epsilon)$, then σ_1 and σ_2 must have the same set of successor states.

P4: If σ is defined in x and $M(\sigma) = M(\epsilon)$, then σ and ϵ must have the same set of successor states, and there must be a self-loop on σ.

Remark 1 P2 and P4 can be regarded as special cases of P1 and P3, respectively, provided we agree by convention to regard there to be a self-loop on ϵ defined in each state.

Definition 2 Given an NSM P and control and observation masks (C, M), the *augmentation of P with respect to (C, M)*, denoted P^{CM}, is obtained from P by adding for each state x transitions as follows:

1. If $(x, \sigma, x') \in P$ with $x \neq x'$ and $M(\sigma) = M(\epsilon)$, add transitions (x, σ, x) and (x, ϵ, x').
2. If (x, σ_1, x_1), $(x, \sigma_2, x_2) \in P$ with $x_1 \neq x_2$ and $M(\sigma_1) = M(\sigma_2) \neq M(\epsilon)$, add transitions (x, σ_1, x_2) and (x, σ_2, x_1).
3. If $\sigma \notin \overline{\Sigma}_P(x)$, $C(\sigma) \cap \overline{\Sigma}_P(x) \neq \emptyset$ (i.e., σ is completely uncontrollable or control-equivalent to an existing event), and $M(\sigma) \cap \overline{\Sigma}_P(x) = \emptyset$ (i.e., σ is neither completely unobservable nor observation-equivalent to an existing event), add a self-loop transition (x, σ, x).
4. If $\sigma \notin \overline{\Sigma}_P(x)$, $C(\sigma) \cap \overline{\Sigma}_P(x) \neq \emptyset$, $M(\sigma) \neq M(\epsilon)$, and $M(\sigma) \cap \Sigma_P(x) \neq \emptyset$ (i.e., σ is not completely unobservable and is observation-equivalent to an existing event), add transitions $\{(x, \sigma, x') |\, x' \in \delta_P(x, \sigma'),\ \sigma' \in M(\sigma) \cap \Sigma_P(x)\}$.
5. If $\sigma \notin \overline{\Sigma}_P(x)$, $C(\sigma) \cap \overline{\Sigma}_P(x) \neq \emptyset$, $M(\sigma) = M(\epsilon)$, (i.e., σ is completely unobservable), add the self-loop transition (x, σ, x).

The construction in the proof of Lemma 2 requires the following modification of the operation of augmentation.

Definition 3 Given an NSM P and control and observation masks (C, M), the *modified augmentation of P with respect to (C, M)* is the NSM P' obtained from P by applying operations 1,2,4,5 in Definition 2 together with the following substitute for operation 3:

3′. Add a "dump state" x_d with self-loops in x_d on every event in $C(\epsilon)$. If $\sigma \notin \overline{\Sigma}_P(x)$, $C(\sigma) \cap \overline{\Sigma}_P(x) \neq \emptyset$, and $M(\sigma) \cap \overline{\Sigma}_P(x) = \emptyset$, add a transition (x, σ, x_d).

The definition of the augmentation of an NSM is extended in a trivial way to define the augmentation of a logic module:

Definition 4 Given a logic module $(P, \hat{P})$, its augmentation with respect to the control and observation masks (C, M) is the logic module

$$(P, \hat{P})^{CM} := (P^{CM}, \hat{P}).$$

Example 2 Consider the process P shown in Figure 1. P is a deterministic process with $L(P) = pr\{dd, ca, b\}$. Suppose that the C-equivalence classes are $A_0 = \{\epsilon, b\}$, $A_1 = \{a, c\}$, $A_2 = \{d\}$, and the M-equivalence classes are $B_0 = \{\epsilon, c\}$, $B_1 = \{a, b, d\}$. The augmented process P^{CM} is also shown in Figure 1.

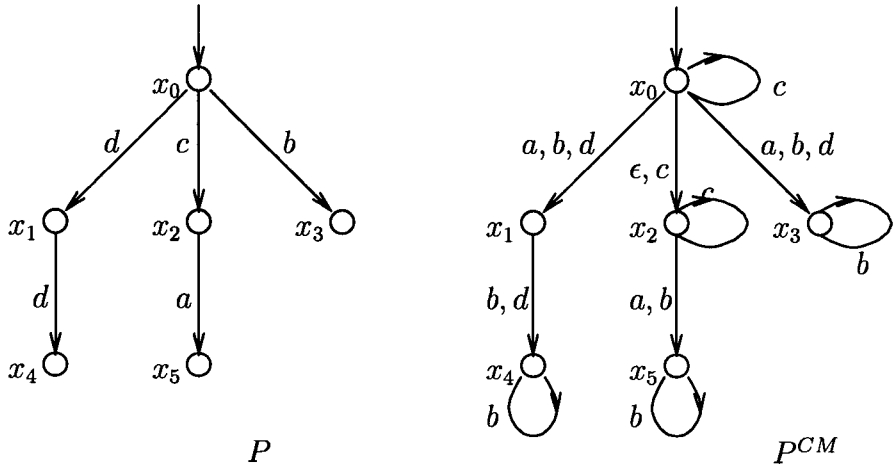

Figure 1: Diagram illustrating Example 2

Remark 2 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 $\delta_P(x, \sigma) = \delta_P(x, \sigma')$.

Definition 5 Given an NSM P and control and observation masks (C, M), we say P is a (C, M)-*invariant process* if $P^{CM} = P$.

The following result is a straightforward consequence of Definitions 2 and 5.

Proposition 1 Given an NSM P and control and observation masks (C, M), P is a (C, M)-invariant process if and only if it satisfies properties P1-P4.

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 6 Let K be a language over Σ 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 $\sigma, \sigma' \in \overline{\Sigma}$ with $\sigma' \in C(\sigma)$ and $s\sigma \in \overline{K}$, then $s\sigma' \in \overline{K}$.

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

CM3 If $\sigma, \sigma' \in \overline{\Sigma}$ with $\sigma' \in [C(\epsilon) \cup C(\sigma)] \cap M(\sigma)$ and $s\sigma t \in \overline{K}$, then $s\sigma' t \in \overline{K}$.

Remark 3 A useful derived property of a (C, M)-closed language is the following:

CM4 If $\sigma \in M(\epsilon)$ and $s\sigma t \in \overline{K}$, then $s\sigma^* t \subseteq \overline{K}$.

To obtain this, suppose $\sigma \in M(\epsilon)$ and $s\sigma t \in \overline{K}$. Applying CM2 with $\sigma' = \sigma$ gives $s\sigma^*\sigma t \subseteq \overline{K}$. Applying CM3 with $\sigma' = \epsilon$ gives $st \in \overline{K}$. So $s\sigma^* t \subseteq \overline{K}$.

Remark 4 The three defining properties of a (C, M)-closed language depend only on the prefix-closure $\overline{K}$ of the language and can be restated in terms of Nerode equivalence classes with respect to $\overline{K}$ [14, p. 65]. Given $s \in \Sigma^*$, let $[s]_{\overline{K}}$ denote the Nerode equivalence class of s; i.e., $[s_1]_{\overline{K}} = [s_2]_{\overline{K}}$ provided that $s_1 t \in \overline{K} \Leftrightarrow s_2 t \in \overline{K}$, $\forall t \in \Sigma^*$. Let $N(\overline{K})$ denote the set of Nerode equivalence classes. We can define a partial order on $N(\overline{K})$ by specifying that $[s_1]_{\overline{K}} \leq [s_2]_{\overline{K}} \Leftrightarrow \{t \in \Sigma^* |\ s_1 t \in \overline{K}\} \subseteq \{t \in \Sigma^* |\ s_2 t \in \overline{K}\}$. The poset $N(\overline{K})$ has an infimal element:

$$\inf N(\overline{K}) = \Sigma^* - \overline{K}.$$

Then the conditions for K to be a (C, M)-closed language are equivalent to the following:

CM1 $\sigma' \in C(\sigma) \Longrightarrow [[s\sigma]_{\overline{K}} = \inf N(\overline{K}) \Leftrightarrow [s\sigma']_{\overline{K}} = \inf N(\overline{K})]$

CM2 $\sigma' \in C(\sigma) \cap M(\epsilon) \Longrightarrow [s\sigma]_{\overline{K}} \leq [s\sigma'\sigma]_{\overline{K}}$

CM3 $\sigma' \in [C(\epsilon) \cup C(\sigma)] \cap M(\sigma) \Longrightarrow [s\sigma]_{\overline{K}} \leq [s\sigma']_{\overline{K}}$

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 6 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.

Proof: Let R be any NSM with the following property:

$$\forall x \in X_R,\ \sigma, \sigma' \in \Sigma:\ \ [\sigma, \sigma' \notin C(\epsilon)] \wedge [\sigma, \sigma' \in \Sigma_R(x)] \Rightarrow C(\sigma) = C(\sigma'). \tag{1}$$

In other words, whenever two events that are not completely uncontrollable are defined in the same state, they must be control-equivalent.

Let Q be an NSM that satisfies (1). Let P be the modified augmentation of Q with respect to (C, M). (See Definition 3.) We claim that

$$L(P) \subseteq [L(Q)]^{CM}. \tag{2}$$

To establish this, suppose that a transition is added to Q via one of the five operations of modified augmentation, and let Q' be the resulting NSM. We claim that

$$L(Q') \subseteq [L(Q)]^{CM}. \tag{3}$$

For a type 1 operation, (3) follows from CM4. For a type 2 operation, (3) follows from CM3. For a type 3′ operation, (3) follows from CM1. For a type 4 operation, (3) follows from CM3. For a type 5 operation, (3) follows from CM2. Thus, (3) holds. Furthermore, Q' also satisfies (1), so the argument can be repeated for the next transition added via modified augmentation. Continuing in this way yields (2).

It is straightforward to verify that an NSM obtained by modified augmentation is a (C, M)-invariant process. Hence, it follows from Lemma 1 that $L(P)$ is a (C, M)-closed language. Using this fact together with (2) yields

$$L(P) \subseteq [L(Q)]^{CM} \subseteq [L(P)]^{CM} = L(P).$$

Thus, P is a (C, M)-invariant process with generated language $[L(Q)]^{CM}$.

