Supervisory Control Architecture for Discrete-Event Systems

Lei Feng, *Member, IEEE*, and W.M. Wonham, *Life Fellow, IEEE*

Abstract

A flexible decentralized and hierarchical architecture is presented to reduce computational effort in designing optimal nonblocking supervisors for discrete-event systems (DES). We organize a DES into modular subsystems that embody internal interacting dependencies. Verification of, and coordination among modular subsystems are achieved through their model abstractions. Sufficient conditions are presented to guarantee that coordinators and modular supervisors result in maximally permissive and nonblocking control. A medium-sized example demonstrates the computational effectiveness of our approach.

Index Terms

Discrete-event systems, decentralized control, hierarchical control, model abstraction, observer.

I. Introduction

A fundamental obstruction to the development of *supervisory control theory* (SCT) of *discrete-event systems* (DES) [1], [33] is the computational complexity of synthesizing maximally permissive and nonblocking supervisors. Indeed, the nonblocking supervisory control problem for DES is NP-hard [7]. In current algorithms, the space (and time) required are exponential in the number of plant components and control specifications included in the DES model. Researchers are therefore seeking effective control methods for various subclasses of discrete-event systems that enjoy special structure. Such structure will admit *modularity* [2], [13], [20], [21], [27] and *model abstraction* [8], [11], [12], [30] to circumvent computing global dynamic models. This paper presents a control architecture that flexibly integrates decentralized and hierarchical control to mitigate the computational complexity.

An ideal structure allows a system to be divided into mutually nonconflicting modules; for instance, modules that share no joint events. To identify more general conditions that guarantee mutually nonconflicting modules, [10], [13], [19], [27] introduce various sufficient conditions to ensure that the conjunction of modular or decentralized supervisors is equivalent to the monolithic supervisor. These conditions, however, may fail to hold for many realistic discrete-event systems. In this paper we seek alternative decentralized control methods based on weaker conditions.

As is well known, the exponential complexity of supervisor design arises from synchronizing subsystems into a global system model. To avoid this, Leduc *et al.* [11], [12] partition systems into *master* and *slave* subsystems which

The authors are with the Systems Control Group, Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. e-mail:fenglei,wonham@control.utoronto.ca

need not be synchronized for design purposes. Instead, master subsystems call for services from slave subsystems solely through *interfaces*, which pass service requests from masters to slaves, and confirmation of services performed from slaves to masters. As long as master and slave subsystems satisfy certain "level-wise consistency" conditions with respect to their interfaces, the original system is provably nonblocking under decentralized control.

The success of the interface-based approach is due to model abstraction and information hiding. In SCT, model abstraction of a DES is achieved by a *causal reporter* map [30], [32], [33], which reports only events related to external control specifications and relevant plant components. The interface models designed intuitively in [11], [12] turn out to be model abstractions obtained formally by reporting only the start and finish events of subsystem activity. In this paper we generalize the interface-based approach as a formal method.

To this end, we substitute model abstractions for the original subsystems. If two subsystems share only a small number of common events, their abstractions tend to be small, and either verifying the nonconflicting property (if it holds) or designing a *coordinator* to achieve it, may require only modest effort. Loosely stated, a coordinator is a decentralized supervisor that does not enforce any given control specification but only resolves conflict among other decentralized supervisors. Crucial to successful model abstraction is that the reporter map be as coarse as possible subject to preserving the information needed for reliable representation of the nonblocking property. The most effective model abstraction operator in SCT is the causal reporter map having the *observer* property [28], [30]–[32].

We note that, with *state tree* structures [15] and *binary decision diagram* (BDD) encoding of the constituent DES, systems with state sizes beyond 10^{20} states have been treated. This suggests that the interface architecture combined with BDDs could be especially efficient [23].

In this paper a DES is assumed to consist of a network of simple plant components subject to a conjunction of modular control specifications. The *shuffle* system structure defined in [5] is an instance. The structure assumed in this paper is, however, more general, inasmuch as plant component alphabets need not be pairwise disjoint and control specifications are not restricted to *buffers* or *servers* as in [5].

Given a DES control problem with networked structure, our architectural approach to supervisory control design is the following.

Step 1. Obtain decentralized supervisors for individual specifications using the method introduced in Section III. Because a specification often imposes restrictions involving only a subset of the alphabet of the DES, the decentralized supervisor for a specification can be synthesized from its "local" plant [2], usually the synchronous product of plant components that share joint events with the specification. Such a supervisor can often be found by inspection.

Step 2. Partition the plant components and decentralized supervisors into modular subsystems according to their interaction dependencies. *Control-flow decomposition* [5] is an effective method for selecting subsystems appropriately. If each subsystem contains only a few plant components and decentralized supervisors, or admits simple control logic, we can find its maximally permissive and nonblocking control with only modest computation or by qualitative reasoning [5]. The main challenge is to treat the interactions among subsystems.

Step 3. Design the model abstraction of each subsystem using *natural observers*, whose properties are explained

in Section II. Abstraction introduces hierarchy into system structure, as it reports only the events shared with other subsystems and conceals the rest. The fewer the reported events, the greater state reduction will be achieved.

Step 4. Organize these model abstractions into groups according to their interconnections. In simple cases, all the model abstractions will belong to just one group. For each group, check whether these model abstractions are nonconflicting; if they are not, design a coordinator to resolve conflict by the method in Proposition 7. In the pleasant circumstance that the system is a *product* system with pairwise disjoint alphabets, the method in Section IV is applicable.

Step 5. Repeat Steps 3 and 4 by regarding the model abstractions as higher-level plant components and the groups of model abstractions as subsystems according to Step 2, until there is only one group in Step 4. The end result will be a hierarchy of decentralized supervisors and coordinators. The paper presents conditions sufficient to guarantee that these decentralized supervisors and coordinators provide maximally permissive and nonblocking supervisory control equivalent to the monolithic controller.

Section V estimates the computational complexity of the proposed approach and discusses how to apply the latter effectively. Section VI demonstrates its efficiency applied to a well-known example. The latter suggests that we can obtain intelligible decentralized supervisors with minimal computational effort by supplementing the architectural approach with control-flow decomposition [5].

Our approach exploits a (more-or-less universal) architecture also utilized in [19]–[21], with the difference that we do not require *a priori* that the modular subsystems be nonconflicting. In fact, coordinating these modules to eliminate blocking is in our view the main challenge and is our primary objective.

Recently Hill and Tilbury [8] have proposed an incremental hierarchical computational approach, which uses control architecture and model abstraction similar to this paper and its precursor [4]. Their approach does not, however, allow subsystems at the same level to share events. The result is a slender hierarchy with more levels. Furthermore, the supervisors in [8] may combine control actions for enforcing control specifications with actions that resolve system blocking. In contrast, our approach will result in a greater number of relatively simple decentralized supervisors: some are dedicated to the given control specifications, preferably in one-to-one correspondence, while others are dedicated only to resolving conflicts. Finally, the approach in [8] cannot guarantee maximally permissive control.

II. Natural Observers

For a system described by a language $L \subseteq \Sigma^*$ and a *causal reporter map* θ [30], [33], the *model abstraction* of the system is the induced system representing language $\theta(L)$. Especially important in SCT are reporter maps with the *observer* property [30]. While [28] treated hierarchical control using general causal reporter maps, this paper will construct model abstractions only with *natural observers*, i.e., *natural projections* [1], [33] with the observer property. This simplification better matches our more practical perspective. In Section II-A we review the properties of natural projections that enable compositional computation. In Section II-B we state special properties of the natural observer and present a few applications.

A. Properties of Natural Projections

Recall [1], [33] the definitions of natural projection and its inverse image function. Standard simple properties include the following.

Proposition 1: Given event sets Σ and $\Sigma_0 \subseteq \Sigma$, consider the natural projection $P : \Sigma^* \to \Sigma_0^*$ and four arbitrary languages $A, B \subseteq \Sigma^*$ and $C, D \subseteq \Sigma_0^*$. Then

1) P is surjective, i.e., Im $P = \Sigma_0^*$.
2) P is idempotent, namely, $P \circ P = P$.
3) P and P^{-1} are monotone.
4) $P \circ P^{-1} = 1$, where 1 is the identity function on Σ_0^*, and $P^{-1}P(A) \supseteq A$.
5) P and P^{-1} are prefix-preserving, i.e., $\overline{P(A)} = P(\overline{A})$ and $\overline{P^{-1}(C)} = P^{-1}(\overline{C})$.
6) $P(A \cap B) \subseteq P(A) \cap P(B)$.
7) P is a morphism of set-union, i.e., $P(A \cup B) = P(A) \cup P(B)$.
8) P^{-1} is a morphism of set-intersection, i.e., $P^{-1}(C \cap D) = P^{-1}(C) \cap P^{-1}(D)$.
9) P^{-1} is a morphism of set-union, i.e., $P^{-1}(C \cup D) = P^{-1}(C) \cup P^{-1}(D)$.

Proof: See [3]. ■

Proposition 2: Consider two alphabets Σ_1, Σ_2, and their intersection $\Sigma_0 := \Sigma_1 \cap \Sigma_2$. Define the natural projections $P_i : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_i^*$ $(i = 0, 1, 2)$ and $Q_j : \Sigma_j^* \to \Sigma_0^*$ $(j = 1, 2)$. Extend these natural projections to the corresponding image functions [16] and denote them with the same symbols as the functions, i.e., $P_i : 2^{(\Sigma_1 \cup \Sigma_2)^*} \to 2^{\Sigma_i^*}$ $(i = 0, 1, 2)$, $Q_j : 2^{\Sigma_j^*} \to 2^{\Sigma_0^*}$ $(j = 1, 2)$. Then for $i, j = 1, 2$ and $i \neq j$,

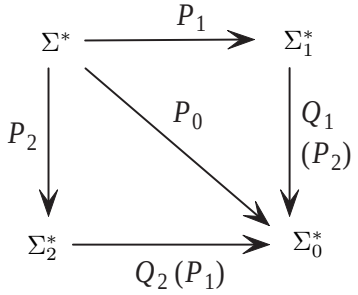
1) $Im\ P_i|\Sigma_j^* = (Im\ P_i) \cap \Sigma_j^* = \Sigma_0^*$. With codomain restricted to its image, $\Sigma_0^*|P_i|\Sigma_j^*$ is identical to the natural projection Q_j.
2) $P_0|\Sigma_i^* = Q_i$.
3) P_1 and P_2 commute, i.e., $P_i \circ P_j = Q_j \circ P_j = P_0 = Q_i \circ P_i = P_j \circ P_i$.
4) P_i and P_0 commute, i.e., $P_i \circ P_0 = P_0 = P_0 \circ P_i$.
5) $P_i^{-1} \circ Q_i^{-1} = P_0^{-1}$.
6) $P_i \circ P_j^{-1} = Q_i^{-1} \circ Q_j$.

Proof: See [3]. ■

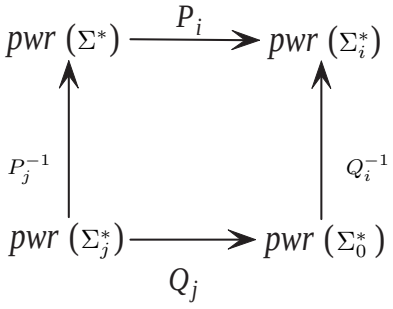
Proposition 2 is summarized by the commutative diagrams in Fig. 1, where $\Sigma := \Sigma_1 \cup \Sigma_2$. Statements 3 and 4 imply that the composition of natural projections is a natural projection whose observable event set is the intersection of the observable event sets of the factors. In particular, composition is associative.

Corollary 1: Consider a family of natural projections $P_i : \Sigma_i^* \to \Upsilon_i^*$ $(i \in \mathbf{n})$. Let $\Sigma := \Sigma_1$ and $\Upsilon := \bigcap_{i=1}^n \Upsilon_i$. Define the new natural projection $Q : \Sigma^* \to \Upsilon^*$. Suppose $\Upsilon_i \subseteq \Sigma_{i+1}$ $(i \in \mathbf{n} - \mathbf{1})$. Then $Q = P_n \circ P_{n-1} \circ \cdots \circ P_1$.

Turning to supervisory control, consider a discrete-event system consisting of two components, with languages L_i over alphabets Σ_i $(i = 1, 2)$. We first recall the definition of *synchronous product* [33] $L_1||L_2$. Let $P_i : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_i^*$ $(i = 1, 2)$ be the natural projections with inverse image functions $P_i^{-1} : 2^{\Sigma_i^*} \to 2^{(\Sigma_1 \cup \Sigma_2)^*}$ $(i = 1, 2)$. Then



(a) Statement 3: $P_i \circ P_j = P_0 = P_j \circ P_0$



(b) Statement 6: $P_i \circ P_j^{-1} = Q_i^{-1} \circ Q_j$

Here $pwr(X) := 2^X$

Fig. 1. Commutative Diagrams for Proposition 2.

$L_1||L_2 := P_1^{-1}(L_1) \cap P_2^{-1}(L_2)$.

