Computationally Efficient Supervisor Design: Abstraction and Modularity

Lei Feng, W.M. Wonham
Department of Electrical and Computer Engineering
University of Toronto, Toronto, Canada
fenglei, wonham@control.utoronto.ca

***Abstract*— A flexible modular and hierarchical structure is presented to reduce computational effort in designing optimal nonblocking supervisors for discrete-event systems (DES). The structure organizes the system into modular subsystems that embody internal interacting dependencies. Verification and coordination among modular subsystems are achieved through their model abstractions. Sufficient conditions are presented to guarantee that these coordinators and modular supervisors result in optimal nonblocking control. A medium-sized example demonstrates the computational advantage of our approach.**

I. Introduction

Since the nonblocking supervisory control problem in the RW framework [18] is NP-hard [4], [10], research is best directed to efficient solutions for various subclasses of discrete-event systems that enjoy special structure. Such structure enables us to exploit *modularity* [8], [11], [12], [14] and *model abstraction* [6], [7], [16] so as to circumvent computing global models.

This paper presents a flexible modular and hierarchical structure that has been found effective in reducing the computational complexity of control synthesis. The structure organizes the plant components and local modular supervisors into weakly dependent (ideally, uncoupled) subsystems according to their interacting dependencies. If conflict (blocking) arises within a subsystem, we design a coordinator of the modular supervisors within each subsystem, with relatively small effort provided each subsystem is much smaller than the original system.

To verify the nonconflicting relation among subsystems, criteria using structural properties [8], [14] rather than brute-force computation have been proposed. As these structures are not universal, computational verification and coordination [17] of the modular subsystems are often inevitable. To reduce computational complexity, we use proper abstractions of these subsystems instead of the original models. Viewing these model abstractions as high level plant components, and the related local modular supervisors as high level specifications, we may resolve blocking (if any) through control synthesis for the high level control problem.

Repetition of this process leads to a hierarchy of local modular supervisors and coordinators. The paper presents sufficient conditions to guarantee that these modules result in optimal nonblocking control, i.e., their synchronization is equivalent to the monolithic controller. Methods for selecting subsystems appropriately are discussed in the companion paper [3].

The paper exploits the same DES structure as that in [11], [12], but with the fundamental difference that we do not assume *a priori* that the modular subsystems are nonconflicting. Coordinating these modules to eliminate blocking is our primary objective.

The paper is organized as follows. Section II introduces fundamental properties of the *natural projection* and the *observer*. Section III presents a sufficient condition for optimal nonblocking local control with partial observation. A group of such local modular supervisors may be conflicting, hence Section IV introduces a systematic coordination scheme based on model abstraction. Section V demonstrates its efficiency through a realistic example.

II. Natural Observer

Given a system described by a language $L \subseteq \Sigma^*$, its *model abstraction* is the induced system described by the language $\theta(L)$, where θ is a *causal reporter map* [16], [18]. This paper will construct model abstractions using a *natural observer*, i.e., natural projection [1], [18] having the *observer* property [16]. Natural projection allows compositional computation. Consider a product system consisting of two components, whose languages are L_i over alphabet $\Sigma_i, i = 1, 2$. We first recall the definition of the *synchronous product* [18] $L_1||L_2$. Let $P_i : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_i^* (i = 1, 2)$ be the natural projections with inverse image functions $P_i^{-1} : Pwr(\Sigma_i^*) \to Pwr((\Sigma_1 \cup \Sigma_2)^*)$. Then

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

To obtain an abstraction of the product system, we can first compute its global behavior $L_1||L_2$ and then compute its projection. However, when the shared events of the two components are all observable, we can instead obtain it through abstractions of the two languages, thus avoiding computation of the global behavior. This structural property (Exercise 3.3.7 in [18]), summarized as Proposition 1, is central to our method.