Now let K be a nonempty language. We define an NSM Q satisfying (1) with $L(Q) = \overline{K}$ as follows: Let $\{A_i\}_{i=0}^{n}$ denote the C-equivalence classes of the elements of $\overline{\Sigma}$, with $\epsilon \in A_0$. Let $[s]$ denote the Nerode equivalence class of $s \in \overline{K}$ induced by $\overline{K}$. Let $\Sigma_{\overline{K}}(s) = \{\sigma \in \Sigma |\ s\sigma \in \overline{K}\}$. For the state space of Q, define $X_Q = \{([s], 0) |\ s \in \overline{K}\} \cup \{([s], i) |\ A_i \cap \Sigma_{\overline{K}}(s) \neq \emptyset,\ i > 0\}$.[4] For the set of initial states of Q, define $X_Q^0 = \{([\epsilon], i) \in X_Q |\ i \geq 0\}$. [5] The transitions of Q are as follows:

$$\{(([s], i), \sigma, ([s\sigma], j)) |\ ([s], i), ([s\sigma], j) \in X_Q,\ \sigma \in A_i\}$$

Clearly, $L(Q) = \overline{K}$.

[4]For a given $s \in \overline{K}$, if there is some $i > 0$ such that the state $([s], i)$ is defined, then the state $([s], 0)$ can be omitted.

[5]If we require that there be a unique initial state x_Q^0, then simply augment X_Q by this state and define ϵ-transitions from x_Q^0 to each state in X_Q^0.

Letting P be the modified augmentation of Q gives a (C, M)-invariant process with generated language K^{CM}. If K is regular, then Q is finite-state. Thus, P is finite-state, so K^{CM} is regular.

■

Example 3 Let $K = \overline{\{dd, ca, b\}}$, and let C, M be as in Example 2. The strings in $\overline{K}$ form four Nerode equivalence classes, namely $[\epsilon]$, $[d]$, $[c]$, $[dd] = [ca] = [b]$. When the algorithm described in the proof of Lemma 2 is applied, we obtain the (C, M)-invariant process P shown in Figure 2. $L(P) = \overline{c^*(\epsilon + a)b^* + (b + d)(\epsilon + d)b^*}$, and it is easy to verify directly that this

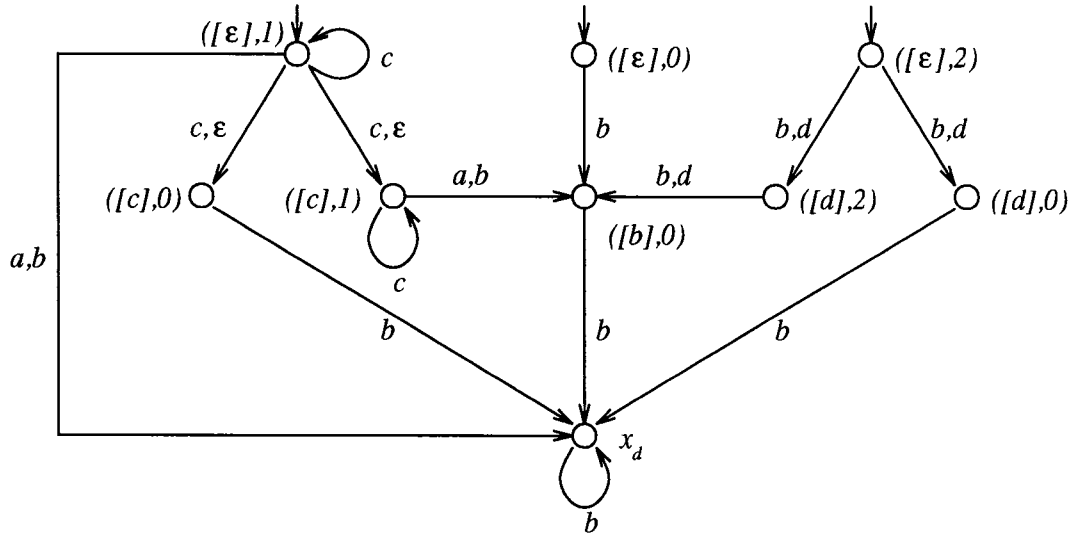

Figure 2: Diagram illustrating Example 3

is the smallest prefix-closed superlanguage of K that satisfies the properties required of a (C, M)-closed language.

Remark 5 From Lemma 2, it follows that if $K = \overline{K}$ is regular, then so is K^{CM}. In fact, if the number of states in the minimal deterministic generator for K is n, then the number of states in the generator for K^{CM} as constructed in the proof of Lemma 2 is no more than the number of control equivalence classes multiplied by n. Since the number of control equivalence classes is bounded above by the size of the event set, it follows that the NSM generator for K^{CM} has $O(n)$ states. In order to test whether a given language K is (C, M)-closed, it suffices to check whether $K^{CM} \subseteq \overline{K}$, which is equivalent to checking whether $K^{CM} \cap \overline{K}^c = \emptyset$ (where the superscript 'c' denotes complement). Since $\overline{K}$ has a deterministic generator with n states, the acceptor for $\overline{K}^c$ has $n + 1$, i.e., $O(n)$ states. So the emptiness condition can be verified in $O(n^2)$ time.

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 $L(P) = K$ if and only if K is a (C, M)-closed language. If in addition 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 7 A process object $((P, \hat{P}), C, M)$ is called a *deterministic process object* if P^{CM} is a deterministic state machine.

Definition 8 A language K is called a *deterministic (C, M)-closed language* if for each $s \in \Sigma^*, \sigma \in \overline{\Sigma}$ such that $s\sigma \in \overline{K}$, it satisfies conditions CM1, CM2, CM3, and the following additional condition:

CMD4 If $\sigma, \sigma' \in \overline{\Sigma}$ with $\sigma' \in M(\sigma)$ and $s\sigma t, s\sigma' \in \overline{K}$, then $s\sigma' t \in \overline{K}$; i.e., $s\sigma$ and $s\sigma'$ are Nerode equivalent relative to $\overline{K}$.

The following results relate the concept of deterministic (C, M)-closed language to controllability and observability. Recall from [26] that given a prefix-closed language H, K is said to be (H, C)-controllable if

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

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

The equivalence relation M on $\overline{\Sigma}$ induces an equivalence relation M^* on Σ^* defined inductively as follows:

- $M^*(\sigma) = M(\epsilon)^* M(\sigma) M(\epsilon)^*$, $\forall \sigma \in \overline{\Sigma}$.
- $M^*(s\sigma) = M^*(s) M^*(\sigma)$.

A second equivalence relation on Σ^*, denoted by $\tilde{M}$, is derived from M^* as follows[6]:

- $\tilde{M}(\sigma) = \{\sigma\}$, $\forall \sigma \in \overline{\Sigma}$.
- $\tilde{M}(s\sigma) = M^*(s)\sigma$, $\forall \sigma \in \Sigma$.

In the sequel, we will suppress the superscript on M^* and denote the M^*-equivalence class of a string s simply by $M(s)$. K is said to be (H, M)-observable [24] if

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

[6] In the case where the mask M is projection-type, $\tilde{M}$ was introduced by Rudie-Wonham [28].

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

Proof: Suppose K is (Σ^*, M)-observable. We show that CMD4 holds by induction on $|t|$. If $|t| = 0$, then CMD4 holds trivially. For the induction step, suppose $t = \overline{t}\overline{\sigma}$. Then from induction hypothesis $s\sigma'\overline{t} \in \overline{K}$. Since $s\sigma t = s\sigma\overline{t}\overline{\sigma} \in \overline{K}$ and $M(s\sigma\overline{t}) = M(s\sigma'\overline{t})$, (Σ^*, M) observability of K implies $s\sigma'\overline{t}\overline{\sigma} = s\sigma' t \in \overline{K}$, completing the induction step.

Conversely, suppose CMD4 holds. In order to prove (Σ^*, M)-observability of K, it suffices to show that for $s, s' \in \overline{K}$ with $M(s) = M(s')$, $[s]_{\overline{K}} = [s']_{\overline{K}}$. We prove this by induction on $|s| + |s'|$. If $|s| + |s'| = 0$, then $s = s' = \epsilon$ and the assertion trivially holds. For the induction step, suppose $s = \overline{s}\,\overline{\sigma}$ with $M(s) = M(s')$. Then we have two cases: (i) $M(\overline{\sigma}) = M(\epsilon)$. Then $M(\overline{s}) = M(s')$. By induction hypothesis $[\overline{s}]_{\overline{K}} = [s']_{\overline{K}}$. Also, since $\overline{\sigma} \in M(\epsilon)$ and $\overline{s}, s = \overline{s}\,\overline{\sigma} \in \overline{K}$, it follows from CMD4 that $[s]_{\overline{K}} = [\overline{s}]_{\overline{K}}$. So $[s]_{\overline{K}} = [s']_{\overline{K}}$. (ii) $M(\overline{\sigma}) \neq M(\epsilon)$. Then without loss of generality, we may assume $s' = \overline{s}'\,\overline{\sigma}'$ with $M(\overline{s}) = M(\overline{s}')$ and $M(\overline{\sigma}) = M(\overline{\sigma}')$. [7] By induction hypothesis, $[\overline{s}]_{\overline{K}} = [\overline{s}']_{\overline{K}}$. Then

$$[s]_{\overline{K}} = [\overline{s}\,\overline{\sigma}]_{\overline{K}} = [\overline{s}'\,\overline{\sigma}]_{\overline{K}} = [\overline{s}'\,\overline{\sigma}']_{\overline{K}} = [s']_{\overline{K}},$$

where the second equality follows from the induction hypothesis and the third equality follows from 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 (Σ^*, C)-controllable and (Σ^*, M)-observable.

Proof: As mentioned above, condition CM1 is the same as (Σ^*, C)-controllability, and by Lemma 3, CMD4 is equivalent to (Σ^*, M)-observability. So it suffices to show that CM1 and CMD4 together imply CM2 and CM3.