A string $s \in L_1||L_2$ if and only if $s \in (\Sigma_1 \cup \Sigma_2)^*$ and has the property $P_i(s) \in L_i$ $(i = 1, 2)$. Let $t_i := P_i(s)$ $(i = 1, 2)$. By Proposition 2, $P_2(t_1) = P_1(t_2)$, namely, strings t_1 and t_2 have the same image in their joint event set $\Sigma_1 \cap \Sigma_2$. This makes clear the form of strings in $L_1||L_2$. If two strings $t_i \in L_i$ $(i = 1, 2)$ contain the same events in $\Sigma_1 \cap \Sigma_2$ and these occur in the same sequence, then t_1 and t_2 synchronize by merging common events in $\Sigma_1 \cap \Sigma_2$ and arbitrarily shuffling events outside $\Sigma_1 \cap \Sigma_2$.

Lemma 1: Define the natural projections P_i and Q_j as in Proposition 2 for $i = 0, 1, 2$ and $j = 1, 2$. For any two languages $L_i \subseteq \Sigma_i^*$ $(i = 1, 2)$ we have $L_1 \| L_2 \neq \emptyset \Leftrightarrow Q_1 L_1 \cap Q_2 L_2 \neq \emptyset$.

Proof: See [3] and Appendix. ■

To obtain an abstraction of the system, we could first compute the system's global behavior $L_1||L_2$ and then its projection. When, however, the shared events of the two components are all observable the result is obtained more economically from abstractions of the components, according to the following (Exercise 3.3.7 in [33]). This result is central to our method.

Proposition 3: Let $L_i \subseteq \Sigma_i^*$ $(i = 1, 2)$. Assume $\Sigma_0 \subseteq \Sigma := \Sigma_1 \cup \Sigma_2$, $P_i : \Sigma^* \to \Sigma_i^*$ $(i = 0, 1, 2)$, $Q_j : \Sigma_j^* \to (\Sigma_j \cap \Sigma_0)^*$ $(j = 1, 2)$. If $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma_0$, then $P_0(L_1 \| L_2) = Q_1(L_1) \| Q_2(L_2)$.

Proof: See [3] and Appendix. ■

The result is displayed by the commutative diagram, Fig. 2, where $\Sigma_{10} := \Sigma_1 \cap \Sigma_0$ and $\Sigma_{20} := \Sigma_2 \cap \Sigma_0$. The extension to an arbitrary number of synchronized factors is straightforward [24].

B. Observer

Consider a DES described by language L over alphabet Σ. Given an observable event subset Σ_0, we can define the natural projection $P : \Sigma^* \to \Sigma_0^*$ and then the abstraction $P(L)$. Since $P(L)$ is obtained from L through *partial observation*, the abstraction may remove critical information and be inconsistent with the original DES with respect

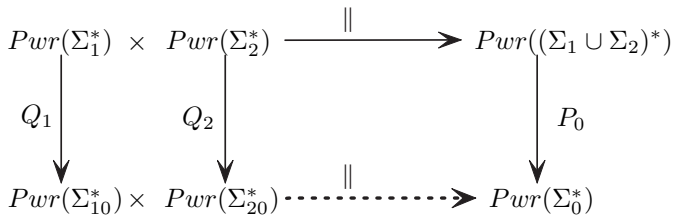


Fig. 2. Proposition 3

to controllability and nonblocking. For instance, the projection of a blocking DES could be nonblocking, so a nonblocking supervisor designed from the abstraction could result in a blocking supervisor for the original DES. To avoid this pitfall, one must carefully select the observable events of a DES.

Fig. 3 displays a natural projection with a "good" selection. Whenever an observed string, say $P(s), s \in \overline{L}$, can reach a marker state in the abstracted model via a string $t \in \Sigma_0^*$, the original system, at string s, must be able to reach a marker state from s, via some string $u \in \Sigma^*$ such that $P(su) = P(s)t$, as illustrated in Fig. 3. Briefly, what we expect in the observation model is surely realizable in the original model. If so, a nonblocking supervisor for the abstracted model is also nonblocking for the original system. A natural projection defined by such a "good" observable event set is called an *observer* [28], [30].

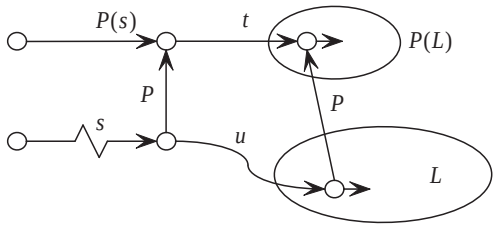


Fig. 3. Observer

*Definition 1: [Observer [30]] Assume language $L \subseteq \Sigma^*$ and let $\Sigma_0 \subseteq \Sigma$ be an observable event subset. The natural projection $P : \Sigma^* \to \Sigma_0^*$ is an L-*observer *if*

$$(\forall t \in P(L))(\forall s \in \overline{L})\, P(s) \leq t \implies (\exists u \in \Sigma^*)\, su \in L \,\&\, P(su) = t.$$

$\diamond$

We call a natural projection with the observer property a *natural observer*. The symbol $\leq$ means that the first string is a prefix of the second one [1], [33]. If $\Sigma_0 = \Sigma$ or $\emptyset$, P is automatically an L-observer. The observer concept is equivalent [30] to that of *observation equivalence* introduced by Milner [17]. An observer is just a function whose equivalence kernel is a *quasi-congruence* [32] of the dynamic system modeled by L. Denote by $||L||$ the state size of the *canonical recognizer* [33] of L. If the natural projection P is an L-observer, the abstraction

$P(L)$ can be computed in polynomial time in $||L||$, and $||P(L)|| \leq ||L||$ [29]. In fact, an arbitrary natural projection can be modified in polynomial time to be a natural observer by enlarging the observable event set [6].

Proposition 4: An L-observer is also an $\overline{L}$-observer.

Proof: See [3]. ■

Proposition 4 implies that the requirement of L-observer is stronger than $\overline{L}$-observer, because L embodies all the "dynamics" in $\overline{L}$ together with extra information about marked strings. The result follows easily from Definition 1.

If a system consists of more than one plant component, for instance, $L := L_1 \| L_2$, it would be more economical to check the observer property component-wise without computing the synchronous product first. Proposition 5 presents a sufficient condition for this simplification to be valid.

Proposition 5: Consider two languages $L_i \subseteq \Sigma_i^*$ $(i = 1, 2)$. Define the natural projections $P_i : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_i^*$ $(i = 0, 1, 2)$, where $\Sigma_0 \subseteq \Sigma_1 \cup \Sigma_2$. If $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma_0$ and for both $i = 1, 2, P_0|\Sigma_i^*$ is an L_i-observer, then P_0 is an $L_1 \| L_2$-observer.

Proof: See [3] and Appendix. ■

Proposition 8 in [31] is a special case where $\Sigma_1 \cap \Sigma_2 = \emptyset$. Our application of the observer property will be to guarantee nonblocking control for a partially observed system, according to Theorem 6 in [30]. Replacing the causal reporter map θ with the natural projection $P : \Sigma^* \to \Sigma_0^*$, we can affirm that if P is an L-observer for a language $L \subseteq \Sigma^*$, then

$$(\forall N \subseteq P(L))\, P^{-1}(\overline{N}) \cap \overline{L} = \overline{P^{-1}(N) \cap L}. \tag{1}$$

In particular, if N is a controllable sublanguage for the abstracted model $P(L)$, Equation (1) means that its inverse projection $P^{-1}(N)$ is nonconflicting with the original system L. Hence N is a nonblocking supervisory control under partial observation.

If two languages $L_i \subseteq \Sigma_i^*$ $(i = 1, 2)$ have the same alphabet, i.e., $\Sigma_1 = \Sigma_2$, they are *nonconflicting* [34] provided $\overline{L_1 \cap L_2} = \overline{L_1} \cap \overline{L_2}$. For the general case where $\Sigma_1 \neq \Sigma_2$, we define

Definition 2: [Synchronously Nonconflicting [33]] Two languages $L_i \subseteq \Sigma_i^*, i = 1, 2$, are synchronously nonconflicting *if $\overline{L_1 \| L_2} = \overline{L_1} \| \overline{L_2}$.* ◇

Languages L_1 and L_2 are synchronously nonconflicting if and only if $P_1^{-1}(L_1)$ and $P_2^{-1}(L_2)$ are nonconflicting, where P_i are the natural projections $(\Sigma_1 \cup \Sigma_2)^* \to \Sigma_i^*$ $(i = 1, 2)$. Similarly, we say n (≥ 2) languages L_i $(i \in \mathbf{n})$ are synchronously nonconflicting if they satisfy $\overline{\|_{i=1}^n L_i} = \|_{i=1}^n \overline{L_i}$. According to [34], if two languages $K_1, K_2 \subseteq L$ are nonconflicting, and each is controllable with respect to a prefix closed language $L \subseteq \Sigma^*$, with $\Sigma_u \subseteq \Sigma$, then $K_1 \cap K_2$ is also controllable with respect to L and Σ_u. Likewise, if K_1 and K_2 have different alphabets, we have a similar result on the controllability of their synchronous product $K_1 \| K_2$, as generalized in Proposition 6.

Proposition 6: For $i \in \mathbf{n}$, let $K_i \subseteq L_i \subseteq \Sigma_i^*$ be controllable with respect to $\Sigma_{i,u} \subseteq \Sigma_i$ and a prefix closed language L_i. If the K_i are synchronously nonconflicting, then $||_{i=1}^n K_i$ is controllable with respect to $\bigcup_{i=1}^n \Sigma_{i,u}$ and $||_{i=1}^n L_i$.

Proof: See [3]. ■

An observer determines a "reliable interface" for a DES, hence the interaction between two complex DES may be examined through their projections. If P_0 has the observer property, we can check if two languages L_1 and L_2 are synchronously nonconflicting by checking whether their projections $P_0(L_1)$ and $P_0(L_2)$ are synchronously nonconflicting. Since the automaton models of $P_0(L_i)$ are smaller than those of L_i, we may save significant computational effort, in accordance with the following.

Theorem 1 (Synchronously Nonconflicting Criterion): Let $L_i \subseteq \Sigma_i^*$ $(i = 1, 2)$, and $\Sigma_0 \supseteq \Sigma_1 \cap \Sigma_2$. If $Q_i : \Sigma_i^* \to (\Sigma_i \cap \Sigma_0)^*$ are L_i-observers $(i = 1, 2)$, then $\overline{L_1 \| L_2} = \overline{L_1} \,\|\, \overline{L_2}$ if and only if $\overline{Q_1(L_1) \| Q_2(L_2)} = Q_1(\overline{L_1}) \,\|\, Q_2(\overline{L_2})$.

Proof: Define the natural projections $P_i : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_i^*$ $(i = 0, 1, 2)$ and $T : \Sigma_0^* \to (\Sigma_1 \cap \Sigma_2)^*$. Let $R_i := P_i | \Sigma_0^*$ $(i = 1, 2)$. These projections are illustrated by the commutative diagram, Fig. 4.

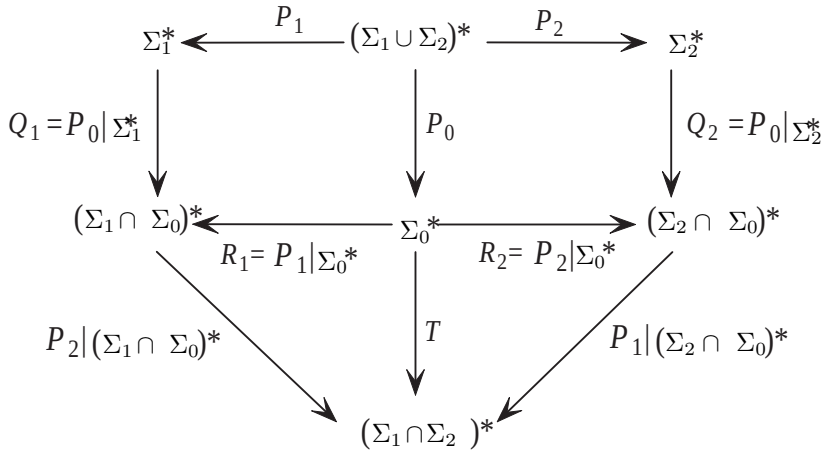


Fig. 4. Natural Projections When $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma_0$

(If) We always have $\overline{L_1 \| L_2} = \overline{P_1^{-1}(L_1) \cap P_2^{-1}(L_2)} \subseteq P_1^{-1}(\overline{L_1}) \cap P_2^{-1}(\overline{L_2}) = \overline{L_1} \,\|\, \overline{L_2}$. For the reverse inclusion, let $s \in \overline{L_1} \| \overline{L_2}$. We must show that there exists a string $u \in (\Sigma_1 \cup \Sigma_2)^*$ with $su \in L_1 \| L_2$. Since $s \in \overline{L_1} \| \overline{L_2}$,

$$P_i(s) \in \overline{L_i}, \ i = 1, 2. \tag{2}$$

Moreover, $P_0(s) \in P_0(\overline{L_1} \| \overline{L_2})$. Because of the assumption $\Sigma_0 \supseteq \Sigma_1 \cap \Sigma_2$ and Proposition 3,

$$P_0(s) \in Q_1(\overline{L_1}) \| Q_2(\overline{L_2}).$$

By the assumption $\overline{Q_1(L_1) \| Q_2(L_2)} = Q_1(\overline{L_1}) \,\|\, Q_2(\overline{L_2})$, we have $P_0(s) \in \overline{Q_1(L_1) \| Q_2(L_2)} = \overline{P_0(L_1 \| L_2)}$. Then there must exist a string $t \in \Sigma_0^*$ such that $P_0(s)t \in P_0(L_1 \| L_2)$, and therefore

$$R_i[P_0(s)t] \in R_i P_0(L_1 \| L_2), \ i = 1, 2.$$

From Fig. 4, we see that $R_i \circ P_0 = Q_i \circ P_i$ $(i = 1, 2)$, and by Proposition 3, $P_i(L_1 \| L_2) \subseteq L_i$ $(i = 1, 2)$.