Proposition 1: Let $L_i \subseteq \Sigma_i^*, i = 1, 2$; $\Sigma_0 \subseteq \Sigma := \Sigma_1 \cup \Sigma_2$; $P_0 : \Sigma^* \to \Sigma_0^*$; and $Q_j : \Sigma_j^* \to (\Sigma_j \cap \Sigma_0)^*, (j = 1, 2)$, with P_0 and the Q_j natural projections. If $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma_0$, then

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

The proposition is illustrated by the commutative diagram, Fig. 1. Here $\Sigma_{10} := \Sigma_1 \cap \Sigma_0$ and $\Sigma_{20} := \Sigma_2 \cap \Sigma_0$.

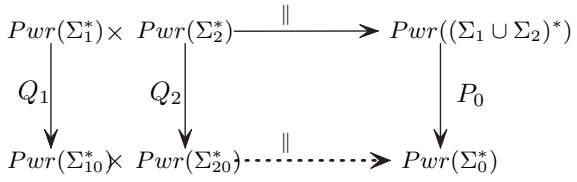


Fig. 1. Proposition 1

Natural projection may discard critical information about the original behavior, so the resulting abstraction may not be consistent with the original behavior with respect to nonblocking. The following observer property is required to ensure that whenever the observed string, say $P(s), s \in \overline{L}$, can reach a marker state in the abstracted model via string $t \in \Sigma_0^*$, the system must also be able to reach a marker state from string s, via string $u \in \Sigma^*$ such that $P(su) = P(s)t$, as illustrated in Fig. 2.

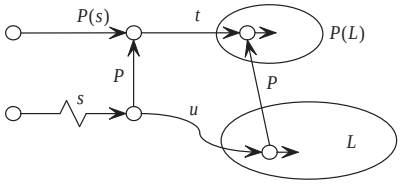


Fig. 2. Observer

Definition 1 (Observer): [16]Suppose 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

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

The symbol $\leq$ means that the first string is a prefix of the second one [1], [18]. If $\Sigma_0 = \Sigma$ or $\emptyset$, P is automatically an L-observer. Denote by $||L||$ the state size of the *canonical recognizer* [18] of L. If the natural projection P is an L-observer, the abstraction $P(L)$ can be computed in polynomial time in $||L||$, and $||P(L)|| \leq ||L||$ [15]. In fact, an arbitrary natural projection can be modified in polynomial time to be a natural observer by enlarging the observable event set.

According to Theorem 6 in [16], the observer property guarantees nonblocking control for a partially observed system. If P is an L-observer, then for a language $L \subseteq \Sigma^*$,

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

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

The synchronous product of two languages is nonblocking if and only if they are *synchronously nonconflicting* [18].

Definition 2 (Synchronously Nonconflicting Relation): Two languages $L_i \subseteq \Sigma_i^*, i = 1, 2$, are *synchronously nonconflicting* if

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

If P_0 has the observer property, we can check if two languages L_1 and L_2 are synchronously nonconflicting by checking this relation between their projections $P_0(L_1)$ and $P_0(L_2)$ instead. Since the models of $P_0(L_i)$ are smaller than those of L_i, significant computational effort may be saved, as suggested by the following proposition.

Proposition 2 (Synchronously Nonconflicting Criterion): Let $L_i \subseteq \Sigma_i^*, i = 1, 2$, and $\Sigma_0 \supseteq \Sigma_1 \cap \Sigma_2$. Let natural projections $Q_i : \Sigma_i^* \to (\Sigma_i \cap \Sigma_0)^*$ be L_i-observers for $i = 1, 2$; then

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

If $\Sigma_1 = \Sigma_2$, the synchronously nonconflicting property is identical to the usual nonconflicting property as in [1], [18]. Just as the nonconflicting property preserves controllability, so does the synchronously nonconflicting property.

Proposition 3: For $i = 1, 2$, let $K_i \subseteq L_i \subseteq \Sigma_i^*$ be controllable with respect to $\Sigma_{i,u} \subseteq \Sigma_i$ and L_i. If K_1 and K_2 are synchronously nonconflicting, then $K_1 \| K_2$ is controllable with respect to $\Sigma_{1,u} \cup \Sigma_{2,u}$ and $L_1 \| L_2$.

In case the two languages L_1 and L_2 are synchronously conflicting, another language, called a *coordinator*, must be introduced to resolve the conflict. As in Proposition 2, the two languages may interact only locally, i.e., share only a subset of events. In that case, instead of computing the coordinator based on the full two languages themselves, we perform this computation through their abstractions.