In order to establish CM2 and CM3, consider $\sigma' \in [C(\sigma) \cup C(\epsilon)]$ and $s\sigma t \in \overline{K}$. Since $s\sigma \in \overline{K}$ and $\sigma' \in C(\sigma) \cup C(\epsilon)$, it follows from CM1 that $s\sigma' \in \overline{K}$. Suppose $\sigma' \in M(\sigma)$. Then since $s\sigma t \in \overline{K}$, it follows from CMD4 that $s\sigma' t \in \overline{K}$, establishing CM3. On the other hand, if $\sigma' \in M(\epsilon)$, then $M(s) = M(s\sigma')$. So from CMD4, $[s]_{\overline{K}} = [s\sigma']_{\overline{K}}$, which implies $[s]_{\overline{K}} = [s(\sigma')^j]_{\overline{K}}$ for any $j \geq 0$. Since $s\sigma t \in \overline{K}$, this implies that $s(\sigma')^j \sigma t \in \overline{K}$ for any $j \geq 0$, proving CM2. ∎

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

Proof: Let $K := L(P)$. By Lemma 1, K is a (C, M)-closed language. Thus, it suffices to show that CMD4 holds. Pick $\sigma' \in M(\sigma)$ and s such that $s\sigma, s\sigma' \in \overline{K}$. Then both σ and σ' are defined in the (unique) state of P reached by execution of s. Since $M(\sigma) = M(\sigma')$ and

[7] If the final event in s' is in $M(\epsilon)$, then by reversing the roles of s, s' it reduces to case (i).

P is a (C, M)-invariant process, the transitions on σ and σ' lead to the same successor state. This implies that $[s\sigma]_{\overline{K}} = [s\sigma']_{\overline{K}}$. ■

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.

Proof: Let $\hat{K} := K^{CMD}$, and let P be the deterministic state machine obtained by the Nerode construction for $\hat{K}$. By CM1 it follows that if $C(\sigma) = C(\sigma')$ and σ is defined in state $[s]_{\hat{K}}$, then σ' is defined in state $[s]_{\hat{K}}$. Also it follows that each event in $C(\epsilon)$ is defined in state $[s]_{\hat{K}}$. By CMD4 it follows that if $M(\sigma) = M(\epsilon)$ and σ is defined in $[s]_{\hat{K}}$ then the transition is a self-loop. It also follows that if $M(\sigma) = M(\sigma')$, and σ, σ' are both defined in $[s]_{\overline{K}}$, then the corresponding transitions have the same successor states. Thus, P is a deterministic (C, M)-invariant process with $L(P) = K^{CMD}$.

To complete the proof, we show that if K is regular, then K^{CMD} is regular, so P is finite-state. First we claim that

$$K^{CMD} = (K^{CID})^{IMD}. \tag{4}$$

In order to establish (4), it suffices to show that if N is prefix-closed and (Σ^*, C)-controllable, then N^{IMD} is (Σ^*, C)-controllable. Applying this assertion with $N := K^{CID}$ implies that $(K^{CID})^{IMD}$ is a prefix-closed superlanguage of K that is both (Σ^*, C)-controllable and (Σ^*, M)-observable. Thus,

$$K^{CMD} \subseteq (K^{CID})^{IMD}.$$

On the other hand, K^{CMD} is a prefix-closed (Σ^*, M)-observable superlanguage of K^{CID}, so the reverse inclusion also holds.

It follows from [17, Theorem 3] that if $N \subseteq \Sigma^*$,

$$N^{IMD} = \underline{L}; \quad L := \{t \in \Sigma^* | \ \tilde{M}(t) \cap \overline{N} \neq \emptyset\} \tag{5}$$

where $\underline{L}$ denotes the *supremal prefix-closed sublanguage* of a language L. It follows easily from (5) that if N is prefix-closed and (Σ^*, C)-controllable, then the same is true of N^{IMD}, thereby establishing (4).

If K is regular, it is obvious (and well-known) that K^{CID} is regular. However, it follows easily from (5) that if N is regular, then so is N^{IMD}. (See [17, Remark 1].) From (4) we conclude that K^{CMD} is regular. [8] ∎

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 6 Whenever the observation mask *refines* the control mask, i.e., $M(\sigma) \subseteq C(\sigma)$ for each $\sigma \in \overline{\Sigma}$, 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.

Example 4 Let $\Sigma = \{a, b, c, d\}$. Suppose the C-equivalence classes are $\{\epsilon\}$, $\{a, b\}$, $\{c\}$, $\{d\}$ and the M-equivalence classes are $\{\epsilon, a\}$, $\{b, c\}$, $\{d\}$. Let $K = acd + b$. Clearly any prefix-closed (C, M)-closed superlanguage of K must contain $pr(a^*cd + a^*b)$. On the other hand, the (C, M)-invariant process P depicted in Figure 3 generates this language. Thus, $K^{CM} = pr(a^*cd + a^*b)$. Since $pr(a^*cd + a^*b)$ is not (Σ^*, M)-observable, K^{CM} is a proper

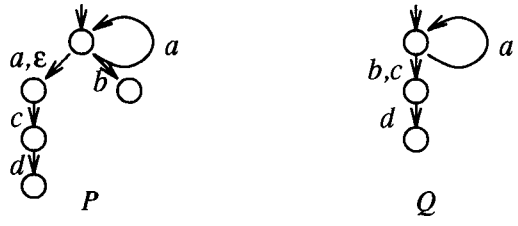

Figure 3: Diagram illustrating Example 4

subset of K^{CMD}. It is easily verified that $K^{CMD} = pr\{a^*(b + c)d\}$. A deterministic (C, M)-invariant process Q that generates K^{CMD} is depicted in Figure 3.

[8] In the case where M is projection-type, the fact that the regularity of N implies the regularity of N^{IMD} was shown in [28].

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 sychronous product of logic modules.

Definition 9 Let $(P_i, \hat{P}_i)$ $(i = 1, 2)$ be logic modules over Σ with $P_i = (\Sigma, X_{P_i}, \delta_{P_i}, X^0_{P_i})$, and let $\delta_{\hat{P}_i}$ denote the transition function for the sub-NSM $\hat{P}_i$. The *synchronous product* of the logic modules is denoted by

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

and is defined as follows:

- $P := P_1 \| P_2$, the standard strict synchronous composition (SSC) [9] of P_1, P_2.
- The transition function of $\hat{P}$ is defined by

$$\begin{aligned} \forall \sigma \in \Sigma : \delta_{\hat{P}}((x_1, x_2), \sigma) &= [\delta_{\hat{P}_1}(x_1, \sigma) \times \delta_{P_2}(x_2, \sigma)] \cup [\delta_{P_1}(x_1, \sigma) \times \delta_{\hat{P}_2}(x_2, \sigma)] \\ \delta_{\hat{P}}((x_1, x_2), \epsilon) &= [\delta_{P_1}(x_1, \epsilon) \times \{x_2\}] \cup [\{x_1\} \times \delta_{P_2}(x_2, \epsilon)] \cup \\ &\quad [\delta_{P_1}(x_1, \epsilon) \times \delta_{P_2}(x_2, \epsilon)] \end{aligned}$$

Proposition 2 The synchronous product of logic modules is associative:

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

Proof: Straightforward. ∎

Definition 10 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

$$((P_0, \hat{P}_0)^{C_0 M_0} \| (P_1, \hat{P}_1)^{C_1 M_1},\ C_0 \wedge C_1,\ M_0 \wedge M_1),$$

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.

[9] 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 σ occurs, it is synchronously executed by each process. An ϵ-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 ϵ 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 ϵ 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.

Thus the NSM $P := P_0^{C_0 M_0} \| P_1^{C_1 M_1}$ of the composition is obtained by SSC of the augmentations of the constituent processes. It contains both the real and virtual transitions and represents the real transitions that would exist if the two constituent processes were interconnected to a "universal event-generator"–i.e., a process object whose logic component consists of an NSM with a single state and real self-loops on every event in Σ. 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. Finally, the control and observation equivalence relations of the composition are obtained by conjunction of the corresponding equivalence relations of the constituent process objects.

Remark 7 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 5 Let $\Sigma = \{a, b\}$, and let $P_0 = \hat{P}_0$, $P_1 = \hat{P}_1$ be as shown in Figure 4. Suppose that

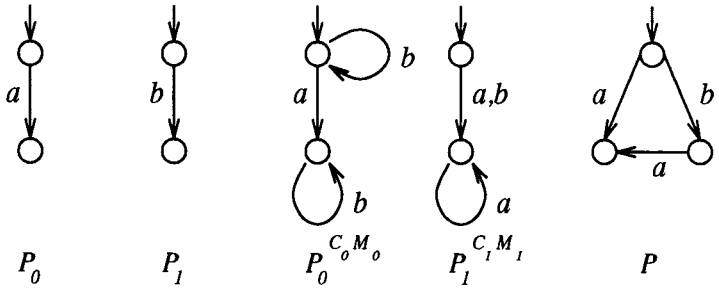

Figure 4: Diagram illustrating Example 5

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 $P_0^{C_0 M_0}$, $P_1^{C_1 M_1}$ are shown in Figure 4. From Definition 10, 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 4, 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.

The next example illustrates a subtlety concerning the role of completely unobservable events in masked composition. It also demonstrates important distinctions between the roles of unmodeled transitions (i.e., ϵ-transitions) and transitions labeled by completely unobservable events.

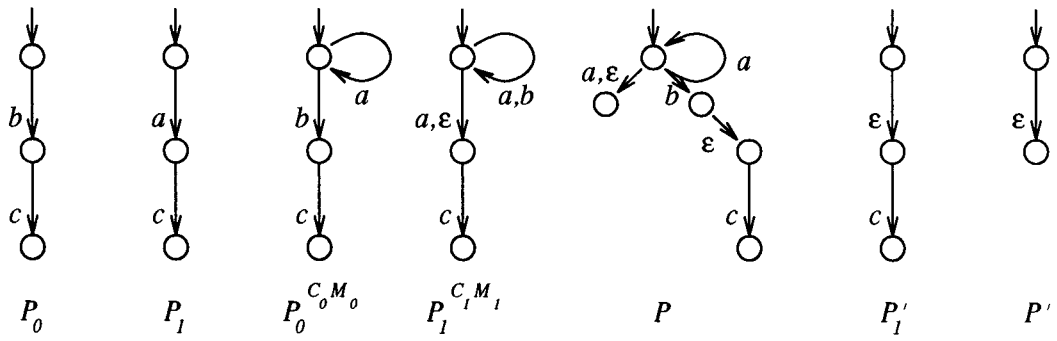

Figure 5: Diagram illustrating Example 6

Example 6 Let $\Sigma = \{a, b, c\}$, and let $P_0 = \hat{P}_0$, $P_1 = \hat{P}_1$ be as shown in Figure 5. Suppose that the C_0 equivalence classes are $\{\epsilon\}$, $\{a, b\}$, $\{c\}$; $M_0 = I$; $C_1 = C_0$; the M_1 equivalence classes are $\{\epsilon, a\}$, $\{b\}$, $\{c\}$. The augmented processes $P_0^{C_0M_0}$, $P_1^{C_1M_1}$ are shown in Figure 5. From Definition 10, 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 5, and that every transition in P is real except for the self-loop on a in the initial state.