Consequently,

$$R_iP_0(s)R_i(t) = Q_iP_i(s)R_i(t) \in Q_i(L_i),\ i = 1, 2.$$

From Equation (2), $P_i(s) \in \overline{L_i}$. Since Q_i is an L_i-observer, there exist strings $w_i \in \Sigma_i^*$ with $P_i(s)w_i \in L_i$ and

$$Q_i(P_i(s)w_i) = Q_iP_i(s)R_i(t), i = 1, 2. \tag{3}$$

Because $Q_i(P_i(s)w_i) = Q_iP_i(s)Q_i(w_i)$, we have $Q_i(w_i) = R_i(t)$ $(i = 1, 2)$. Applying P_j $(j = 1, 2; j \neq i)$ to both sides of this equation, we get

$$P_jQ_i(w_i) = P_jR_i(t) = T(t).$$

According to $\Sigma_0 \supseteq \Sigma_1 \cap \Sigma_2$ and Corollary 1,

$$P_jQ_i(w_i) = P_j(w_i),\ i, j = 1, 2; i \neq j.$$

$$P_2(w_1) = T(t) = P_1(w_2).$$

Therefore, $K := \{w_1\}\|\{w_2\}$ is nonempty by Lemma 1. Taking $u \in K$, we have $P_i(u) = w_i$ $(i = 1, 2)$. According to Equation (3), we have

$$P_i(s)w_i = P_i(s)P_i(u) = P_i(su) \in L_i,\ i = 1, 2.$$

Consequently, $su \in L_1\|L_2$, as required.

(Only if): According to the assumption, we know $\overline{L_1\|L_2} = \overline{L_1}\|\overline{L_2}$. Applying P_0 to both sides, we get by Proposition 3,

$$P_0(\overline{L_1\|L_2}) = \overline{P_0(L_1\|L_2)} = \overline{Q_1(L_1)\|Q_2(L_2)},$$

$$P_0(\overline{L_1}\|\overline{L_2}) = Q_1(\overline{L_1})\|Q_2(\overline{L_2}).$$

Hence $\overline{Q_1(L_1)\|Q_2(L_2)} = Q_1(\overline{L_1})\|Q_2(\overline{L_2})$. ■

In case the two languages L_1 and L_2 are synchronously conflicting, a third language L_0, called a *coordinator*, must be introduced to resolve the conflict. As in Theorem 1, the two languages may interact only "locally", i.e., share only a proper subset of events. In that case, instead of computing the coordinator directly from the two languages themselves, we perform this computation through their abstractions.

Proposition 7: Let $L_i \subseteq \Sigma_i^*$ $(i = 1, 2)$, and $\Sigma_0 \supseteq \Sigma_1 \cap \Sigma_2$. In the notation of Theorem 1, if for $i = 1, 2$, Q_i is an L_i-observer and there is a language $L_0 \subseteq \Sigma_0^*$ which satisfies

$$\overline{Q_1(L_1)||Q_2(L_2)||L_0} = Q_1(\overline{L_1})||Q_2(\overline{L_2})||\overline{L_0}$$

then $\overline{L_1||L_2||L_0} = \overline{L_1}||\overline{L_2}||\overline{L_0}$.

Proof: See [3] and Appendix. ■

The coordinator L_0 depends only upon the event set Σ_0, which contains the shared events of L_1 and L_2 and

defines the required natural observers. As long as L_0 can resolve the conflict between $Q_1(L_1)$ and $Q_2(L_2)$, it will resolve the conflict between L_1 and L_2.

III. Optimal Nonblocking Decentralized Control

A decentralized controller monitors and disables only events in an *observable* event subset. Lin and Wonham [14] found conditions for the equivalence between decentralized and monolithic DES control, but considered only prefix-closed languages and did not address the nonblocking problem. Moreover, their conditions are not easy to check, because *normality* [14], [33] must be verified for the monolithic supervisor. In the previous section we saw that the observer property is a sufficient condition for nonblocking decentralized control. We now extend this to achieve *optimality* (i.e., *maximal permissiveness*). Though [28], [30], [31] have studied the counterpart problems for hierarchical control with a general causal reporter map θ, natural projection endows decentralized control with distinct characteristics: more structure is evident and less computation is necessary.

An optimal supervisor with full observation usually disables the nearest controllable events preceding or "upstream" to a prohibited uncontrollable event (σ, say). If, however, some of these controllable events are unobservable, a decentralized supervisor must disable controllable events further back, and so is more restrictive. For this restriction to be relaxed, the observable event set must be large enough to contain all the upstream controllable events nearest to σ. Such a decentralized supervisor will prevent the occurrence of an uncontrollable event while allowing maximal freedom of system behavior. A projection with such an observable event set is called *output control consistent* (OCC) [35].

Definition 3: [OCC] Let $\Sigma_0, \Sigma_u \subseteq \Sigma$ be the observable and uncontrollable event sets. The natural projection $P : \Sigma^* \to \Sigma_0^*$ is output control consistent *(*OCC*) for the prefix-closed language $L \subseteq \Sigma^*$, if for every string $s \in L$ of the form*

$$s = \sigma_1 \cdots \sigma_k \text{ or } s = s'\sigma_1 \cdots \sigma_k,\ k \geq 1$$

which satisfies the conditions that s' terminates with an event in Σ_0, $\sigma_i \in \Sigma - \Sigma_0$ $(i \in \mathbf{k} - \mathbf{1})$ and $\sigma_k \in \Sigma_0$, we have the property that $\sigma_k \in \Sigma_u \Rightarrow (\forall i \in \mathbf{k})\sigma_i \in \Sigma_u$. $\diamond$

In the definition, when σ_k is observable and uncontrollable, its immediately preceding unobservable events must all be uncontrollable, namely, its nearest controllable event must be observable. This definition is adapted from the same concept in [35], where it was defined for general causal reporter maps. References [33], [35] provide a polynomial algorithm to refine a natural projection to be OCC. Notice that if $\Sigma_0 = \Sigma$ or $\emptyset$, P is automatically OCC for L.

We can now state a practical and concise sufficient condition for optimal nonblocking decentralized control.

Theorem 2 (Optimal Nonblocking Decentralized Control): Let a nonblocking plant be described by closed and marked languages $L, L_m \subseteq \Sigma^*$ with $\overline{L_m} = L$, along with observable and uncontrollable event sets $\Sigma_0, \Sigma_u \subseteq \Sigma$, respectively. Suppose the control specification is $E \subseteq \Sigma_0^*$. If the natural projection $P : \Sigma^* \to \Sigma_0^*$ is an L_m-observer and OCC for L, then

$$\sup \mathcal{C}(E \| L_m, L) = \sup \mathcal{C}_0(E \cap P(L_m), P(L)) \| L_m.$$

Proof: See [3] and Appendix. ■

$\mathcal{C}(E\|L_m, L)$ denotes the family of languages that are sublanguages of $E\|L_m$ and controllable with respect to L and Σ_u [1], [33]; $\sup\mathcal{C}(E\|L_m, L)$ is the supremal element of this family and represents the behavior that the optimal supervisor with full observation can obtain for these data. Similarly $\sup\mathcal{C}_0(E \cap P(L_m), P(L))$ describes the decentralized supervisor with partial observation on Σ_0. When this supervisor is synchronized with the plant, the final controlled behavior is the language on the right side of the equation. The result of Theorem 2 is displayed in the commutative diagram, Fig. 5.

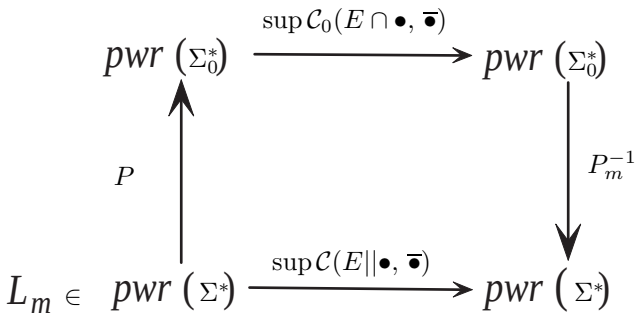


Fig. 5. Optimal Nonblocking Local Control

We can extend Theorem 2 to Proposition 8 to accommodate systems composed from two components.

Proposition 8: Let the plant consist of two nonblocking components with marked languages $M_i \subseteq \Sigma_i^*$ $(i = 1, 2)$. The marked and closed languages of the plant are then $L_m := M_1\|M_2$ and $L := \overline{M_1}\|\overline{M_2}$. Let the observable event subset be $\Sigma_0 \supseteq \Sigma_1 \cap \Sigma_2$ and corresponding natural projection $P_0 : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_0^*$. Suppose the control specification is $E \subseteq \Sigma_0^*$. If for $i = 1, 2$, $P_0|\Sigma_i^*$ is an M_i-observer and OCC for $\overline{M_i}$, then

$$\sup\mathcal{C}(E\|L_m, L) = \sup\mathcal{C}_0[E \cap P_0(L_m), P_0(L)]\|L_m.$$

Proof: See [3] and Appendix. ■

The proposition replaces the languages L_m and L in Theorem 2 with the synchronous products $M_1||M_2$ and $\overline{M_1}||\overline{M_2}$. The replacement of L_m is justified by Proposition 5, and the new requirement for the observer property in Proposition 8 is equivalent to that in Theorem 2. However, the replacement of L is more subtle because the new requirement for the OCC property in Proposition 8 is much weaker than that in Theorem 2. Indeed, when $P_0|\Sigma_i^*$ is OCC for $\overline{M_i}$ $(i = 1, 2)$, P_0 need not be OCC for $\overline{M_1}||\overline{M_2}$.

Notice in particular that M_1 and M_2 need not be synchronously nonconflicting. Hence it is possible for the plant to be blocking, i.e., $\overline{L_m} \subset L$ with strict inclusion. Thanks to Proposition 8, we need not actually compute the global behavior of a networked system. As long as each component is properly abstracted and the abstraction is properly controlled, we can achieve optimal and nonblocking control for the global system with reduced computational effort.

When the plant consists of more than two components, equality between monolithic and decentralized supervisors may also hold. In case the plant components are *independent* agents (i.e., their alphabets are pairwise disjoint), they are necessarily synchronously nonconflicting. By an argument similar to that for Proposition 8, we obtain Corollary 2.

Corollary 2: Let the plant consist of $n \geq 2$ nonblocking, and independent components with marked languages $M_i \subseteq \Sigma_i^*$ $(i \in \mathbf{n})$. Assume $\Sigma_i \cap \Sigma_j = \emptyset$ $(i \neq j)$. The marked and closed languages of the plant are then $L_m := \|_{i=1}^n M_i$ and $L := \|_{i=1}^n \overline{M_i}$. The alphabet of the plant is $\Sigma := \bigcup_{i=1}^n \Sigma_i$, with assumed uncontrollable and observable event subsets Σ_u, Σ_0, respectively. Let the control specification be $E \subseteq \Sigma_0^*$. Define the following natural projections:

$$P_0 : \Sigma^* \to \Sigma_0^*$$

$$Q_i := P_0 | \Sigma_i^* : \Sigma_i^* \to (\Sigma_i \cap \Sigma_0)^*,\ i \in \mathbf{n}.$$

If for $i \in \mathbf{n}$, Q_i is an M_i-observer and OCC for $\overline{M_i}$, then

$$\sup \mathcal{C}(E \| L_m, L) = \sup \mathcal{C}_0[E \cap P_0(L_m), P_0(L)] \, \| L_m.$$

Proof: See [3]. ■

In case the observable event subset is the union of a sub-collection of component alphabets, namely,

$$\Sigma_0 = \bigcup_{j=1}^{m} \Sigma_{\phi j},\ m \leq n$$

where ϕ is a permutation [16] of $\mathbf{n}$, the conditions in the corollary hold automatically. By the corollary the optimal decentralized control of such a product system depends only on the components sharing events with the control specifications. The other unrelated plant components play no role in the control synthesis. This property was pointed out by Queiroz and Cury [2], and Wonham [33], but Corollary 2 extends it to a more general situation.

IV. Control of Product Systems

Having seen how to synthesize the decentralized supervisor for one control specification, one can obtain without difficulty a group of decentralized supervisors for the full set of control specifications imposed on the plant. One must then examine whether there is conflict among these decentralized supervisors and, in that case, design a coordinator to resolve it.

In this section, we consider the supervisory control problem of a plant consisting of $n\,(\geq 1)$ *nonblocking* and *independent* components (i.e., disjoint alphabets) whose marked languages are $M_i \subseteq \Sigma_i^*$ $(i = 1 \in \mathbf{n})$, with $\Sigma_i \cap \Sigma_j = \emptyset$ $(i \neq j)$. The marked and prefix closed languages of the plant are therefore

$$L_m = ||_{i=1}^n M_i \text{ and } L = \overline{L_m} = ||_{i=1}^n \overline{M_i}.$$

The complete alphabet is $\Sigma := \bigcup_{i=1}^n \Sigma_i$.