Proposition 4: Let $L_i \subseteq \Sigma_i^*, i = 1, 2$, and $\Sigma_0 \supseteq \Sigma_1 \cap \Sigma_2$. In the notation of Proposition 2, if Q_i is an L_i-observer ($i = 1, 2$) 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

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

Here, the two languages L_1 and L_2 may be synchronously conflicting. The coordinator L_0 depends only on the event set that 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 Local Control

A *local* [9] controller monitors and disables only the events in an *observable* event subset. In the previous section we saw that the observer property is a sufficient condition for nonblocking local control. Now we find conditions for *optimal* (i.e., *maximally permissive*) local control.

An optimal supervisor with full observation normally disables the nearest controllable events preceding or "upstream"

to a prohibited uncontrollable event (σ, say). However, if some of these controllable events are unobservable, the local supervisor must disable other controllable events encountered earlier to preclude σ. Hence the local supervisor is often more restrictive. To remove this restriction, the observable event set must be taken large enough to contain all the nearest controllable events preceding σ. Such a local supervisor will prevent the occurrence of an uncontrollable event while allowing maximal freedom to the system. A projection with such an observable event set is called *output control consistent* (OCC) (cf. [19]).

Definition 3 (OCC): Let $\Sigma_0 \subseteq \Sigma$ be an observable event set and $\Sigma_u \subseteq \Sigma$ be an uncontrollable event set. The natural projection $P : \Sigma^* \to \Sigma_0^*$ is *output control consistent (OCC)* for language $L \subseteq \Sigma^*$, if for every string $s \in L$ such that

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

where

$$P(s') \neq \epsilon \text{ and } (\forall t < s')P(t) < P(s'),$$

$$\sigma_i \in \Sigma - \Sigma_0, i = 1, \ldots, k-1, \text{ but } \sigma_k \in \Sigma_0,$$

we have the property

$$\sigma_k \in \Sigma_u \Longrightarrow (\forall i = 1, \ldots, k)\sigma_i \in \Sigma_u.$$

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. A polynomial algorithm has been developed to refine a natural projection to be OCC. Note that if $\Sigma_0 = \Sigma$ or $\emptyset$, P is automatically OCC for L.

Equipped with this new concept, we will bring in a practical and concise sufficient condition for optimal nonblocking local control. First, we introduce a notation for a useful class of controllable sublanguages. With $E, L_m \subseteq \Sigma^*$, let $\mathcal{C}(E\|L_m, \overline{L_m})$ be the class of controllable sublanguages of $E\|L_m$ with respect to $\overline{L_m}$.

Since plant components are often autonomous and independent agents, we shall assume the following structure for the plant. Let the index set be $\mathbf{n} := \{1, \cdots, n\}$ and assume $\Sigma_i \cap \Sigma_j = \emptyset, i \neq j \in \mathbf{n}$. Let $M_i \subseteq \Sigma_i^*, i \in \mathbf{n}$, and define

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

Definition 4 (Shuffle Plant): A *shuffle plant* is composed from $n(\geq 1)$ nonblocking and pairwise disjoint components, in the foregoing sense, whose marked languages are $M_i, i \in \mathbf{n}$.

The full alphabet of the shuffle plant is $\Sigma := \bigcup_{i=1}^n \Sigma_i$.

Proposition 5 (Optimal Nonblocking Local Control): Let the uncontrollable and observable event subsets of the shuffle plant be $\Sigma_u, \Sigma_0 \subseteq \Sigma$, respectively, and let the control specification be $E \subseteq \Sigma_0^*$. Bring in natural projections:

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

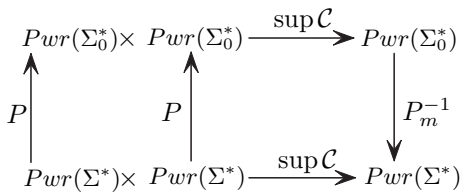
$$Q_i := P_0|\Sigma_i^* : \Sigma_i^* \to (\Sigma_i \cap \Sigma_0)^*, i \in \mathbf{n}.$$

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

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

Proof: See [2]. ■

The proposition is illustrated by the commutative diagram, Fig. 3. Language $\sup \mathcal{C}(E\|L_m, L)$ describes the optimal supervisor obtained by the monolithic approach. It is computed from E and the full plant L_m. On the other hand, the language $\sup \mathcal{C}(E \cap P(L_m), P(L))$ describes the local supervisor obtained from E and the plant abstraction $P_0(L_m)$. It requires less computation. Moreover, the combination of this local supervisor and the plant is equivalent to the monolithic supervisor, i.e., has the same controlled behavior.