Let us interpret the real transitions of P (i.e., the transitions of $\hat{P}$) in its initial state in terms of the 3-step synchronization protocol described in §2. Since a, b are control-equivalent in each process, it follows that the enabled event set in the initial state of P is $\{a, b\}$. If P_0 generates b, P_1 has no observation-equivalent event enabled, so the transition occurs solely in P_0. If P_1 generates a, P_0 has no observation-equivalent event enabled, so the transition occurs solely in P_1. The third possibility is that P_1 executes the completely unobservable event a as a *response transition* to the null event ϵ generated in its environment. This is possible according to the synchronization protocol because a is observation-equivalent to ϵ for P_1. This explains the ϵ-transition in the initial state of P. This transition in P is labeled by ϵ (not by a) since a real transition in a masked composition is always labeled by the event that generates it.

We see that the presence of a completely unobservable event in a constituent process can lead to ϵ-transitions when interconnected with another process via masked composition. Nevertheless, *there are important differences between the functions of ϵ-transitions and those of transitions labeled by completely unobservable events.* To illustrate this, we modify P_1 to obtain a new process $P_1' = \hat{P}_1'$ by replacing the transition on a by an ϵ-transition. If P_1 is replaced by P_1' in the masked composition, we obtain the process P' depicted in Figure 5. P' can only execute an ϵ-transition and then deadlock, so it is drastically different from P.

Focusing on the initial states of P and P', let us examine why this is the case. In the original masked composition, the transition on a in P_1 plays 3 distinct roles corresponding to the 3 steps in the synchronization protocol. (1) It *enables* the event b which can then be generated by P_0. (2) It can *generate* a transition on a. (3) It can be executed as a *response* to the generated event ϵ. When a is replaced by ϵ to obtain P_1', the first two roles are lost. This is why there is only a transition on ϵ in the initial state of P' while there are transitions

on b, a, ϵ in the initial state of P. Thus, *unobservable transitions resemble unmodeled (silent) transitions only in their function as response transitions, but not in their functions in event enablement and event generation.*

In the following remark we note some of the generalities of the masked composition operation.

Remark 8 We have defined masked composition using augmentation. It is useful to note that augmentation can be described using masked composition as follows:

$$\begin{aligned}((P,\hat{P}),C,M) \parallel ((det(\Sigma^*),det(\Sigma^*)),I,I) &= ((P^{CM}[\![det(\Sigma^*),P^{CM}[\![det(\Sigma^*)),I,I)\\ &= ((P^{CM},P^{CM}),I,I),\end{aligned}$$

where $det(\Sigma^*)$ is any deterministic process with language Σ^*.

Next we show that masked composition with a certain kind of process can be used to obtain *hiding* of a given set of events. Given an NSM P and a set of events $B \subset \Sigma$, we define the *restriction of P to B* (or "P hide $\Sigma - B$"), denoted $P|_B$, to be the NSM obtained from P by replacing each label in $\Sigma - B$ by ϵ. Then one can observe that

$$\begin{aligned}((P,P),I,\pi_B) \parallel ((det(B^*),det(B^*)),I,I) &= ((P^{I\pi_B}[\![det(B^*),P^{I\pi_B}[\![det(B^*)),I,I)\\ &= ((P|_B,P|_B),I,I),\end{aligned}$$

where $det(B^*)$ is any deterministic process with language B^*. [10] Thus, restricting P to B is equivalent to taking the masked composition with the deterministic process with language B^* and the events in $\Sigma - B$ regarded as completely unobservable to P.

Furthermore, the *prioritized synchronous composition* (PSC) [10, 31] 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 π_A, π_B respectively.[11] The *full synchronous composition* operator in concurrency theory [12, 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 [12, p. 119] can also be obtained using masked composition as illustrated in Example 7.

Finally, the traditional supervisory control model [26, 24, 3] 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 Σ_c, Σ_o denote the set of controllable events and set of observable events respectively.

[10] Actually, determinism is not essential here.

[11] Actually, there is a subtle difference between PSC and this special case of MC. In PSC, it is not possible for one process to execute an event σ by itself when the other process is in a state where σ is possible after first executing an ϵ-transition. In MC, this is entirely possible.

Example 7 In this example, we demonstrate how interleaving composition [12, 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 $C_i = M_i = \pi_{\Sigma_i}$. 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. (See also Example 11.)

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

Theorem 4 Masked composition is associative:

$$\begin{aligned}[[((P_0,\hat{P}_0),C_0,M_0) \parallel ((P_1,\hat{P}_1),C_1,M_1)] \quad &\parallel \quad ((P_2,\hat{P}_2),C_2,M_2) = \\ ((P_0,\hat{P}_0),C_0,M_0) \quad &\parallel \quad [((P_1,\hat{P}_1),C_1,M_1) \parallel ((P_2,\hat{P}_2),C_2,M_2)].\end{aligned}$$

Proof:

$$\begin{aligned}
&[((P_0,\hat{P}_0),C_0,M_0) \parallel ((P_1,\hat{P}_1),C_1,M_1)] \parallel ((P_2,\hat{P}_2),C_2,M_2) \\
&= ((P_0,\hat{P}_0)^{C_0M_0}[\!](P_1,\hat{P}_1)^{C_1M_1}, C_0 \wedge C_1, M_0 \wedge M_1) \parallel ((P_2,\hat{P}_2),C_2,M_2) \\
&= ((P_0,\hat{P}_0)^{C_0M_0}[\!](P_1,\hat{P}_1)^{C_1M_1})^{C_0\wedge C_1\ M_0\wedge M_1}[\!](P_2,\hat{P}_2)^{C_2M_2}, (C_0 \wedge C_1) \wedge C_2, (M_0 \wedge M_1) \wedge M_2) \\
&= (((P_0,\hat{P}_0)^{C_0M_0}[\!](P_1,\hat{P}_1)^{C_1M_1})[\!](P_2,\hat{P}_2)^{C_2M_2}, (C_0 \wedge C_1) \wedge C_2, (M_0 \wedge M_1) \wedge M_2) \qquad (6)
\end{aligned}$$

where the final equality is a consequence of the fact that $C_0 \wedge C_1$ refines both C_0, C_1 and $M_0 \wedge M_1$ refines both M_0, M_1. Since both the synchronous product of logic modules and the conjunction of equivalence relations are associative, the associativity of masked composition follows immediately. ■

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

- $[C_i(\sigma) = \{\sigma\}] \wedge [M_i(\sigma) = \{\sigma\}]$.
- Every transition on σ in P_i is also a transition in $\hat{P}_i$.

Then $\hat{P} = P = P_0^{C_0M_0}[\!]\cdots[\!]P_n^{C_nM_n}$.

Proof: From Definitions 9 and 10, it follows that the every ϵ-transition in P is a transition in $\hat{P}$. Let $\sigma \in \Sigma$. Without loss of generality, suppose that $C_0(\sigma) = \{\sigma\}$ and $M_0(\sigma) = \{\sigma\}$, and that every transition on σ in P_0 is a transition in $\hat{P}_0$. Then the transitions labeled by σ in $P^{C_0 M_0}$ are precisely those labeled by σ in $\hat{P}_0$. It follows from Definitions 9 and 10 that

$$\begin{aligned}
\delta_{\hat{P}}((x_0, \ldots, x_n), \sigma) &\supseteq \delta_{\hat{P}_0}(x_0, \sigma) \times \delta_{P_1^{C_1 M_1}}(x_1, \sigma) \times \ldots \times \delta_{P_n^{C_n M_n}}(x_n, \sigma) \\
&= \delta_{P_0^{C_0 M_0}}(x_0, \sigma) \times \delta_{P_1^{C_1 M_1}}(x_1, \sigma) \times \ldots \times \delta_{P_n^{C_n M_n}}(x_n, \sigma) \\
&= \delta_P(x_0, \ldots, x_n), \sigma)
\end{aligned}$$

■

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 [31, Proposition 4].

Corollary 1 Under the hypotheses of Theorem 5 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 9 The result of Theorem 5 can be specialized to the situation in the Ramadge-Wonham theory [27]. 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 $\hat{P} = P = P_0 \| P_1^{C_1 M_1}$. 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 so that the language intersection result of Corollary 1 holds. Thus under this assumption, the synthesis problem is to construct P_1 such that

$$L(P_0^{C_0 M_0}) \cap L(P_1^{C_1 M_1}) = K. \tag{7}$$

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

Definition 11 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:

$$L(P_0^{C_0M_0}) \cap K^{C_1M_1} = \overline{K}. \tag{8}$$

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), and a language $K \subseteq L(P_0^{C_0M_0})$. Suppose

$$\forall \sigma \in \Sigma : [C_0(\sigma) = \{\sigma\} \wedge M_0(\sigma) = \{\sigma\}] \vee [C_1(\sigma) = \{\sigma\} \wedge M_1(\sigma) = \{\sigma\}].$$

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.

Proof: Suppose there exists P_1 such that $L(P_0^{C_0M_0}) \cap L(P_1^{C_1M_1}) = K$. Then clearly, $K = \overline{K} \neq \emptyset$. So it suffices to show that $L(P_0^{C_0M_0}) \cap K^{C_1M_1} = K$, or equivalently, $L(P_0^{C_0M_0}) \cap K^{C_1M_1} \subseteq K$. From hypothesis, it follows that $L(P_1^{C_1M_1})$ is a superlanguage of K. Also, since it is prefix-closed, and since $P_1^{C_1M_1}$ is a (C_1, M_1)-invariant process, it follows from Theorem 1 that $L(P_1^{C_1M_1}) \in \overline{C_1M_1}(K)$, i.e., $K^{C_1M_1} \subseteq L(P_1^{C_1M_1})$. So we obtain $L(P_0^{C_0M_0}) \cap K^{C_1M_1} \subseteq L(P_0^{C_0M_0}) \cap L(P_1^{C_1M_1}) = K$, as desired.