For definiteness, consider the following problem **3SPEC**, containing three control specifications E_j $(j = 1, 2, 3)$ for the product plant, where each E_j applies only to those plant components whose indices make up the set $N_j \subseteq \mathbf{n}$ $(j = 1, 2, 3)$. Each E_j could in turn be an intersection of simpler specifications on M_i $(i \in N_j)$. Assume $N_i \not\subseteq N_j$ $(i, j \in \{1, 2, 3\}; i \neq j)$, as otherwise two groups could be combined into one. The corresponding subsystems have marked languages

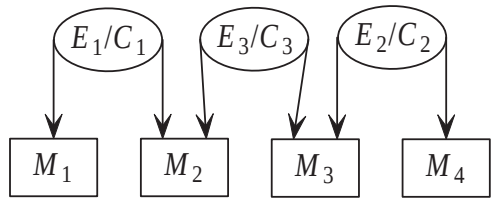
$$H_i := ||_{j \in N_i} M_j, \ i = 1, 2, 3.$$

The subsystem alphabets are $\Upsilon_i := \bigcup_{j \in N_i} \Sigma_j$ $(i = 1, 2, 3)$. Thus $E_i \subseteq \Upsilon_i^*$ $(i = 1, 2, 3)$.

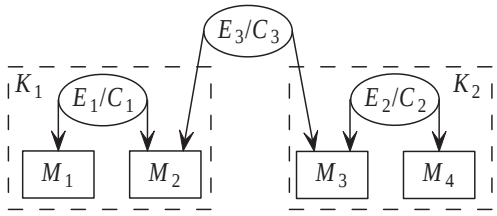
The nonblocking and maximally permissive supervisory control for the system can in principle be computed in one monolithic supervisor as

$$K := \sup \mathcal{C}(E_1||E_2||E_3||L_m, L). \tag{4}$$

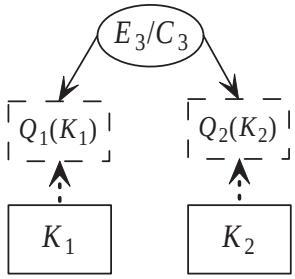
As indicated in the Introduction, our architectural control approach attacks the problem in five steps. We illustrate the procedure through the simple but representative example in Fig. 6. Here $n = 4, N_1 = \{1, 2\}, N_2 = \{3, 4\}$, and $N_3 = \{2, 3\}$.



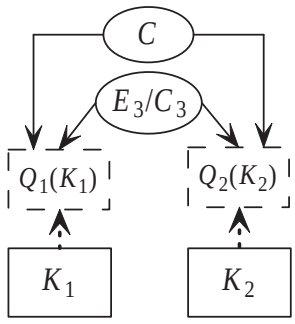
(a) Step 1



(b) Step 2



(c) Step 3



(d) Step 4

Fig. 6. Nonblocking Control of A Product System

As shown in Fig. 6(a), Step 1 is to find decentralized supervisors for all three control specifications, namely the

languages

$$K_i := \sup \mathcal{C}(E_i \cap H_i, \overline{H_i}),\ i = 1, 2, 3. \tag{5}$$

For simpler representation and implementation, we find their reductions [25] C_i $(i = 1, 2, 3)$ such that

$$K_i = C_i || H_i \text{ and } \overline{K_i} = \overline{C_i} || \overline{H_i}. \tag{6}$$

Language C_i results from K_i via *supervisor reduction* [25], [26], which in effect projects out the transition structure of the plant from that of the supervisor. The reduction algorithm in [26] can achieve a reduced supervisor having minimal state size, but the computation generally demands time exponential in the state size of the input supervisor. Since the main theme of this paper is to save computational effort, the general algorithm is not acceptable. Instead we use the algorithm of [25][1], which runs in polynomial time but may or may not yield a minimal state result. This tradeoff is (probably) unavoidable inasmuch as [25] showed that the problem of finding a minimal state reduction is NP-hard. Despite this limitation, the algorithm in [25] often significantly reduces the state size of the supervisor in reasonable time.

In most cases, the alphabet of C_i is smaller than event set Υ_i, i.e., $C_i \subseteq (\Upsilon_i')^*$ for some $\Upsilon_i' \subseteq \Upsilon_i$. Thus more events may be erased through model abstraction.

Since E_j $(j = 1, 2)$ may be the intersection of simpler specifications, each subsystem modeled by K_1 or K_2 may also be the synchronous product of a group of decentralized supervisors. The coordinators of these decentralized supervisors can be attained by monolithic control or successive application of the five-step procedure, depending on the state sizes of the subsystems. In addition to the connection via C_3, the two subsystems may also share common events, although this situation is not reflected in Fig. 6(b).

Rather than verify the nonconflicting property among the three decentralized supervisors, Step 2 organizes the product system and the decentralized supervisors as the hierarchical structure in Fig. 6(b). Here the controlled systems represented by K_1 and K_2 are considered as two plant components which are further regulated by supervisor C_3.

Using the model abstraction technique, Step 3 checks if the product system supervised by the three decentralized supervisors C_i $(i = 1, 2, 3)$ is nonblocking. Since each subsystem is only partially related with C_3, we erase any events not shared with C_3 from the models of K_1 and K_2 through natural observers Q_1 and Q_2 as specified in Theorem 3. Then the verification that K_1, K_2, and K_3 are synchronously nonconflicting is achieved through the simpler computation on $Q_1(K_1)$, $Q_2(K_2)$ and K_3, as specified in Fig. 6(c).

Theorem 3: For **3SPEC**, let $\Upsilon_0 \supseteq \Upsilon_3 \cup (\Upsilon_1 \cap \Upsilon_2)$ and $Q_i : \Upsilon_i^* \to (\Upsilon_i \cap \Upsilon_0)^*$ $(i = 1, 2)$. If the Q_i are K_i-observers $(i = 1, 2)$ and

$$\overline{Q_1(K_1) || Q_2(K_2) || K_3} = Q_1(\overline{K_1}) || Q_2(\overline{K_2}) || \overline{K_3},$$

[1]Implemented as **supreduce** in software XPTCT [33].

where $K_i (i = 1, 2, 3)$ are defined in Equation (5), then K_1, K_2, K_3 are synchronously nonconflicting and $K_1||K_2||K_3$ is the maximally permissive solution to the control problem.

Proof: By Proposition 7, we can immediately show that $\overline{K_1||K_2||K_3} = \overline{K_1}||\overline{K_2}||\overline{K_3}$, namely, the K_i $(i = 1, 2, 3)$ are synchronously nonconflicting. Consequently the existing theory of modular control [1], [2], [33], [34] yields the conclusion

$$K_1||K_2||K_3 = \sup \mathcal{C}(E_1||E_2||E_3||L_m, L). \tag{7}$$

■

In case the condition in Theorem 3 fails, i.e., $Q_1(K_1), Q_2(K_2), K_3$ conflict, we should proceed to Step 4 to design a coordinator. The coordinator design for two subsystems is spelled out in Proposition 7. Using the same idea, we can design a coordination scheme for the three supervisors K_i $(i = 1, 2, 3)$ of the product system.

In principle, the coordinator for the three supervisors is another supervisor for the new plant composed from K_3 and the abstractions of K_1 and K_2. The specification for this high level plant is simply Σ^*, because the only purpose of the coordinator is to resolve the conflict among the three supervisors. This coordination scheme is shown in Fig. 6(d).

The supervisor for the high level plant is then

$$K_C := \sup \mathcal{C}(Q_1(K_1)||Q_2(K_2)||K_3, Q_1(\overline{K_1})||Q_2(\overline{K_2})||\overline{K_3}) \tag{8}$$

Its reduction is language C such that

$$K_C = C||Q_1(K_1)||Q_2(K_2)||K_3 \tag{9}$$

$$\overline{K_C} = \overline{C}||Q_1(\overline{K_1})||Q_2(\overline{K_2})||\overline{K_3} \tag{10}$$

The following theorem provides a sufficient condition for a supervisor recognizing language C to be a coordinator for the three modular supervisors.

Theorem 4: For **3SPEC**, let $\Upsilon_0 \supseteq \Upsilon_3 \cup (\Upsilon_1 \cap \Upsilon_2)$ and $P_0 : \Sigma^* \to \Upsilon_0^*$. Define $Q_i := P_0|\Upsilon_i^*$ $(i = 1, 2)$. If the Q_i are K_i-observers $(i = 1, 2)$ and for $j \in \mathbf{n}$, $P_0|\Sigma_j^*$ are OCC for $\overline{M_j}$, then the language C defined by Equations (8),(9),(10) is a coordinator for supervisors K_i $(i = 1, 2, 3)$ defined by Equations (5), namely,

$$\overline{K_1||K_2||K_3||C} = \overline{K_1}||\overline{K_2}||\overline{K_3}||\overline{C} \tag{11}$$

$$K_1||K_2||K_3||C = \sup \mathcal{C}(E_1||E_2||E_3||L_m, L). \tag{12}$$

Proof: See [3], [4] and Appendix. ■

When the component index sets meet the condition $N_3 \subseteq N_1 \cup N_2$, we can replace language K_3 in Theorem 4 with its reduction C_3, because the plant information embodied in K_3 is already included in K_1 and K_2. Then the implementation of K_C from Equation (8) simplifies to

$$K'_C := \sup \mathcal{C}(Q_1(K_1)||Q_2(K_2)||C_3, Q_1(\overline{K_1})||Q_2(\overline{K_2})||\overline{C_3}). \tag{13}$$

Here supervisor K_3 is replaced by its reduction C_3. Let the reduction of K'_C be C'.

$$K'_C = Q_1(K_1)||Q_2(K_2)||C_3||C' \quad (14)$$

$$\overline{K'_C} = Q_1(\overline{K_1})||Q_2(\overline{K_2})||\overline{C_3}||\overline{C'} \quad (15)$$

Thus C' is the coordinator for the three decentralized supervisors.

Corollary 3: For **3SPEC**, let $N_3 \subseteq N_1 \cup N_2$. Let $C_3 \subseteq (\Upsilon'_3)^*$ be the supervisor reduction of K_3 as defined by Equation (6), $\Upsilon_0 \supseteq \Upsilon'_3 \cup (\Upsilon_1 \cap \Upsilon_2)$ and $P_0 : \Sigma^* \to \Upsilon_0^*$. If $P_0|\Upsilon_i^*$ are K_i-observers ($i = 1, 2$) and $P_0|\Sigma_j^*$ are OCC for $\overline{M_j}$ ($j \in \mathbf{n}$), then the language C' defined by Equations (14) and (15) is a coordinator for supervisors K_i ($i = 1, 2, 3$) defined by Equation (5), namely,

$$\overline{K_1||K_2||K_3||C'} = \overline{K_1}||\overline{K_2}||\overline{K_3}||\overline{C'} \quad (16)$$

$$K_1||K_2||K_3||C' = \sup \mathcal{C}(E_1||E_2||E_3||L_m, L) \quad (17)$$

Proof: See [3]. ■

One advantage of using this corollary is to compute the coordinator with the reduction of supervisor K_3. Another is that the automaton models of the abstractions of supervisors K_1 and K_2 may have smaller state sizes. The alphabet Υ'_3 of supervisor reduction C_3 is normally smaller than Υ_3. Hence the observable event set Υ_0 in the corollary is smaller than that appearing in Theorem 4. More events may be suppressed by the natural projections defined in this corollary.

Step 5 of the architectural control approach may consist of an iteration of Steps 3 and 4, if necessary. In the illustrative example in Fig. 6, there are only two model abstractions and we have already designed the coordinator between them. Hence the computation process for the illustrative example terminates at Step 5 without any further iteration. However, in more complex situations, we regard a group of already coordinated model abstractions as a new subsystem, which is maximally permissive and nonblocking, at a higher level and find its model abstraction again. We repeat the verification and coordination methods presented in this section on these newly generated model abstractions.

V. Complexity Study

This section naively estimates the time and space complexities [22] of the architectural supervisory control approach. We first summarize the computational complexity of the fundamental algorithms. Then the time and space complexities of the control synthesis process proposed in Section IV are estimated and compared with those of monolithic supervision. Based on the complexity study, we suggest how to apply the approach effectively.

Table I lists the computational complexities of the algorithms [18] employed by the architectural control approach. The first column lists the algorithms, the second and third columns list the time and space complexities, and the last column the state sizes of the algorithm outputs. For simplicity we assume that an upper bound of the state sizes of the automaton models of plant components M_i ($i \in \mathbf{n}$) is M and of the specifications is N.

TABLE I
COMPLEXITIES OF FUNDAMENTAL ALGORITHMS

Algorithm	**Time**	**Space**	**Output**
$\|\|_{i=1}^{k} M_i$	$O(M^k)$	$O(M^k)$	M^k
$\sup \mathcal{C}(E_i \|\| M_j, \overline{M_j})$	$O(M^2N^2)$	$O(MN)$	$\alpha MN(\alpha \leq 1)$
$P(M_i)$	$O(2^M)$	$O(2^M)$	$\lambda 2^M(\lambda \leq 1)$
Compt. M_i-observer	$O(M^3)$	$O(M)$	$\lambda M(\lambda \leq 1)$
Compt. OCC for $\overline{M_i}$	$O(M^2)$	$O(M)$	M

An algorithm that extends the observable event set of a natural projection to produce a natural observer is presented in [6]. That algorithm can also obtain an automaton model of the projected language using the natural observer; this model will have no more (in general, fewer) states than the input language [29]. An algorithm to modify a natural projection to be OCC can be adapted from the algorithm in Section 5.5 of [33].

We first estimate the complexity of monolithic supervision. The monolithic supervisor for the control problem in Section IV is obtained by Equation (4). Accordingly, we need to compute the synchronous product of all the plant components and the specifications. The complexity is

$$\text{Time: } O(M^{2n}N^6), \text{ Space: } O(M^nN^3), \text{ Final Result: } \alpha_m M^n N^3 \tag{18}$$

Here $\alpha_m \leq 1$ is a case-dependent coefficient determined by the transition functions of the plant and specification models.

Next we estimate the complexity of the architectural control approach proposed in Section IV. Suppose the component numbers in the three subsystems $N_i(i = 1, 2, 3)$ are $k_i := ||N_i||(i = 1, 2, 3)$, respectively.

At Step 1 we compute the decentralized supervisors K_i $(i = 1, 2, 3)$ by Equation (5). For that, we need to compute the subplants H_i $(i = 1, 2, 3)$. The computation for one subplant is

$$\text{Time and Space: } O(M^{k_i}), \text{Final Result: } M^{k_i}.$$

Then the computational complexities for decentralized supervisors $K_i(i = 1, 2, 3)$ are

$$\text{Time: } O(M^{2k_i}N^2), \text{ Space: } O(M^{k_i}N), \text{ Final Result: } \alpha_i M^{k_i} N \tag{19}$$

Similarly α_m and $\alpha_i \leq 1$ $(i = 1, 2, 3)$.

At Step 2, we organize the plant and three decentralized supervisors into subsystems. To this end we represent the interconnection relationships of the product system by a graph, for instance, a *control flow net* [6], and partition the graph. By graph theory the algorithms are polynomial in the number of nodes, which is also the number of plant components n. Compared with the time and space used for other steps, the complexity for Step 2 is negligible.

At Step 3 we verify using Theorem 3 whether the three decentralized supervisors are synchronously nonconflicting. The key to using this theorem is to find an event set Υ_0 such that the natural projections used in Theorem 3 have the

observer property. Since Theorem 4 also requires that the natural projections based on Υ_0 have the OCC property, we must ensure that Υ_0 meets this requirement as well.

The OCC property should be checked first, for which the computational complexity is

$$\text{Time: } O(nM^2),\ \text{Space: } O(M),\ \text{Final Result: } M \tag{20}$$

Then the event set that guarantees the OCC property should be enlarged to achieve the observer property. Based on Table I, the computations take

$$\text{Time: } O(\alpha_i^4 M^{4k_i} N^4) = O(M^{4k_i} N^4),\ \text{Space: } O(M^{k_i} N)$$

Meanwhile, the algorithm for achieving the observer property will obtain the automaton models of $Q_i(K_i)$ $(i = 1, 2)$. The two models have state sizes

$$\text{Final Result: } \lambda_i \alpha_i M^{k_i} N,\ i = 1, 2. \tag{21}$$

Here $\lambda_i \leq 1$ $(i = 1, 2)$, because when the natural projection has the observer property, the state size of the projection model is never greater than that of the original model.

We then compute the synchronous product of $Q_1(K_1), Q_2(K_2), K_3$ and check if the result is nonblocking; this requires

$$\text{Time and Space: } O(\lambda_1 \lambda_2 \alpha_1 \alpha_2 M^{k_1+k_2+k_3} N^3) = O(M^n N^3),$$

$$\text{Final Result: } \lambda_1 \lambda_2 \alpha_1 \alpha_2 \alpha_3 M^n N^3$$

where $\alpha_3 \leq 1$ is a coefficient determined by the transition functions of the three languages.

If the result fails to be nonblocking, we proceed to step 4 to design the coordinator K_C by Equation (8).

$$\text{Time: } O(\lambda_1^2 \lambda_2^2 \alpha_1^2 \alpha_2^2 \alpha_3^2 M^{2n} N^6) = O(M^{2n} N^6),$$

$$\text{Space: } O(\lambda_1 \lambda_2 \alpha_1 \alpha_2 \alpha_3 M^n N^3) = O(M^n N^3),$$

$$\text{Final Result: } \lambda_1 \lambda_2 \alpha_1 \alpha_2 \alpha_3 \alpha_4 M^n N^3 \tag{22}$$

where $\alpha_4 \leq 1$ is determined by the algorithm computing the supremal controllable sublanguage.

The total time complexity of the architectural control approach is then

$$O[\sum_{i=1}^{3} (M^{k_i} + M^{2k_i} N^2) + M^{4k_1} N^4 + M^{4k_2} N^4 + nM^2$$

$$+ M^n N^3 + M^{2n} N^6] = O(M^{2n} N^6)$$

Since space is reusable, the space complexity of the whole process is determined by the most costly step, i.e.,

the last. The space complexity is then $O(M^n N^3)$.

On comparing the time and space complexities of the architectural control approach with those of the monolithic approach given in Equation (18), we conclude as follows.

1) The complexity bounds of our design approach are still exponential in the number of plant components and control specifications. Hence the approach will in general still be afflicted by state explosion.
2) The complexity is reduced by the coefficients α_i and λ_i. In favorable cases, the product of a sequence of these coefficients might be small, significantly reducing the computational workload.
3) Because the coefficients α_i are determined by the transition functions of the plant and control specifications, we have no way to decrease them. We can, however, affect the value of each λ_i and the total number of them through our partition of the system.
4) To increase the total number of coefficients λ, we should divide the given system into more subsystems. In the model abstraction of these subsystems, each one will contribute a coefficient λ.
5) To decrease the value of each coefficient λ, the observable event sets of the natural projections should be small.
6) The two options above are mutually countervailing. If a partition results in too many subsystems, many control specifications might interact with the subsystems. Each subsystem must then have a larger observable event set. Conversely, if the system is organized into a small number of subsystems, one subsystem must contain more control specifications and larger computational effort is consumed within each subsystem. A judicious trade-off must be chosen between them.

VI. Example: AGV System

We apply the proposed approach to the decentralized supervisory control of a system of automatic guided vehicles (AGVs) serving a manufacturing work cell. The example is from [9], as augmented in [33]. In [4] we gave a solution to this example, remarking that we had obtained a simpler solution with less computation on combining the architectural control approach with *control flow decomposition*. This section elaborates the better solution.

The system consists of two input stations IPS1 and IPS2 for parts of types 1, 2; three work stations WS1, WS2, WS3; one completed parts station CPS; and five AGVs (A1, ..., A5). The AGVs travel around fixed circular routes, on which they are loaded and unloaded by stations and machines, as shown in Fig. 7. A detailed description can be found in [33], Section 4.7.

Corresponding to the five AGVs, the plant model consists of five automata $\mathbf{A}_i$ with alphabets Σ_i $(i = 1, \ldots, 5)$. There are totally eight automaton models for the control specifications. Four ($\mathbf{Z}_i, i = 1, \ldots, 4$) stand for the zone restrictions, three ($\mathbf{WS}_i, i = 1, \ldots, 3$) for the operational constraints in work stations, and one (**IPS**) for the restriction about the common loading area between IPS1 and IPS2. Events and state transition diagrams of the automata can be found in [33], Section 4.7.

The *control-flow net* of the AGV example is shown in Fig. 8(a). Note that the original definition of specification WS1 in [33] does not meet the requirements of a *buffer* defined by Definition 3 in [5]. WS1 can, however,

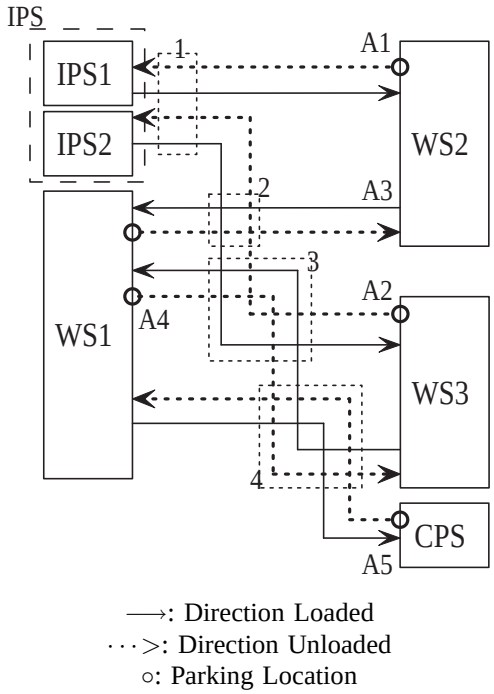


Fig. 7. An AGV System

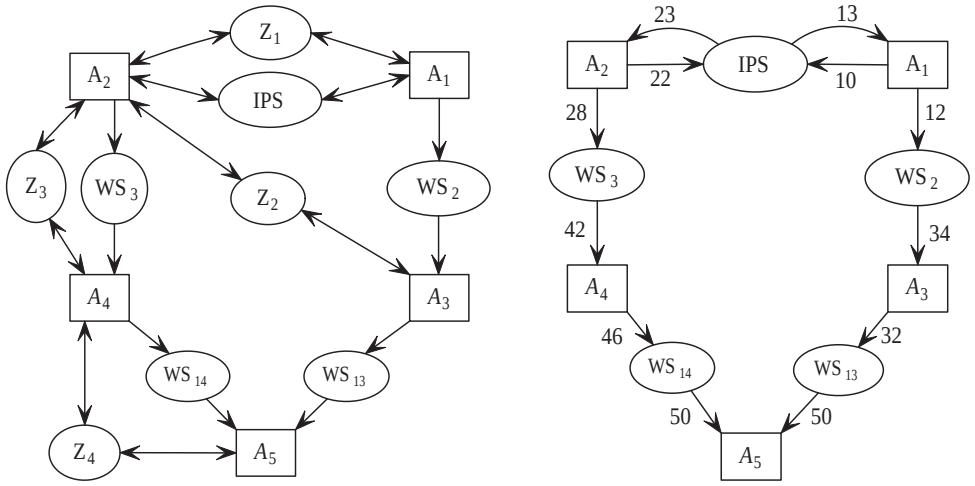
be replaced by two automata that both satisfy the buffer definition, that is, $\mathbf{WS1} = \mathbf{WS13}||\mathbf{WS14}$. The state transition diagrams of the three automata are shown in Fig. 9.

As prescribed in Section IV, we first design the decentralized supervisors of the eight specifications, which can be found in [33], Section 4.7. The state sizes of the nine decentralized supervisors are listed in Table II.

TABLE II
DECENTRALIZED SUPERVISORS

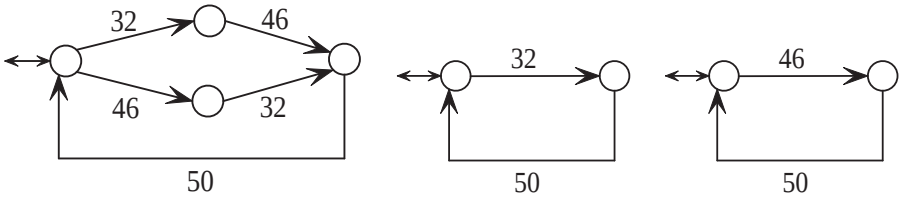
Spec	Sup (State#)	Spec	Sup (State#)
$\mathbf{Z}_1$	$\mathbf{ZR}_1\,(2)$	$\mathbf{Z}_2$	$\mathbf{ZR}_2\,(2)$
$\mathbf{Z}_3$	$\mathbf{ZR}_3\,(2)$	$\mathbf{Z}_4$	$\mathbf{ZR}_4\,(2)$
$\mathbf{WS}_{13}$	$\mathbf{WR}_{13}\,(2)$	$\mathbf{WS}_{14}$	$\mathbf{WR}_{14}\,(2)$
$\mathbf{WS}_2$	$\mathbf{WR}_2\,(2)$	$\mathbf{WS}_3$	$\mathbf{WR}_3\,(2)$
IPS	$\mathbf{IPR}\,(2)$		

While the synchronous product of the first eight decentralized supervisors for control specifications $\mathbf{Z}_i$ $(i = 1, \ldots, 4)$ and $\mathbf{WS}_j$ $(j = 1, 2, 3)$ is nonblocking, conflict arises when we introduce the supervisor for **IPS**. Hence we must design a coordinator to resolve the conflict. The coordinator obtained in [33] was a natural projection of the monolithic supervisor (4406 states) for the full AGV system. After projection of irrelevant events, the coordinator for this system was finally an automaton with 29 states. This approach, however, required the (complex) computation of the monolithic supervisor. In the following, we show how this can be avoided.