Conversely, suppose that (8) holds and $K = \overline{K} \neq \emptyset$. Since K is nonempty, by Theorem 1, there exists P_1 such that $L(P_1^{C_1M_1}) = L(P_1) = K^{C_1M_1}$. Then it follows from the fact that K is relatively (C_1, M_1)-closed that the supervisor $((P_1, P_1), C_1, M_1)$ gives $\overline{K} = K$ as the generated language of the controlled system. Finally, if K is regular, then by Theorem 1, P_1 can be chosen to be finite-state. ∎

Remark 10 It follows from Theorem 6 that the existence of a supervisor for achieving a desired generated language $K \subseteq L(P_0^{C_0M_0})$ requires that the relative (C_1, M_1)-closure condition be verified. Since $K = \overline{K}$ is a sublanguage of $L(P_0^{C_0M_0})$, the condition of relative (C_1, M_1)-closure is equivalent to the forward containment $L(P_0^{C_0M_0}) \cap K^{C_1M_1} \subseteq K$, which is equivalent to the emptiness of the language $L(P_0^{C_0M_0}) \cap K^{C_1M_1} \cap K^c$, where $K^c := \Sigma^* - K$ is the complement of K. Let m be the number of states in the plant, and n be the number of states in the minimal deterministic generator for K. Then by definition of augmentation, the number of states in the augmented plant is still m. Also, by the construction given in the proof of Lemma 2, the number of states in the nondeterministic generator for $K^{C_1M_1}$ is $O(n)$. Finally, the number of states in the generator for K^c is $n + 1$. So by considering

the SSC of the augmented plant NSM, the nondeterministic generator of $K^{C_1 M_1}$, and the deterministic recognizer of K^c, and testing the emptiness of its recognized language, relative (C_1, M_1)-closure can be tested in $O(mn^2)$ time. Furthermore, whenever K is relatively (C_1, M_1)-closed, the required supervisor can be chosen to be the nondeterministic generator of $K^{C_1 M_1}$, which has $O(n)$ states. This is in contrast to the traditional supervisory control where the synthesis of the (deterministic) supervisor when it exists is known to be an NP-complete problem [34].

In the next remark we discuss some special cases of the result obtained in Theorem 6.

Remark 11 First, in order to investigate controllability alone, consider the case of complete observation, i.e., the case when the observation mask of the supervisor is the identity mask $(M_1 = I)$. It follows from Definition 6 that every string in $K^{C_1 M_1}$ is of the form $s_1 \sigma' s_2$, where there exists $\sigma \in C_1(\sigma')$ such that $s_1 \sigma \in \overline{K}$ and $s_2 \in C(\epsilon)^*$. It follows easily from this that the realizability condition (8) is equivalent to the condition

- $\forall \sigma \in \overline{\Sigma}$, $\sigma' \in \Sigma$: $s\sigma \in \overline{K}$, $s\sigma' \in L(P_0^{C_0 M_0})$, $C(\sigma) = C(\sigma') \Longrightarrow s\sigma' \in \overline{K}$.

This represents the generalization of standard controllability [27] to the case where the control mask of the supervisor is not a natural projection and the plant itself has control and observation masks.

Next, in order to investigate observability alone, consider the case when the control mask of the supervisor is the identity mask, i.e., $C_1 = I$. In this case, the realizability condition (8) reduces to the following:

- Any string $t \in L(P_0^{C_0 M_0})$ that can be obtained from a string $s \in \overline{K}$ by replacing each completely unobservable event σ by an arbitrary power σ^j $(j \geq 0)$ must itself be in $\overline{K}$.

This condition is significantly weaker than standard observability [27] with respect to the language of the augmented plant. It is interesting to note that the presence of pairs of events that are indistinguishable from each other but not completely unobservable has no bearing on realizability under the assumption that $C_1 = I$.

Finally, consider the case of standard supervisory control formulation, i.e., where C_1 and M_1 are each natural projections corresponding to the controllable event set Σ_c and the observable event set Σ_o. Then the conditions in Definition 6 reduce to language closure with respect to the following operations:

- appending a string of uncontrollable events
- replacing an unobservable event by a nonnegative power of that event
- inserting an event that is both uncontrollable and unobservable at an arbitrary place in the string

In this case, the realizability condition (8) reduces to standard controllability of K with respect to the language of the augmented plant together the following:

- Any string $t \in L(P_0^{C_0 M_0})$ that can be obtained from a string $s \in \overline{K}$ by replacing each completely unobservable event σ by an arbitrary power σ^j $(j \geq 0)$ and inserting events that are both uncontrollable and unobservable must itself be in $\overline{K}$.

This condition is weaker than the condition of *weak controllability and observability* used by Inan [15] in the case where $C_0 = M_0 = I$ and P_1 is not allowed to have ϵ-transitions.

Now we consider the situation where the given specification language K does not satisfy the relative (C_1, M_1)-closure condition (8). Given a plant $((P_0, P_0), C_0, M_0)$, masks (C_1, M_1), and a language K, let $\underline{P_0 C_0 M_0 C_1 M_1}(K)$ denote the collection of all relatively (C_1, M_1)-closed sublanguages of K.

Lemma 6 $\underline{P_0 C_0 M_0 C_1 M_1}(K)$ is nonempty and closed under union; consequently, it contains a unique supremal element.

Proof: $\emptyset \in \underline{P_0 C_0 M_0 C_1 M_1}(K)$, so $\underline{P_0 C_0 M_0 C_1 M_1}(K)$ is nonempty. Suppose that H_1, H_2 are relatively (C_1, M_1)-closed sublanguages of K. Then

$$L(P^{C_0 M_0}) \cap (H_1^{C_1 M_1} \cup H_2^{C_1 M_1}) = \overline{H}_1 \cup \overline{H}_2 = \overline{H_1 \cup H_2}.$$

Hence, it suffices to show that $(H_1 \cup H_2)^{C_1 M_1} = H_1^{C_1 M_1} \cup H_2^{C_1 M_1}$. This follows easily from the closure of (C_1, M_1)-languages under union. ∎

When P_0, C_0, M_0, C_1, M_1 are clear from the context, we use the symbol $K^{\Uparrow}$ to denote the supremal element of $\underline{P_0 C_0 M_0 C_1 M_1}(K)$. Since K is relatively (C_1, M_1)-closed if and only if the same is true of $\overline{K}$, it follows that $K^{\Uparrow}$ is prefix-closed whenever K is prefix-closed. The result of the following corollary follows from Theorem 6:

Corollary 2 Consider a plant $((P_0, P_0), C_0, M_0)$, control and observation masks (C_1, M_1), and a nonempty and prefix-closed language $K \subseteq L(P_0^{C_0 M_0})$. Suppose

$$\forall \sigma \in \Sigma : [C_0(\sigma) = \{\sigma\} \wedge M_0(\sigma) = \{\sigma\}] \vee [C_1(\sigma) = \{\sigma\} \wedge M_1(\sigma) = \{\sigma\}].$$

Then there exists a supervisor $((P_1, P_1), C_1, M_1)$ such that the generated language of the controlled system is contained in K if and only if $K^{\Uparrow} \neq \emptyset$.

In the remainder of this section, we consider supervisory control under the additional restriction that the supervisor process object be deterministic. This means that $P_1^{C_1 M_1}$ must be a deterministic state machine. We continue to assume that each event is both completely controllable and completely observable to either the plant or the supervisor.

Definition 12 Given a plant $((P_0, P_0), C_0, M_0)$, control and observation masks (C_1, M_1), and a language $K \subseteq L(P_0^{C_0 M_0})$, K is said to be a $((P_0, P_0), C_0, M_0)$-*relatively deterministic* (C_1, M_1)-*closed* language if it satisfies the following condition:

$$L(P_0^{C_0 M_0}) \cap K^{C_1 M_1 D} = \overline{K}. \tag{9}$$

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 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

$$\forall \sigma \in \Sigma : [C_0(\sigma) = \{\sigma\} \wedge M_0(\sigma) = \{\sigma\}] \vee [C_1(\sigma) = \{\sigma\} \wedge M_1(\sigma) = \{\sigma\}].$$

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.

Proof: The proof of the equivalence of parts 1 and 2, and the assertion that regularity of K implies P_1 can be chosen to be finite-state is analogous to the proof of Theorem 6. We establish the equivalence of parts 2 and 3 below.

Suppose K is relatively deterministic (C_1, M_1)-closed. We need to show that $\overline{K}$ is $(L(P_0^{C_0 M_0}), C_1)$-controllable and $(L(P_0^{C_0 M_0}), M_1)$-observable. By Theorem 2, $K^{C_1 M_1 D}$ is (Σ^*, C_1)-controllable and (Σ^*, M_1)-observable. It follows that $K^{C_1 M_1 D} \cap L(P_0^{C_0 M_0}) = \overline{K}$ is $(L(P_0^{C_0 M_0}), C_1)$-controllable and $(L(P_0^{C_0 M_0}), M_1)$-observable.

Conversely, suppose K is $(L(P_0^{C_0 M_0}), C_1)$-controllable and $(L(P_0^{C_0 M_0}), M_1)$-observable. We need to show that $L(P_0^{C_0 M_0}) \cap K^{C_1 M_1 D} \subseteq \overline{K}$. Clearly, this holds when $K = \emptyset$. Otherwise, we prove by induction on $|s|$ that if $s \in L(P_0^{C_0 M_0}) \cap K^{C_1 M_1 D}$, then $s \in \overline{K}$. If $|s| = 0$, then the assertion trivially holds. For the induction step, let $s = \overline{s}\sigma \in L(P_0^{C_0 M_0}) \cap K^{C_1 M_1 D}$. By induction hypothesis, $\overline{s} \in \overline{K}$. Since $K^{C_1 M_1 D}$ is the *infimal* prefix-closed (Σ^*, C_1)-controllable and (Σ^*, M_1)-observable superlanguage of K, this implies that one of the following two conditions must hold:

1. There exists $\sigma' \in \overline{\Sigma}$ with $C_1(\sigma) = C_1(\sigma')$ such that $\overline{s}\sigma' \in \overline{K}$.
2. There exists $\overline{t} \in \overline{K}$ with $M_1(\overline{t}) = M_1(\overline{s})$ such that $\overline{t}\sigma \in \overline{K}$.

In the first case, the $(L(P_0^{C_0 M_0}), C_1)$-controllability of K implies that $\overline{s}\sigma \in \overline{K}$. In the second case, the $(L(P_0^{C_0 M_0}), M_1)$-observability of K implies that $\overline{s}\sigma \in \overline{K}$. This completes the induction step. ∎

Remark 12 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. Consequently, for deterministic supervision, there is no analogue of Corollary 2.

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, and it follows from Theorems 6, 7 and Remark 6.

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

$$\forall \sigma \in \Sigma: \ [C_0(\sigma) = \{\sigma\} \wedge M_0(\sigma) = \{\sigma\}] \vee [C_1(\sigma) = \{\sigma\} \wedge M_1(\sigma) = \{\sigma\}].$$

If $M(\sigma) \subseteq C(\sigma)$ for each $\sigma \in \overline{\Sigma}$, 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 13 Theorem 8 generalizes a previous result of Shayman-Kumar [31, 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., $M_0 = I$.

5 Applications and Examples

In this section, we present several examples that illustrate the generality of the process object/masked composition paradigm for modeling and control synthesis.

5.1 Nondeterministic Supervisors for Minimax Control

In most systems of practical interest, both hard and soft constraints are present. Hard constraints must be satisfied or the performance is deemed unacceptable; safety requirements are typically in the form of hard constraints. Traditional supervisory control theory [27] applies only to problems with hard constraints; a legal language is given, and a supervisor must be synthesized so the closed-loop generated language is contained in the legal language.

Lafortune-Lin [23] represent soft and hard constraints in terms of desirable and tolerable behaviors and study the supervisory control problem in that setting. Since satisfaction of soft constraints is desirable but not essential, an alternative way to represent soft constraints would be to impose a penalty that is finite whenever such a constraint is violated. On the other hand, hard constraints can be represented by penalties that are infinite. Optimization is a natural tool for control design in the presence of soft constraints. It is not our intention to explore optimal control of discrete event systems in detail; rather we simply present an

example to illustrate the need for nondeterministic supervisors. For discussion of optimal control theory for discrete event systems, see e.g., [25, 29, 18, 2]

When both hard and soft constraints are present, it is natural to consider a hierarchical synthesis procedure. In the first step, supervisory control theory is used to construct a maximally permissive supervisor enforcing the hard constraints. In the second step, the controller is refined using optimization techniques in order to best take into account the soft constraints. Thus, supervisory control theory is essentially used to compute a feasible set for an optimization problem.

Since a language may be relatively (C_1, M_1)-closed without being relatively deterministic (C_1, M_1)-closed, a restriction to deterministic supervisor objects in the first step can significantly reduce the size of the feasible set for the second step optimization problem. The following example shows that when the optimization problem is of the *minimax type*, this restriction can result in a substantially higher value for the cost function.

Example 8 Consider the plant P_0 depicted in Figure 6, where the double-circles represent final (marked) states.

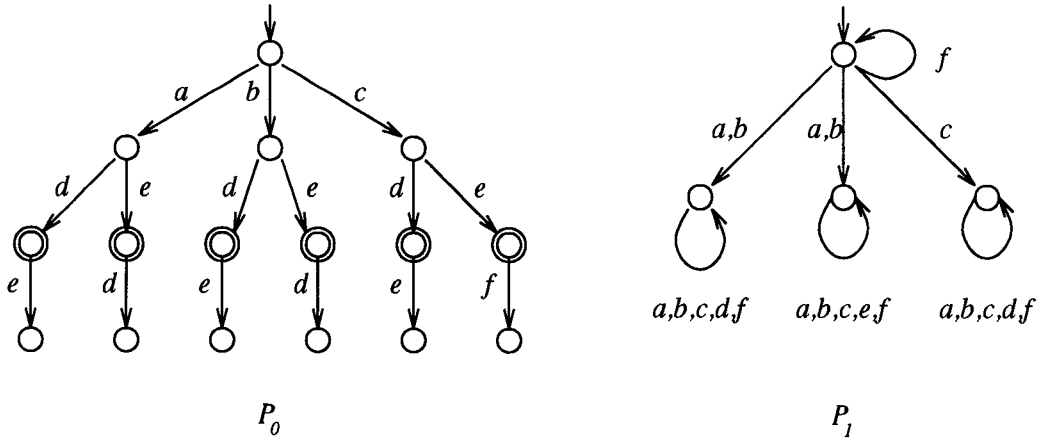

Figure 6: Diagram illustrating Example 8

Suppose that the control and observation masks of the plant are (I, I), and the control and observation masks (C_1, M_1) of the supervisor are such that

- a, b, c, f are completely uncontrollable.
- a and b are observation-equivalent but not completely unobservable.
- d and e are completely unobservable.

Suppose that the hard constraints are represented by the legal language specified by $\overline{K}$, where

$$K = \{ad, ae, bd, be, cd, ce\}.$$

It is also required that the controlled system be nonblocking in the sense that every generated string should be extendable to a string which results in a marked state for the plant–i.e., to a string in K. Suppose that the soft constraints are represented by positive costs of r_1 and r_2 associated with the strings ae and bd, respectively. All other strings in K have cost 0. We assume that if $M_1(a) = M_1(b)$ is observed, there is probability q (respectively, $1-q$) that the plant executed the event a (respectively, b), but that the parameter q is unknown to the control designer. Then a natural design approach would be to construct a nonblocking supervisory controller that ensures that the generated language of the controlled system is contained in $\overline{K}$, and then refine the controller to minimize the maximum cost associated with the soft constraints.

First suppose that the supervisor process object is required to be deterministic. Since the plant has identity masks, it follows from Theorem 7 that a nonempty prefix-closed language $L \subseteq L(P_0)$ can be deterministically realized as the closed-loop generated language if and only if L is $(L(P_0), C_1)$-controllable and $(L(P_0), M_1)$-observable. Two maximal such languages are

$$K_1 := pr\{ad, bd, cd\}, \qquad K_2 := pr\{ae, be, cd\}.$$

If either K_1 or K_2 is imposed, the worst-case cost incurred is either r_2 or r_1 corresponding to the strings bd and ae respectively.

Now suppose that the requirement that the supervisor process object be deterministic is dropped. It is not difficult to show that the supremal relatively (C_1, M_1)-closed sublanguage of $\overline{K}$ is given by

$$\overline{K}^{\Uparrow} = pr\{ad, ae, bd, be, cd\}.$$

A process P_1 for which the generated language of the controlled system is given by

$$L(P_0^{C_0 M_0}) \cap L(P_1^{C_1 M_1}) = L(P_0) \cap L(P_1) = \overline{K}^{\Uparrow}$$

is depicted in Figure 6. In its initial state, this supervisor enables a, b, c, f. If a process in its environment executes either a or b, P_1 makes a nondeterministic choice between two possible successor states. If we postulate that this choice is made based on a single Bernoulli trial with parameter p, then the worst-case expected cost is

$$J(p) = \max_{0 \le q \le 1} (q(1-p)r_1 + (1-q)pr_2).$$

This quantity is minimized by choosing $p = \frac{r_1}{r_1 + r_2}$, resulting in optimal (minimax) expected cost $J(p) = \frac{r_1 r_2^2 + r_1^2 r_2}{(r_1 + r_2)^2}$. Note that if $r_1 = r_2 := r$, the deterministic controllers each give worst-case cost r, while the optimal nondeterministic controller (i.e., optimal randomized strategy) gives an expected cost of $r/2$.

5.2 Open-Loop Control

In a given state x, a supervisor process presents the plant with an enabled event set that consists of every event that is completely uncontrollable (to the supervisor) together with every event that is control-equivalent to an event defined in x. In our framework, the supervisor may be constructed so there is an ϵ-transition from x to a new state x'. Consequently, the set of events enabled by the supervisor can change without the requirement that the supervisor observe an event executed by the plant. This can also occur when the supervisor has a transition labeled by a completely unobservable event. Thus, the control input from the supervisor to the plant can evolve in an open-loop, as well as a closed-loop mode. In contrast, the control input in traditional supervisory control must be constant between observed events, so it is strictly closed-loop. The following example demonstrates that open-loop control may be required to meet a given specification.

Example 9 Consider a system that serves customers of two types A and B. See Figure 7. The events a, b indicate service of customers of types A, B respectively. If a type B

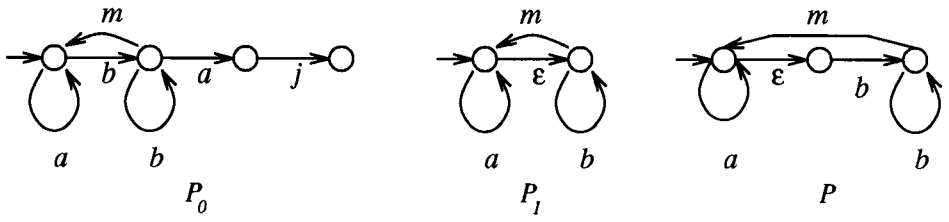

Figure 7: Diagram illustrating Example 9

customer has been served, the server will jam (event j) if a type A customer is served prior to the performance of a maintenance function (event m). We assume that a supervisor cannot observe the servicing of customers but is able to prevent either type of customer from entering service. Thus, a, b are completely unobservable while j is completely uncontrollable to the supervisor. We assume that every event is completely controllable and observable to the plant–i.e., the control and observation masks of the plant are identity masks. The generated language of the plant is

$$L(P_0) = pr(a^*bb^*m)^*a^*bb^*aj. \tag{10}$$

Suppose that the specification is that the plant never jam. Then the specification language is given by

$$K = pr(a^*bb^*m)^*a^*bb^*a.$$

First note that K is not relatively (C_1, M_1)-closed. For example, if we let $s = ba$, $\sigma = \epsilon$, $\sigma' = j$, then $s\sigma \in \overline{K}$, $\sigma' \in C_1(\epsilon)$, $s\sigma' \in L(P_0^{C_0 M_0})$, but $s\sigma' \notin \overline{K}$.

Next, we claim that the supremal relatively (C_1, M_1)-closed sublanguage $K^{\Uparrow}$ of K is given by

$$K_1 := pr(a^*bb^*m)^*.$$

It is easy to see that any relatively (C_1, M_1)-closed sublanguage of $\overline{K}$ must be a sublanguage of K_1, so it suffices to show that K_1 is itself relatively (C_1, M_1)-closed. Let P_1 be as depicted in Figure 7. Then $P_1^{C_1 M_1}$ is obtained from P_1 by adding self-loops labeled by j in each state. The strict synchronous composition $P := P_0^{C_0 M_0} \| P_1^{C_1 M_1}$ is depicted in Figure 7. Since $L(P) = K_1$, it follows from Theorem 6 that K_1 is indeed relatively (C_1, M_1)-closed.