(a) Before Reduction (b) After Reduction

Fig. 8. Control-Flow Nets of AGV



(a) WS1 (b) WS13 (c) WS14

Fig. 9. Redefinition of WS1

By Proposition 3 in [5], the decentralized supervisors for the four zones, Z1 to Z4, are *irrelevant* to the nonblocking property. In other words, if we find the nonblocking supervisor of the remaining system, it must be synchronously nonconflicting with $\mathbf{ZR}_1$ to $\mathbf{ZR}_4$. We therefore ignore the four decentralized supervisors in the coordinator design and simplify the control-flow net in Fig. 8(a) to the net in Fig. 8(b). The structure in the simplified net is very clear. There are two processing paths: A1, WS2, A3, WS13, A5 on the right and A2, WS3, A4, WS14, A5 on the left. The two processing paths share one output machine A5 and their input machines share the server resource IPS. Hence the control-flow net of the AGV example has the parallel connection identified in [5].

The right-hand path, A1, WS2, A3, WS13, A5, determines a subsystem $\mathbf{Sub}_1$ with 140 states. The left-hand path, A2, WS3, A4, WS14, A5, determines subsystem $\mathbf{Sub}_2$ with 330 states.

To apply Corollary 3, we find the events shared between the two subsystems. Let Υ_1, Υ_2 be the alphabets of $\mathbf{Sub}_1, \mathbf{Sub}_2$, respectively. Evidently they share vehicle A5 at the bottom, i.e., $\Upsilon_1 \cap \Upsilon_2 = \Sigma_5$. At the top, vehicles

A1 and A2 are connected via the decentralized supervisor of specification **IPS**. The state transition diagram of the decentralized supervisor is shown in Fig. 10. Let the alphabet of the supervisor in Fig. 10 be $\Upsilon_3' := \{11, 13, 21, 23\}$. According to Corollary 3, the observable event set Υ_0 must be such that $\Upsilon_0 \supseteq \Upsilon_3' \cup (\Upsilon_1 \cap \Upsilon_2)$.

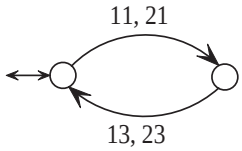


Fig. 10. Decentralized Supervisor of **IPS**

Using the algorithm in [32] or [6], we determine that $\Upsilon_0 = \Upsilon_3' \cup (\Upsilon_1 \cap \Upsilon_2)$; furthermore the natural projection $P_0 : \Sigma_0^* \to \Upsilon_0'$ enjoys the $L_m(\mathbf{Sub}_1), L_m(\mathbf{Sub}_2)$-observer and OCC properties.

Projecting the two subsystems to event set Υ_0, we get their abstractions,

$$\mathbf{Int}_1 := P_0(\mathbf{Sub}_1) \text{ (30 states)}$$

$$\mathbf{Int}_2 := P_0(\mathbf{Sub}_2) \text{ (30 states)}$$

The synchronous product of the two abstractions and the decentralized supervisor in Fig.10 is a *blocking* automaton.

$$\mathbf{IntIP} := \mathbf{Int}_1 || \mathbf{Int}_2 || \mathbf{IPR} \text{ (171 states)}$$

Using Corollary 3, we can find a coordinator for $\mathbf{IntIP}$, $\mathbf{CR}$ with 7 states, as in Fig. 11. The most costly computation in this process is for $\mathbf{Sub}_2$, which has only 330 states.

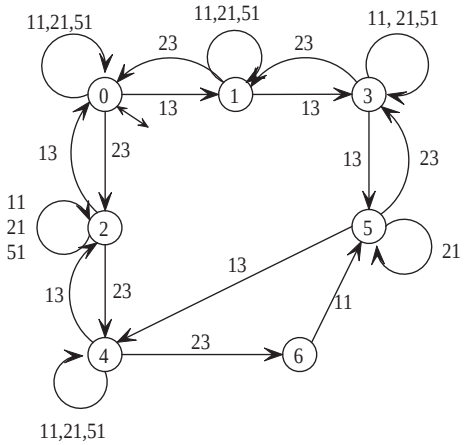


Fig. 11. Coordinator $\mathbf{CR}$

Not only does the new coordinator have smaller state size, but it also reveals the principle for eliminating blocking. Event 11 must be disabled at state 5 where event 13 occurs three more times than event 23, namely subsystem

$\mathbf{Sub}_1$ contains three more parts than $\mathbf{Sub}_2$. Dually, event 21 must be disabled at state 6 where event 23 occurs three more times than event 13, namely subsystem $\mathbf{Sub}_2$ contains three more parts than $\mathbf{Sub}_1$.

We provide a more transparent summary of the control actions of the coordinator. Deadlock arises in the AGV system when the vehicles and the work stations in one path in Fig.8(b) are all full, whereas the vehicles and the work stations in the other path are all empty. For illustration, suppose that WS1, WS2, A1 and A3 are full of type 1 parts, and A1 stops at the input station IPS1, while WS1, WS3, A2 and A5 do not carry any type 2 part. In this scenario, work station WS1 cannot produce any complete part, because of the lack of a type 2 part. Moreover, the type 2 part needed by WS1 cannot enter the system, because A2 cannot reach the input station IPS2 owing to specification **IPS**. Similarly, deadlock arises when WS1, WS3, A2 and A5 are full of type 2 parts, and A2 stops at the input station IPS2, while WS1, WS2, A1 and A3 do not carry any type 1 part. The control action of the coordinator **CR** prevents the occurrence of both scenarios.

VII. Conclusions

Although the synthesis of an optimal and nonblocking supervisor generally demands exponential time, we may often avoid the worst case and design the supervisor using modularity and model abstraction. A networked discrete-event system should, if possible, be divided and organized into subsystems based on the dependency relationships among plant components and control specifications. This decomposition will impose a hierarchical structure on the networked system. We independently compute the supervisors of the low level subsystems without regard to their mutual conflict. Subsequently we design coordinators for these controlled subsystems by high level supervision of them. To reduce computational complexity, we compute the high level coordinators based only on abstracted models of the controlled subsystems. Effective and consistent model abstraction is accomplished through natural projections with the observer and OCC properties.

The contribution of this paper is a practical supervisory control design approach that is capable of achieving maximally permissive and nonblocking control with decentralized supervisors. Thanks to distributed computation and information hiding, the architectural approach to control design is not only computationally efficient, but can also produce intuitively understandable supervisory controllers.

References

[1] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer, Boston, MA, 1999.

[2] M.H. de Queiroz and J.E.R. Cury. Modular control of composed systems. In *Proceedings of the 2000 American Control Conference*, volume 6, pages 4051–4055, Chicago, IL, June 2000.

[3] L. Feng. *Computationally Efficient Supervisor Design in Discrete-Event Systems*. PhD thesis, Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, 2007. In preparation.

[4] L. Feng and W.M. Wonham. Computationally efficient supervisor design: Abstraction and modularity. In S. Lafortune, F. Lin, and D. Tilbury, editors, *Proceedings of the 8th International Workshop on Discrete Event Systems*, pages 3–8, Ann Arbor, MI, July 2006.

[5] L. Feng and W.M. Wonham. Computationally efficient supervisor design: Control flow decomposition. In S. Lafortune, F. Lin, and D. Tilbury, editors, *Proceedings of the 8th International Workshop on Discrete Event Systems*, pages 9–14, Ann Arbor, MI, July 2006.

[6] L. Feng and W.M. Wonham. On the computation of natural observers in discrete-event systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 428–433, San Diego, CA, December 2006.

[7] P. Gohari and W. M. Wonham. On the complexity of supervisory control design in the RW framework. *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics, Special Issue on DES*, 30(5):643–652, 2000.

[8] R. Hill and D. Tilbury. Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction. In S. Lafortune, F. Lin, and D. Tilbury, editors, *Proceedings of the 8th International Workshop on Discrete Event Systems*, pages 399–406, Ann Arbor, MI, July 2006.

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

[10] J. Komenda and J. van Schuppen. Optimal solutions of modular supervisory control problems with indecomposable specification languages. In S. Lafortune, F. Lin, and D. Tilbury, editors, *Proceedings of the 8th International Workshop on Discrete Event Systems*, pages 143–148, Ann Arbor, MI, July 2006.

[11] R.J. Leduc, B.A. Brandin, M. Lawford, and W.M. Wonham. Hierarchical interface-based supervisory control-part I: Serial case. *IEEE Transactions on Automatic Control*, 50(9):1322–1335, 2005.

[12] R.J. Leduc, M. Lawford, and W.M. Wonham. Hierarchical interface-based supervisory control-part II: Parallel case. *IEEE Transactions on Automatic Control*, 50(9):1336–1348, 2005.

[13] S.H. Lee and K.C. Wong. Structural decentralized control of concurrent discrete-event systems. *European Journal of Control*, 8(5):477–491, 2002.

[14] F. Lin and W.M. Wonham. Decentralized supervisory control of discrete-event systems. *Information Sciences*, 44(3):199–224, 1988.

[15] C. Ma and W.M. Wonham. Nonblocking supervisory control of state tree structures. *IEEE Transactions on Automatic Control*, 51(5):782–793, 2006.

[16] S. Mac Lane and G. Birkhoff. *Algebra*. Macmillan Publishing Co., second edition, 1979.

[17] R. Milner. *Communication and Concurrency*. Prentice Hall International, New York, NY, 1989.

[18] K. Rudie. *Software for the Control of Discrete Event Systems: A Complexity Study*. M.A.Sc thesis, Department of Electrical and Computer Engineering, University of Toronto, 1988.

[19] K. Schmidt, H. Marchand, and B. Gaudin. Modular and decentralized supervisory control of concurrent discrete event systems using reduced system models. In S. Lafortune, F. Lin, and D. Tilbury, editors, *Proceedings of the 8th International Workshop on Discrete Event Systems*, pages 149–154, July 2006.

[20] K. Schmidt, T. Moor, and S. Perk. A hierarchical architecture for nonblocking control of decentralized discrete event systems. In *Proceedings of the 13th Mediterranean Conference on Control and Automation*, pages 902–907, Limassol, Cyprus, June 27-29 2005.

[21] K. Schmidt, J. Reger, and T. Moor. Hierarchical control for structural decentralized DES. In *Proceedings of the 7th International Workshop on Discrete Event Systems*, pages 289–294, Reims, France, September 2004.

[22] M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.

[23] R. Song and R.J. Leduc. Symbolic synthesis and verification of hierarchical interface-based supervisory control. In S. Lafortune, F. Lin, and D. Tilbury, editors, *Proceedings of the 8th International Workshop on Discrete-Event Systems*, pages 419–426, Ann Arbor, MI, July 2006.

[24] R. Su. *Distributed Diagnosis for Discrete-Event Systems*. PhD thesis, Department of Electrical and Computer Engineering, University of Toronto, June 2004.

[25] R. Su and W.M. Wonham. Supervisor reduction for discrete-event systems. *Discrete Event Dynamic Systems: Theory and Application*, 14(1):31–53, 2004.

[26] A.F. Vaz and W.M. Wonham. On supervisor reduction in discrete-event systems. *International Journal of Control*, 44(2):475–491, 1986.

[27] Y. Willner and M. Heymann. On supervisory control of concurrent discrete-event systems. *International Journal of Control*, 54(5):1143–1169, 1991.

[28] K.C. Wong. *Discrete-Event Control Architecture: An Algebraic Approach*. PhD thesis, Department of Electrical and Computer Engineering, University of Toronto, 1994.

[29] K.C. Wong. On the complexity of projections of discrete-event systems. In *Proceedings of the 4th International Workshop on Discrete Event Systems*, pages 201–206, Cagliari, Italy, August 1998.

[30] K.C. Wong and W.M. Wonham. Hierarchical control of discrete-event systems. *Discrete-Event Dynamic Systems: Theory and Applications*, 6(3):241–273, July 1996.

[31] K.C. Wong and W.M. Wonham. Modular control and coordination of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Application*, 8(3):247–297, 1998.

[32] K.C. Wong and W.M. Wonham. On the computation of observers in discrete-event systems. *Discrete Event Dynamic Systems: Theory and Application*, 14(1):55–107, 2004.

[33] W.M. Wonham. *Supervisory Control of Discrete-Event Systems* and *Design Software: XPTCT*. Department of Electrical and Computer Engineering, University of Toronto, 2002-2006, http://www.control.toronto.edu/DES, Updated July 1 2006.

[34] W.M. Wonham and P.J. Ramadge. Modular supervisory control of discrete event systems. *Mathematics of Control, Signal and Systems*, 1(1):13–30, 1988.

[35] H. Zhong and W.M. Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Transactions on Automatic Control*, 35(10):1125–1134, 1990.

APPENDIX
SELECTED PROOFS

Proof of Lemma 1:

($\Longrightarrow$) Since $L_1 \| L_2 \neq \emptyset$, there exists some string $s \in L_1 \| L_2$. Then $P_i(s) \in L_i$ $(i = 1, 2)$. Applying natural projections Q_1 and Q_2 respectively to both equations, $Q_1 P_1(s) \in Q_1 L_1$ and $Q_2 P_2(s) \in Q_2 L_2$. Because of Statement 3 of Proposition 2, $Q_i P_i(s) = P_0(s)$ $(i = 1, 2)$. Hence the above equations imply $P_0(s) \in Q_1 L_1$ and $P_0(s) \in Q_2 L_2$. Therefore $P_0(s) \in Q_1 L_1 \cap Q_2 L_2$ and $Q_1 L_1 \cap Q_2 L_2 \neq \emptyset$.