If we choose the supervisor process object to be $((P_1, P_1), C_1, M_1)$, then we obtain K_1 as the generated language of the closed-loop system. Initially, the supervisor only permits service to customers of type A, and then changes its control in an open-loop manner to only permit service to customers of type B. The control is switched back to its initial value upon occurrence of the maintenance event. This controller permits access to the system by both types of customers while preventing jamming by ensuring that maintence is performed before a type A customer can follow a type B customer. Note that the supervisor exercises open-loop (as well as closed-loop) control by changing states on an ϵ-transition–i.e., without observing any event in the plant.

Now let us examine the possibility of using purely closed-loop supervisory control. To meet the requirement that j never occur, the supervisor must disable either a or b initially. If b is disabled, then no observable event can ever occur, so the control input from the supervisor cannot change. Thus, type B customers are permanently barred from the system. Suppose instead that a is disabled initially. There are two possibilities when m is observed: (1) If the control input is changed to permit a and disable b, then type B customers will be barred from that time on. (2) If the control input is not changed, type A customers are barred from the system. Thus, purely closed-loop control gives an unsatisfactory controlled system in which one type of customer is eventually barred.

Finally, we note that a purely closed-loop supervisor could impose the specification and permit service of both types of customers *provided a time-out event was included in the model of the plant.* Then the controller could change access from A to B in response to the time-out event rather than on an ϵ-transition. However, the inclusion of such an event in the plant model represents ad hoc design of a portion of the controller that is incorporated as part of the plant. This is contrary to the philosophy of supervisory control theory that control design should be accomplished by following systematic algorithms rather than by ad hoc methods.

5.3 Reactive Systems

A reactive system is a system that interacts with an environment that includes more than just the controller. Using an example, we will illustrate how the feature of *plant masks*, as opposed to supervisor masks, permits application of our framework to such systems.

Example 10 We consider a simple model for fire control radar on a fighter aircraft. In its initial state, a blip can appear on the radar screen indicating the presence of another aircraft. The events a, b indicate blips corresponding to a friend and foe respectively. Following the occurrence of either blip, the screen may clear, represented by the event c. In addition, the

occurrence of b may be followed by the firing of a missile at the foe, represented by the event f. After the missile is fired, the screen is cleared. The transitions labeled by a and b are regarded as virtual transitions since they cannot be generated by the fighter plane; rather they can only be tracked by the the fighter plane if generated by another process in its environment. This describes the logic module $(P_0, \hat{P}_0)$. Figure 8 depicts P_0 together with the sub-NSM $\hat{P}_0$ obtained by deleting the virtual transitions. In the nominal model of the plant, every event is completely controllable and completely observable–i.e., $C_0 = I$, $M_0 = I$.

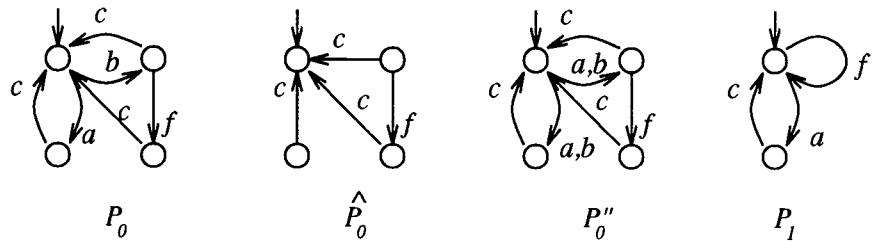

Figure 8: Diagram illustrating Example 10

We assume that the fighter is operating in an environment that can generate both a and b. Since we do not have a detailed model of the environment, we simply represent it by an "event generator" process consisting of a single state with self-loops on a and b representing real transitions. Since the purpose of this process is to generate events rather than to restrict or track events generated by other processes, we use the trivial masks for which every event is completely uncontrollable and completely unobservable.

The process object $((P_0', \hat{P}_0'), C_0', M_0')$ representing the fighter plane operating in this environment is obtained by taking the masked composition of $((P_0, \hat{P}_0), I, I)$ with the event generator. From Definition 10 it follows that $((P_0', \hat{P}_0'), C_0', M_0') = ((P_0, P_0), I, I)$. Thus, the only effect of the presence of the event generator is to make the transitions on a, b real rather than virtual.

Suppose that the specification is that a missile should never be fired following the appearance of a blip representing a friend–i.e., there should be no string containing af. Since this condition is satisfied by every string in $L(\hat{P}_0') = L(P_0)$, the nominal plant satisfies the specification without the need for any additional controller. However, suppose we require a control design that is robust with respect to sensor degradation that makes a, b indistinguishable to the plant. This corresponds to replacing the mask $M_0 = I$ with an observation mask M_0'' that identifies a and b. Let $((P_0'', \hat{P}_0''), I, M_0'')$ represent the fighter plane operating in the environment under the degraded sensor capabilities. In this case, it follows from Definition 10 that $P_0'' = \hat{P}_0'' = P_0^{IM_0''}$ which is shown in Figure 8. Since $af \in L(\hat{P}_0'')$, the perturbed system does not satisfy the specification, so an additional supervisor must be designed.

We assume that the supervisor is implemented via an air traffic controller that can track the positions of friendly aircraft and can communicate a command to the fighter aircraft which effectively disables the event f. We model these capabilities via supervisor masks C_1, M_1 relative to which a, c are completely uncontrollable and completely observable, b is

completely uncontrollable and completely unobservable, and f is completely controllable and observable. A possible choice for the supervisor logic module is given by the process $P_1 = \hat{P}_1$ depicted in Figure 8. Let $((P, \hat{P}), C, M)$ denote the process object representing the controlled system when $((P_0'', \hat{P}_0''), I, M_0'')$ is connected to $((P_1, \hat{P}_1), C_1, M_1)$ by masked composition. It is straightforward to verify that the resulting logic module satisfies $P = \hat{P} = P_0$. Thus, the controlled behavior of the perturbed plant is identical to the uncontrolled behavior of the nominal plant. Consequently, under the control of the given supervisor, the controlled system meets the specifications regardless of whether sensor degradation has occurred.

5.4 Object-Oriented Design

Object-oriented design is desirable because it offers the possibility of developing reusable software modules [4]. Efforts to develop such an approach for continuous variable control systems have already begun [16]. Object-oriented methodologies for modeling discrete event systems are described in [5, 6].

In [8, 7], an object-oriented approach to supervisory control design for discrete event *resource-user* systems is proposed. The software reusability emanates from the fact that the resources (e.g., machining devices) are described by their general behavior on an abstract level. Application-specific details are suppressed in the resource model so that its software is independent of the number and type of users (e.g., parts). Specification processes describing the requirements of each user (e.g., the sequence of machines a part must visit) are composed by *pure interleaving* to obtain an overall specification for the requirements of the set of users sharing the resources. This specification is then composed with the process modeling the resources by *full synchronization*–i.e., synchronization of those events in the intersection of the alphabets. The supervisor obtained from this procedure may fail to be *complete*–i.e., it may disable uncontrollable events. If this is the case, customization of the supervisor is required which depends on the resource model and set of uncontrollable events at hand.

In [8, 7], a machining operation is labeled without reference to specific application not only in the resource model, but also in the specifications. For example, consider a factory that concurrently processes two types of parts. A drill operation would be represented by a generic drill event d in the resource model, and by the same label d in the specification models for each part. Consequently, in the modeling approach proposed in [8, 7], there is no way to represent specifications that are intended to restrict the concurrency in the processing of the two parts. For example, it is impossible to express a requirement that part 1 be drilled before part 2 since both drilling operations are represented by the same label d used in the resource model. Concurrency specifications are common in manufacturing and in other applications such as database management.

The process object/masked composition paradigm yields a different approach to object-oriented modeling and control design for discrete event systems. For concreteness, we describe it in the context of the manufacturing example above. The resource model would be represented by a process object $((P_0, \hat{P}_0), C_0, M_0)$. If there are a maximum of n parts that

can be processed concurrently by the factory, the specification for the sequence of machines part i must visit would be given by a process object $((P_i, \hat{P}_i), C_i, M_i)$ $(i = 1, \ldots, n)$. If there is to be no interaction between the distinct parts as they traverse the factory, then the event labels in P_i and P_j are distinct. Thus, a drill operation on part i would be labeled by an event d_i in P_i, while a drill operation on part j would be labeled by an event d_j in P_j. We choose the control and observation masks so that d_i is completely uncontrollable and completely unobservable to P_j and vice-versa. Consequently, in the masked composition of the two process objects, the drill operations d_i, d_j purely interleave. On the other hand, we choose the control and observation masks C_0, M_0 of the resource process to be such that $C_0(d_i) = C_0(d_j) := d$ and $M_0(d_i) = M_0(d_j) := d$. This has the effect of making the drill operation part-independent in the resource model, thereby permitting software reusability, while retaining the individual identities of d_i and d_j in the part specifications, thereby accomodating concurrency control specifications. This is illustrated by the following example.

Example 11 Suppose the resource process models the continuous independent operation of two machines A and B. It is represented by the process object $((P_0, \hat{P}_0), C_0, M_0)$ where $P_0 = \hat{P}_0$ consists of a single state with self-loops labeled by a, b, and where $C_0(a_i) := a$, $C_0(b_i) := b$, $M_0(a_i) := a$, $M_0(b_i) := b$, $\forall i$. Suppose there are two parts to be machined. Part 1 requires two successive operations on machine A while part 2 requires an operation on A followed by an operation on B. The specification for part i is represented by a process object $((P_i, \hat{P}_i), C_i, M_i)$ where $P_i = \hat{P}_i$ and $a_j, b_j \in C_i(\epsilon) \cap M_i(\epsilon)$ if $i \neq j$ $(i, j = 1, 2)$. The events a_j, b_j represent the operations of part j on A and B respectively. The NSM's P_0, P_1, P_2 are depicted in Figure 9.

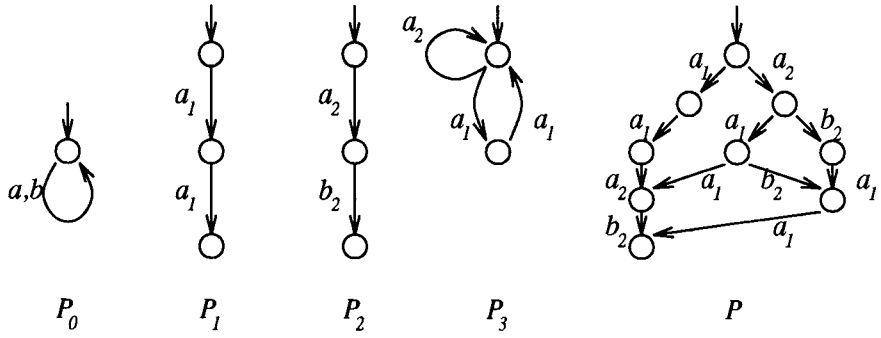

Figure 9: Diagram illustrating Example 11

Now suppose we decide to impose the plausible requirement that the two consecutive operations of part 1 on machine A not be interrupted by the operation of part 2 on machine A. This is a restriction on possible concurrency. It can be represented by a process object $((P_3, \hat{P}_3), C_3, M_3)$ where $\hat{P}_3$ is null, $b_i \in C_3(\epsilon) \cap M_3(\epsilon)$, and the NSM P_3 is depicted in Figure 9. [12] Note that since the concurrency restriction does not involve operations on machine B,

[12] A process object that models a passive constraint does not generate events and hence has a real part that is null.

such operations are completely uncontrollable and completely unobservable to the process that models this restriction. Let $((P, \hat{P}), C, M)$ denote the masked composition of the process objects $\{((P_i, \hat{P}_i), C_i, M_i) | \ i = 0, \ldots, 3\}$. It is straightforward to verify that

$$\hat{P} = P = \|_{i=0}^{3} P_i^{C_i M_i},$$

where the NSM P is depicted in Figure 9. As required,

$$L(P) = pr\{a_1 a_1 a_2 b_2, \ a_2 b_2 a_1 a_1, \ a_2 a_1 a_1 b_2, \ a_2 a_1 b_2 a_1\}.$$

In addition to its ability to accomodate concurrency specifications, the process object/masked composition approach to object-oriented design has two other advantages. (1) Since masked composition is associative, it is well-suited to *modular design*. In contrast, an approach based on the operators of full synchronous composition and interleaving composition is not amenable to modular design since full synchronous composition does not distribute over interleaving composition. (2) Since control and observation properties are encapsulated in process objects as submodules that are completely independent of the logic submodule, the software encoding the logic need never be modified to take into account new actuator or sensor limitations. This is in contrast to the situation in [8, 7] where such limitations (e.g., the uncontrollability of certain events) require customization of the logic to ensure that the logic is compatible with these limitations (e.g., no uncontrollable events are disabled). In our approach, a change in actuator/sensor capabilities simply requires changing the masks in the process object; no change in the NSM is required since there are no compatibility conditions such as supervisor completeness. [13]

6 Conclusion

In this paper, we have introduced a new framework for the modeling and supervisory control of discrete event systems. This framework is based on the paradigm of process objects and masked composition. Process objects encapsulate control and observation interfaces with nondeterministic state machines that describe the process logic. In order to permit the interconnection operation of masked composition to be associative, the NSM must include virtual, as well as real, transitions.

Masked composition is a binary operation on process objects which can be used both to build complex plant models from simple components, and to interconnect plant and supervisor process objects in order to impose control. Most of the usual parallel composition operators may be viewed as special cases of masked composition. Since it is associative, masked composition is suitable for layered control synthesis and modular design.

In order to characterize the languages that are realizable under control, the concepts of (C, M)-closed language and (C, M)-invariant process have been introduced. The (nonempty)

[13] Of course, a change in actuator/sensor capabilities can affect whether a particular supervisor process object yields a controlled system that meets the specifications. (See Example 10.)

prefix-closed (C, M)-closed languages are precisely the languages that can be generated by (C, M)-invariant processes. In standard automata theory, any language that can be generated by a finite-state nondeterministic state machine (with ϵ-transitions) can be generated by a finite-state *deterministic* state machine. This fact is largely responsible for the limited modeling role of NSM's in the traditional theory. When process logic is encapsulated with control and observation interfaces, the situation is different. In general, the class of languages that can be generated by (C, M)-invariant processes is strictly larger than the class of languages that can be generated by deterministic (C, M)-invariant processes. This is responsible for the fact that nondeterministic supervision can achieve controlled behavior that is not attainable under deterministic supervision. However, in the special case where the observation mask of the supervisor refines the control mask, any behavior obtainable via nondeterministic supervision is also obtainable via deterministic supervision.

We have derived necessary and sufficient conditions for a given language to be obtainable as the generated language of the controlled system. The key requirement is that the language be (C_1, M_1)-closed relative to the plant process object. This condition is polynomially testable, and whenever it holds, a (nondeterministic) supervisor of polynomial size can be synthesized. This is in contrast to the traditional supervisory control where the synthesis of the (deterministic) supervisor when it exists is known to be an NP-complete problem [34]. In the case of nondeterministic supervision, the class of languages that are realizable as the generated language of the controlled system is closed under union. Consequently, given a specification language, a supremal realizable sublanguage always exists. This circumvents the problems in conventional supervisory control related to the "misbehavior" of observability relative to unions. We have also derived analogous necessary and sufficient conditions for a given language to be obtainable as the generated language of the controlled system under the restriction that the supervisor process object be deterministic.

The new framework permits the synthesis of supervisors for reactive systems, as well as object-oriented control design for systems such as database and manufacturing systems that have specifications restricting possible concurrency. By allowing nondeterministic supervision as well as open-loop control, successful control design can be accomplished for systems that cannot be adequately controlled using the traditional approach to supervisory control.

References

[1] S. Balemi, G. J. Hoffmann, P. Gyugyi, H. Wong-Toi, and G. F. Franklin. Supervisory control of a rapid thermal multiprocessor. *IEEE Transactions on Automatic Control*, 38(7):1040–1059, July 1993.

[2] Y. Brave and M. Heymann. On optimal attraction of discrete-event processes. *Information Sciences*, 67(3):245–276, 1993.

[3] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya. Supervisory control of discrete event processes with partial observation. *IEEE Transactions on Automatic Control*, 33(3):249–260, 1988.

[4] B. J. Cox. *Object Oriented Programming.* Addison Wesley, Reading, Massachusetts, 1986.

[5] S. Adiga (ed.). *Object-Oriented Software for Manufacturing Systems.* Chapman and Hall, London, 1993.

[6] M. Fabian and B. Lennartson. Distributed objects for real-time control systems. In *Proceedings of the 12th IFAC World Congress*, Sidney, Australia, 1993.

[7] M. Fabian and B. Lennartson. Object-oriented supervisory control with a class of nondeterministic specifications. In *Proceedings of 1994 IEEE Conference on Decision and Control*, pages 3634–3635, Lake Buena Vista, Florida, December 1994.

[8] M. Fabian and B. Lennartson. Petri nets and control synthesis: An object-oriented approach. In *Proceedings of the I.M.S.*, Vienna, 1994.

[9] C.H. Golaszewski and P.J. Ramadge. Supervisory control of discrete event processes with arbitrary controls. In M.J. Denham and A.J. Laub, editors, *Advanced Computing Concepts and Techniques (NATO ASI Series, Vol. F47)*, pages 459–469. Springer-Verlag, 1988.

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

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

[12] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice Hall, Inc., Englewood Cliffs, NJ, 1985.

[13] L. E. Holloway and B. H. Krogh. Synthesis of feedback control logic for a class of controlled Petri nets. *IEEE Transactions on Automatic Control*, 35(5):514–523, May 1990.

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

[15] K. Inan. Nondeterministic supervision under partial observations. In G. Cohen and J.-P. Quadrat, editors, *11th International Conference on Analysis and Optimization of Systems: Discrete Event Systems*, pages 39–48. Springer-Verlag, 1994.

[16] C.P. Jobling, P.W. Grant, H.A. Baker, and P. Townsend. Object-oriented programming in control system design: a survey. *Automatica*, 30(8):1221–1261, 1994.

[17] R. Kumar. Formulas for observability of discrete event dynamical systems. In *Proceedings of 1993 Conference on Information Sciences and Systems*, pages 581–586, Johns Hopkins University, Baltimore, MD, March 1993.

[18] R. Kumar and V. K. Garg. Optimal supervisory control of discrete event dynamical systems. *SIAM Journal of Control and Optimization*, 1992. Accepted.

[19] R. Kumar, V. K. Garg, and S. I. Marcus. On controllability and normality of discrete event dynamical systems. *Systems and Control Letters*, 17(3):157–168, 1991.

[20] R. Kumar and V.K. Garg. *Modeling and Control of Logical Discrete Event Systems.* Kluwer, Boston, 1994.

[21] R. Kumar and M. A. Shayman. Nonblocking supervisory control of nondeterministic discrete event systems. In *Proceedings of the American Control Conference*, pages 1089–1093, Baltimore, MD, June 1994.

[22] R. Kumar and M. A. Shayman. Supervisory control of nondeterministic systems under partial observation. In *Proceedings of the IEEE Conference on Decision and Control*, pages 3649–3654, Lake Buena Vista, FL, December 1994.

[23] S. Lafortune and F. Lin. On tolerable and desirable behaviors in supervisory control of discrete event systems. *Discrete Event Dynamical System: Theory and Application*, 1(1):61–92, 1991.

[24] F. Lin and W. M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44(3):173–198, 1988.

[25] K. M. Passino and P. J. Antsaklis. On the optimal control of discrete event systems. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2713–2718, Tampa, FL, December 1989.

[26] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.

[27] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of IEEE: Special Issue on Discrete Event Systems*, 77:81–98, 1989.

[28] K. Rudie and W. M. Wonham. The infimal prefix closed and observable superlanguage of a given language. *Systems and Control Letters*, 15(5):361–371, 1990.

[29] R. Sengupta and S. Lafortune. Optimal control of a class of discrete event systems. In *Proceedings of 1991 IFAC symposium on distributed intelligence systems*, pages 25–30, Arlington, VA, August 1991.

[30] M. A. Shayman and R. Kumar. A new framework for supervisory control. In *Proceedings of American Control Conference*, pages 3141–3145, Seattle, WA, 1995. To appear.

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

[32] M.A. Shayman and R. Kumar. A new approach to supervisory control. In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, pages 861–870, Allerton, IL, September 1994.

[33] J. A. Stiver and P. J. Antsaklis. On the controllability of hybrid control systems. In *Proceedings of the IEEE Conference on Decision and Control*, pages 294–299, San Antonio, TX, December 1993.

[34] J. N. Tsitsiklis. On the control of discrete event dynamical systems. *Mathematics of Control Signals and Systems*, 2(2):95–107, 1989.