($\Longleftarrow$) Since $Q_1 L_1 \cap Q_2 L_2 \neq \emptyset$, we can select a string $t \in Q_1 L_1 \cap Q_2 L_2$, of form

$$t = \sigma_0 \, \sigma_1 \cdots \sigma_n, \; n \geq 0, \sigma_0 = \epsilon, \sigma_i \in \Sigma_0, i \in \mathbf{n}$$

Note that if $n = 0$ string t is just the empty string ϵ. Hence there must exist strings $s_1 \in L_1$ and $s_2 \in L_2$ such that $Q_1(s_1) = t = Q_2(s_2)$, having the form

$$\begin{aligned} s_1 &= u_0 \, \sigma_1 \, u_1 \cdots u_{n-1} \, \sigma_n \, u_n, \; u_i \in (\Sigma_1 - \Sigma_0)^*, \; i \in \{0, \ldots, n\} \\ s_2 &= v_0 \, \sigma_1 \, v_1 \cdots v_{n-1} \, \sigma_n \, v_n, \; v_i \in (\Sigma_2 - \Sigma_0)^*, \; i \in \{0, \ldots, n\}. \end{aligned}$$

From s_1, s_2 we can construct a new string

$$\omega := u_0 \, v_0 \, \sigma_1 \, u_1 \, v_1 \cdots u_{n-1} \, v_{n-1} \, \sigma_n \, u_n \, v_n.$$

Evidently, $P_1(\omega) = u_0 \, \sigma_1 \, u_1 \cdots u_{n-1} \, \sigma_n \, u_n = s_1 \in L_1$ and $P_2(\omega) = v_0 \, \sigma_1 \, v_1 \cdots v_{n-1} \, \sigma_n \, v_n = s_2 \in L_2$. Therefore $\omega \in L_1 \| L_2$ and hence $L_1 \| L_2 \neq \emptyset$. ■

Proof of Proposition 3

The proof needs the following lemma.

Lemma 2: Let $L_i \subseteq \Sigma_i^*$ $(i = 1, 2)$. Let $\Sigma_0 := \Sigma_1 \cap \Sigma_2$. Define natural projections as in Fig. 1 (a). Then for $i, j = 1, 2$ and $i \neq j$, $P_i(L_1 \| L_2) = L_i \cap Q_i^{-1} Q_j(L_j)$.

Proof: ($\subseteq$)

$$P_i(L_1\|L_2) = P_i(P_i^{-1}L_i \cap P_j^{-1}L_j) \subseteq P_iP_i^{-1}(L_i) \cap P_iP_j^{-1}(L_j),\ i,j = 1,2; i \neq j.$$

By Statement 6 of Proposition 2, we then have $P_iP_i^{-1}(L_i) \cap P_iP_j^{-1}(L_j) = L_i \cap Q_i^{-1}Q_j(L_j)$.

($\supseteq$) Let $s \in L_i \cap Q_i^{-1}Q_j(L_j)$. Thus $s \in L_i$, $s \in Q_i^{-1}Q_j(L_j)$, and

$$Q_i(s) \in Q_j(L_j). \tag{23}$$

Define the language $K = \{s\}\|L_j = P_i^{-1}\{s\} \cap P_j^{-1}(L_j)$. Since $s \in L_i, K \subseteq L_1\|L_2$. Based on Lemma 1 and Equation (23), we know $K \neq \emptyset$. For any string $t \in K$, we have $t \in L_1\|L_2$ and $t \in P_i^{-1}\{s\}$. Hence $P_i(t) = s$. Therefore $s \in P_i(L_1\|L_2)$ as required. ■

Returning to the proof of Proposition 3, we bring in the natural projections defined in Fig. 4. Let $R_i := P_i|\Sigma_0^*$ and $Q_i := P_0|\Sigma_i^*$ $(i = 1,2)$.

It is straightforward to show that $P_0(L_1\|L_2) \subseteq Q_1(L_1)\|Q_2(L_2)$. Actually, by Propositions 1 and 2,

$$\begin{aligned} P_0(L_1\|L_2) &= P_0(P_1^{-1}(L_1) \cap P_2^{-1}(L_2)) \subseteq P_0P_1^{-1}(L_1) \cap P_0P_2^{-1}(L_2) \\ &= R_1^{-1}Q_1(L_1) \cap R_2^{-1}Q_2(L_2) = Q_1(L_1)\|Q_2(L_2). \end{aligned}$$

Conversely let string $s \in Q_1(L_1)\|Q_2(L_2) \subseteq \Sigma_0^*$. Then $R_i(s) \in Q_i(L_i)$ $(i = 1,2)$. We define language K as

$$K := P_0^{-1}\{s\} \cap P_1^{-1}(L_1) \cap P_2^{-1}(L_2) = \{s\}\|L_1\|L_2$$

Since synchronous product is associative and commutative [33], $K = (\{s\}\|L_1)\|(\{s\}\|L_2)$. Let $K_i = \{s\}\|L_i$ $(i = 1,2)$. Because $s \in \Sigma_0^*$, then $K_i \subseteq (\Sigma_0 \cup \Sigma_i)^*$ $(i = 1,2)$. The joint event set of K_1 and K_2 is $(\Sigma_0 \cup \Sigma_1) \cap (\Sigma_0 \cup \Sigma_2) = \Sigma_0$. To prove that $K \neq \emptyset$, by Lemma 1 we need to calculate $P_0(K_1)$ and $P_0(K_2)$. Since $\{s\} \subseteq \Sigma_0^*$, we compute $P_0(K_i)$ $(i = 1,2)$ by Lemma 2:

$$P_0(K_i) = P_0(\{s\}\|L_i) = \{s\} \cap R_i^{-1}Q_i(L_i),\ i = 1,2. \tag{24}$$

According to the assumption that $s \in Q_1(L_1)\|Q_2(L_2)$, $R_i(s) \in Q_i(L_i)$ and hence $s \in R_i^{-1}Q_i(L_i)$ $(i = 1,2)$. Therefore Equation (24) yields $P_0(K_i) = \{s\}$ $(i = 1,2)$. This result means $P_0(K_1) \cap P_0(K_2) = \{s\} \neq \emptyset$. According to Lemma 1, $K = K_1\|K_2 \neq \emptyset$.

By the definition of language K, we know that $K \subseteq P_0^{-1}\{s\}$ and $K \subseteq L_1\|L_2$. Let string $t \in K$. Then $P_0(t) = s$ and $t \in L_1\|L_2$. Consequently $s \in P_0(L_1\|L_2)$ and $Q_1(L_1)\|Q_2(L_2) \subseteq P_0(L_1\|L_2)$. ■

Proof of Proposition 5:

According to Fig. 4, let $Q_i := P_0|\Sigma_i^*$ and $R_i := P_i|\Sigma_0^*$ $(i = 1,2)$. Let $t \in P_0(L_1\|L_2)$, $s \in \overline{L_1\|L_2}$, and $P_0(s) \leq t$. To justify the statement, we must find a string $w \in (\Sigma_1 \cup \Sigma_2)^*$ such that $sw \in L_1\|L_2$ and $P_0(sw) = t$.

Since it is always true that $\overline{L_1 \| L_2} \subseteq \overline{L_1} \| \overline{L_2}$, $s \in \overline{L_1} \| \overline{L_2}$. Consequently, for $i = 1, 2$,

$$R_i(t) \in R_i P_0(L_1 \| L_2) = Q_i P_i(L_1 \| L_2) \subseteq Q_i(L_i)$$

and $P_i(s) \in \overline{L_i}$. Because $P_0(s) \leq t$, we can apply R_i on both sides, to get

$$R_i P_0(s) = Q_i P_i(s) \leq R_i(t).$$

Now that $R_i(t) \in Q_i(L_i)$, $P_i(s) \in \overline{L_i}$, we can conclude by the hypothesis of the proposition that Q_i is an L_i-observer, so

$$(\exists u_i \in \Sigma_i^*) P_i(s) u_i \in L_i \tag{25}$$

and

$$Q_i[(P_i(s)u_i] = R_i(t),\ i = 1, 2. \tag{26}$$

Apply P_j $(j = 1, 2; j \neq i)$ on both sides of Equation (26) to get

$$\begin{aligned} P_j[Q_i((P_i(s)u_i)] &= P_j R_i(t) \\ P_j[Q_i P_i(s) Q_i(u_i)] &= T(t) \end{aligned}$$

According to Corollary 1, $P_j Q_i P_i(s) = P_1 P_2(s)$ and $P_j Q_i(u_i) = P_j(u_i)$. Hence the preceding equation shows that

$$P_1 P_2(s) P_1(u_2) = T(t) = P_1 P_2(s) P_2(u_1) \Rightarrow P_1(u_2) = P_2(u_1).$$

Recall that $u_i \in \Sigma_i^*$. Then we conclude by Lemma 1 that $\{u_1\} \| \{u_2\} \neq \emptyset$. Suppose $v \in \Sigma_0^*$ with

$$P_0(s) v = t. \tag{27}$$

Let $X := (\{u_1\} \| \{u_2\}) \| \{v\}$. We already know that $\{u_1\} \| \{u_2\}$ is nonempty. Hence we should further prove that $v \in P_0(\{u_1\} \| \{u_2\})$ to establish $X \neq \emptyset$. Applying R_i $(i = 1, 2)$ on both sides of Equation (27), we get $R_i P_0(s) R_i(v) = R_i(t)$ and hence $Q_i P_i(s) R_i(v) = R_i(t)$ $(i = 1, 2)$. Comparing this result with Equation (26), we obtain $R_i(v) = Q_i(u_i)$ $(i = 1, 2)$. Therefore $v \in Q_1\{u_1\} \| Q_2\{u_2\}$. By Proposition 3, $v \in P_0(\{u_1\} \| \{u_2\})$. Then by Lemma 1, $X = (\{u_1\} \| \{u_2\}) \| \{v\} \neq \emptyset$.

Taking any string from set X, say $w \in X$, we see immediately that $P_0(w) = v$ and $P_i(w) = u_i$ $(i = 1, 2)$. Insert these equalities into Equations (25), (27), to get

$$P_i(sw) \in L_i,\ i = 1, 2,$$

and $P_0(sw) = P_0(s)P_0(w) = P_0(s)v = t$. Therefore $sw \in L_1 \| L_2$ and $P_0(sw) = t$. Thus P_0 is an $L_1 \| L_2$-observer. ■

Proof of Proposition 7:

Let $L_0' := Q_1(L_1)||Q_2(L_2)||L_0$. Then we have

$$\overline{L_0'} = \overline{Q_1(L_1)||Q_2(L_2)||L_0} = Q_1(\overline{L_1})||Q_2(\overline{L_2})||\overline{L_0}.$$

Consequently,

$$L_1||L_2||L_0' = (L_1||Q_1(L_1))||(L_2||Q_2(L_2))||L_0 = L_1||L_2||L_0.$$

and similarly, $\overline{L_1}||\overline{L_2}||\overline{L_0'} = \overline{L_1}||\overline{L_2}||\overline{L_0}$. The proposition is now reduced to showing that $\overline{L_1||L_2||L_0'} = \overline{L_1}||\overline{L_2}||\overline{L_0'}$.

We first show that L_1 and L_2 are each synchronously nonconflicting with L_0'. For $i = 1, 2$, since $Q_i(L_i)||L_0' = L_0'$ and $Q_i(\overline{L_i})||\overline{L_0'} = \overline{L_0'}$, we can claim that $Q_i(L_i)$ and L_0' are synchronously nonconflicting, i.e., $\overline{Q_i(L_i)||L_0'} = Q_i(\overline{L_i})||\overline{L_0'}$. According to the assumption that Q_i is an L_i-observer, Proposition 1 ensures that L_i and L_0' are synchronously nonconflicting, i.e.,

$$\overline{L_i||L_0'} = \overline{L_i}||\overline{L_0'},\ i = 1, 2. \tag{28}$$

Let $J_i := L_i \| L_0' \subseteq (\Sigma_i \cup \Sigma_0)^*$ $(i = 1, 2)$. By Proposition 5, P_0 is then also a J_i-observer. Because synchronous product is associative and commutative,

$$L_1 \| L_2 \| L_0' = (L_1 \| L_0') \| (L_2 \| L_0') = J_1 \| J_2.$$

Next we show that J_1 and J_2 are also synchronously nonconflicting. Since $J_i \subseteq (\Sigma_i \cup \Sigma_0)^*$ $(i = 1, 2)$, the joint event set between J_1 and J_2 is then Σ_0. By Proposition 3,

$$P_0 J_i = P_0(L_i \| L_0') = Q_i(L_i) \| L_0' = L_0',\ i = 1, 2.$$

Hence $P_0 J_1 \cap P_0 J_2 = L_0'$. Similarly by Equation (28),

$$P_0 \overline{J_i} = P_0(\overline{L_i \| L_0'}) = P_0(\overline{L_i} \| \overline{L_0'}) = Q_i(\overline{L_i}) \| \overline{L_0'} = \overline{L_0'},\ i = 1, 2.$$

Therefore $\overline{P_0 J_1 \cap P_0 J_2} = P_0\overline{J_1} \cap P_0\overline{J_2}$. The equation demonstrates that $P_0 J_1$ and $P_0 J_2$ are nonconflicting. Since P_0 is a J_1 and J_2-observer, we conclude that J_1 and J_2 are synchronously nonconflicting, namely, $\overline{J_1 \| J_2} = \overline{J_1} \| \overline{J_2}$. Using the definition of J_i $(i = 1, 2)$ in the above equation and applying Equation (28), we get $\overline{L_1 \| L_2 \| L_0'} = \overline{L_1} \| \overline{L_2} \| \overline{L_0'}$. ■

Proof of Theorem 2:

For this we introduce a new lemma.

Lemma 3: Let L and X be prefix closed languages over the alphabet Σ and let $X \subseteq L$. Suppose language $K \subseteq X$ is controllable with respect to $\Sigma_u \subseteq \Sigma$ and L. If the natural projection $P : \Sigma^* \to \Sigma_0^*$ is an X-observer and is OCC for L, then $P(K)$ is controllable with respect to $P(X)$ and $\Sigma_u \cap \Sigma_0$.

Proof: Let $t \in P(\overline{K})$, $\sigma \in \Sigma_u \cap \Sigma_0$ and $t\sigma \in P(X)$. There must exist a string $s \in \overline{K}$ with $t = P(s)$. We

select s so that

$$t = \epsilon \implies s = \epsilon \tag{29}$$

$$t \neq \epsilon \implies (\forall s' < s) P(s') < t. \tag{30}$$

Because P is an X-observer, there is a string $v \in \Sigma^*$ with $sv \in X$ and $P(sv) = P(s)P(v) = tP(v) = t\sigma$. Hence we can find a string $u \in (\Sigma - \Sigma_0)^*$ such that $v = u\sigma$.

Because $\sigma \in \Sigma_u \cap \Sigma_0$ and P is OCC for L, we know from Definition 3 that $u \in \Sigma_u^*$. Since K is controllable with respect to L, $su\sigma \in \overline{K}$ and $P(su\sigma) = t\sigma \in P(\overline{K})$.

The procedure above shows that

$$\overline{P(K)}(\Sigma_u \cap \Sigma_0) \cap P(X) \subseteq \overline{P(K)}.$$

This means that $P(K)$ is indeed controllable with respect to $P(X)$, as required. ■

The proof of Theorem 2 is as follows.

($\supseteq$) Let $K_0 := \sup \mathcal{C}_0[E \cap P(L_m), P(L)]$ and $K := \sup \mathcal{C}(E \| L_m, L)$. Since $\sup \mathcal{C}_0[E \cap P(L_m), P(L)] \subseteq E \cap P(L_m)$, there follows

$$\begin{aligned} K_0 \| L_m &\subseteq [E \cap P(L_m)] \| L_m = P^{-1}[E \cap P(L_m)] \cap L_m \\ &= P^{-1}(E) \cap P^{-1}P(L_m) \cap L_m = E \| L_m. \end{aligned} \tag{31}$$

Next we show that $K_0 \| L_m$ is controllable with respect to L. By its definition, $K_0 \subseteq P(L_m)$. Since we assume that P is an L_m-observer, Theorem 1 implies

$$\overline{K_0 \| L_m} = \overline{K_0} \| \overline{L_m} = \overline{K_0} \| L,$$

namely, K_0 and L_m are synchronously nonconflicting.

Because K_0 is controllable with respect to $P(L)$, Proposition 6 implies that $K_0 || L_m$ is controllable with respect to $P(L) || L = L$. Using Equation (31), we conclude

$$K_0 \| L_m = \sup \mathcal{C}_0[E \cap P(L_m), P(L)] \| L_m \subseteq \sup \mathcal{C}(E \| L_m, L) = K.$$

($\subseteq$) To prove that $K \subseteq K_0 \| L_m$, it suffices to prove that $P(K) \subseteq K_0$ and $K \subseteq L_m$. As the latter inclusion is evident, we need only prove the first statement. Since $K \subseteq E \| L_m$, Proposition 3 implies $P(K) \subseteq P(E \| L_m) = E \cap P(L_m)$.

Using Lemma 3 we can also show that $P(K)$ is controllable with respect to $P(L)$, because P is an L-observer and OCC for L. Now that $P(K) \in \mathcal{C}_0(E \cap P(L_m), P(L))$, there follows

$$P(K) \subseteq K_0 = \sup \mathcal{C}_0(E \cap P(L_m), P(L)).$$

■

Proof of Proposition 8

Lemma 4: Let L_i be two prefix-closed languages over alphabet Σ_i $(i = 1, 2)$ and let $X \subseteq L_1||L_2$ be a third prefix-closed language. Suppose language $K \subseteq X$ is controllable with respect to $\Sigma_u \subseteq \Sigma_1 \cup \Sigma_2$ and $L_1\|L_2$. Let the observable event set be $\Sigma_0 \supseteq \Sigma_1 \cap \Sigma_2$. Refer to the natural projections defined in Fig. 4. If P_0 is an X-observer and Q_i is OCC for L_i $(i = 1, 2)$, then $P_0(K)$ is controllable with respect to $\Sigma_u \cap \Sigma_0$ and $P_0(X)$.

Proof: Let $t \in P_0(\overline{K}), \sigma \in \Sigma_u \cap \Sigma_0$, and $t\sigma \in P_0(X)$. We will show that $t\sigma \in P_0(\overline{K})$. First, there exists a string $s \in \overline{K}$ such that $P_0(s) = t$ and s satisfies Equations (29), (30).

Since $t\sigma = P_0(s)\sigma \in P_0(X)$ and P_0 is an X-observer, there is a string $u \in (\Sigma_1 \cup \Sigma_2)^*$ such that $su \in X$ and $P_0(su) = P_0(s)\sigma$. Clearly $P_0(u) = \sigma$. Hence there is a string $v \in (\Sigma_1 \cup \Sigma_2 - \Sigma_0)^*$ with $u = v\sigma$ and $sv\sigma \in X$.

If $\sigma \in \Sigma_u \cap \Sigma_1 \cap \Sigma_2$, then $P_i(sv\sigma) = P_i(s)P_i(v)\sigma \in P_i(X)$ $(i = 1, 2)$. Because $X \subseteq L_1\|L_2$,

$$P_i(X) \subseteq P_i(L_1\|L_2) \subseteq L_i,\ i = 1, 2. \tag{32}$$

Consequently, $P_i(s)P_i(v)\sigma \in L_i$ $(i = 1, 2)$. Evidently $P_i(s) \in L_i, P_i(v) \in (\Sigma_i - \Sigma_0)^*$. Since Q_i is OCC for L_i and $\sigma \in \Sigma_u \cap \Sigma_0$, $P_i(v) \in (\Sigma_i \cap \Sigma_u)^*$ $(i = 1, 2)$. Thus we can infer that $v \in \Sigma_u^*$. Because $s \in \overline{K}$, $sv\sigma \in L_1\|L_2$, and K is controllable with respect to $L_1\|L_2$, we have $sv\sigma \in \overline{K}$ and $P_0(sv\sigma) = P_0(s)\sigma = t\sigma \in P_0(\overline{K})$, as required.

If $\sigma \in \Sigma_u \cap \Sigma_0 \cap \Sigma_i - \Sigma_j$ $(i \neq j)$, then, by Equation (32), $P_i(sv\sigma) = P_i(s)P_i(v)\sigma \in L_i$ $(i = 1$ or $2)$. Let $w := P_i(v) \in (\Sigma_i - \Sigma_0)^*$. Then

$$P_i(sw\sigma) = P_i(s)w\sigma \in L_i$$

$$P_j(sw\sigma) = P_j(s) \in L_j$$

These equations imply $sw\sigma \in L_1\|L_2$. Furthermore Q_i is OCC for L_i and $w \in (\Sigma_i - \Sigma_0)^*$. Hence we can infer that $w \in (\Sigma_i - \Sigma_0)^* \cap \Sigma_u^*$. Considering the assumption that K is controllable with respect to $L_1\|L_2$, we can then conclude that $sw\sigma \in \overline{K}$. Therefore $P_0(sw\sigma) = P_0(s)\sigma = t\sigma \in P_0(\overline{K})$. As we have considered all cases for σ, the proof of Lemma 4 is complete. ■

The proof of Proposition 8 is as follows.

($\supseteq$) Let $K := \sup\mathcal{C}(E\|L_m, L)$ and $K_0 := \sup\mathcal{C}_0[E\cap P_0(L_m), P_0(L)]$. From the definition of the local supervisor K_0, we have

$$K_0 \subseteq P_0(L_m) = P_0(M_1||M_2) = Q_1(M_1)||Q_2(M_2).$$

Applying prefix closure, we obtain $\overline{K_0} \subseteq Q_1(\overline{M_1})||Q_2(\overline{M_2})$. Therefore,

$$\overline{Q_1(M_1)||Q_2(M_2)||K_0} = \overline{K_0} = Q_1(\overline{M_1})||Q_2(\overline{M_2})||\overline{K_0}.$$

Because Q_i is an M_i-observer ($i = 1, 2$), Proposition 7 implies that

$$\overline{L_m||K_0} = \overline{M_1||M_2||K_0} = \overline{M_1}||\overline{M_2}||\overline{K_0} = L||\overline{K_0}.$$

Moreover, K_0 is controllable with respect to $P_0(L)$. Using Proposition 6, we conclude that $K_0\|L_m$ is controllable with respect to $P_0(L)\|L = L$.

Because $K_0 \subseteq E \cap P_0(L_m)$, we know

$$K_0\|L_m \subseteq (E \cap P_0(L_m))\|L_m$$

$$= P_0^{-1}(E) \cap P_0^{-1}P_0(L_m) \cap L_m = E\|L_m.$$

Since we already know that $K_0\|L_m$ is controllable with respect to L, $K_0\|L_m \subseteq K$.

($\subseteq$) While the condition that the Q_i are L_i-observers ($i = 1, 2$) implies that P_0 is an L_m-observer, the condition that Q_i are OCC for $\overline{M_i}$ ($i = 1, 2$) does not imply that P_0 is OCC for L. So we cannot directly use the result of Theorem 2.

To show that $K \subseteq K_0\|L_m$, we must show that $P_0(K) \subseteq K_0$ and $K \subseteq L_m$. Since $K \subseteq E\|L_m$, we know $K \subseteq L_m$ and

$$P_0(K) \subseteq P_0(E\|L_m) = E \cap P_0(L_m) \tag{33}$$

Recall that $L = \overline{M_1}\|\overline{M_2}$ and Q_i is an M_i-observer ($i = 1, 2$). Proposition 4 ensures that Q_i is also an $\overline{M_i}$-observer ($i = 1, 2$). By Proposition 5, we know that P_0 is an L-observer. Because K is controllable with respect to L, $P_0(K)$ is then controllable with respect to $P_0(L)$ by Lemma 4. Together with Equation (33), it implies that $P_0(K) \subseteq K_0$, which is the key statement for the proof of the second part. ■

Proof of Theorem 4:

From Equations (9) and (10),

$$K_1||K_2||K_3||K_C = K_1||K_2||K_3||C$$

$$\overline{K_1}||\overline{K_2}||\overline{K_3}||\overline{K_C} = \overline{K_1}||\overline{K_2}||\overline{K_3}||\overline{C}.$$

Furthermore, Equation (8) yields

$$K_C \subseteq Q_1(K_1)||Q_2(K_2)||K_3$$

$$\overline{K_C} \subseteq Q_1(\overline{K_1})||Q_2(\overline{K_2})||\overline{K_3}.$$

Hence $K_C \subseteq R^{-1}(K_3)$ and $\overline{K_C} \subseteq R^{-1}(\overline{K_3})$, where R is the natural projection $R : \Upsilon_0^* \to \Upsilon_3^*$. We immediately obtain the following two equations.

$$K_3||K_C = K_C \text{ and } \overline{K_3}||\overline{K_C} = \overline{K_C} \tag{34}$$

Finally we have

$$\begin{aligned} K_1||K_2||K_3||C &= K_1||K_2||K_C \\ \overline{K_1}||\overline{K_2}||\overline{K_3}||\overline{C} &= \overline{K_1}||\overline{K_2}||\overline{K_C} \end{aligned}$$

The proof of Equation (11) is now reduced to the proof of $\overline{K_1||K_2||K_C} = \overline{K_1}||\overline{K_2}||\overline{K_C}$, which will follow shortly from Proposition 7. The assumption already ensures that the Q_i are K_i-observers $(i = 1, 2)$, so we need only show

$$\overline{Q_1(K_1)||Q_2(K_2)||K_C} = Q_1(\overline{K_1})||Q_2(\overline{K_2})||\overline{K_C}. \tag{35}$$

Considering Equation (34), we have

$$Q_1(K_1)||Q_2(K_2)||K_C = Q_1(K_1)||Q_2(K_2)||K_3||K_C = K_C$$

$$Q_1(\overline{K_1})||Q_2(\overline{K_2})||\overline{K_C} = Q_1(\overline{K_1})||Q_2(\overline{K_2})||\overline{K_3}||\overline{K_C} = \overline{K_C}.$$

Thus Equation (35) must follow and the proof of Equation (11) is complete.

The second part of the proof is for Equation (12). Since we have already shown that $K_1||K_2||K_3||C = K_1||K_2||K_3||K_C$, Equation (12) is transformed to $K_1||K_2||K_3||K_C = \sup \mathcal{C}(E_1||E_2||E_3||L_m, L)$. The proof of this equation follows from Proposition 4.2.2 of [33]. ■