In the diagram $P_m^{-1}(L) := P^{-1}(L) \cap L_m$

Fig. 3. Optimal Nonblocking Local Control

In case the observable event subset is the union of a subcollection of component alphabets, namely,

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

(ϕ being a permutation of $\mathbf{n} := \{1, \ldots, n\}$) the conditions in the proposition hold automatically. By the proposition the optimal control of a shuffle plant is determined only by the components sharing events with the control specifications. The other unrelated plant components play no role in the control synthesis. This property has been pointed out in [9], [18], but Proposition 5 extends it to a more general situation.

IV. Control of the Shuffle Plant

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

For definiteness, suppose there are three control specifications $E_i (i = 1, 2, 3)$ for the shuffle plant and each one applies only to those plant components whose indices make up the set $N_i \subseteq \mathbf{n}, i = 1, 2, 3$. The three groups of plant components come from the decomposition of the given plant and each specification may in fact be a combination of several simple specifications within the group. Assume

$$(\forall i, j \in \{1, 2, 3\})N_i \not\subseteq N_j.$$

as otherwise the two groups could be combined as one. The corresponding subsystems have the marked languages

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

The alphabets of the subsystems are

$$\Upsilon_i := \bigcup_{j\in N_i} \Sigma_j, i = 1, 2, 3.$$

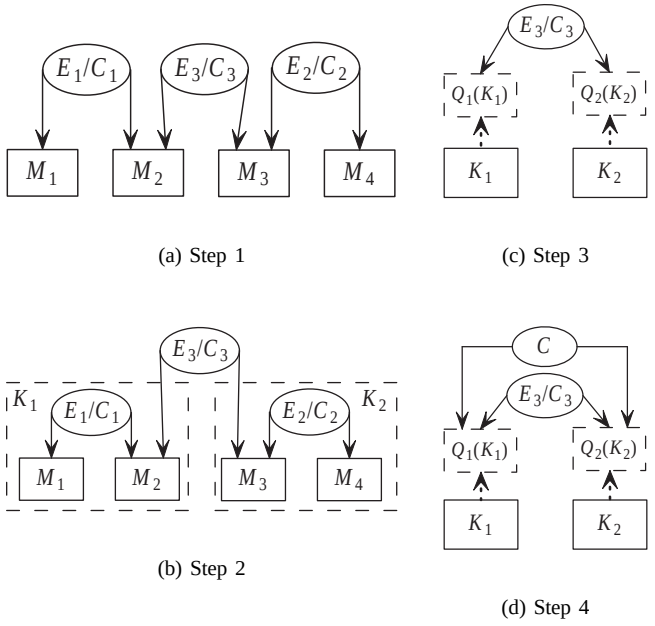
Thus, $E_i \subseteq \Upsilon_i^*, i = 1, 2, 3$.

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

$$K := \sup \mathcal{C}(E_1||E_2||E_3||L_m, L). \quad (2)$$

An automaton recognizing K is the monolithic supervisor for the control problem.

The proposed structural approach solves the problem in four steps. We illustrate the procedure by the example in Fig. 4. Here $n = 4, N_1 = \{1, 2\}, N_2 = \{3, 4\}$, and $N_3 = \{2, 3\}$.



(a) Step 1

(c) Step 3

(b) Step 2

(d) Step 4

Fig. 4. Nonblocking Control of A Shuffle System

As shown in Fig 4(a), the first step is to find local modular supervisors for all three control specifications, namely the languages

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

For simpler representation and implementation, it is useful to compute their supervisor reductions [13] $C_i, i = 1, 2, 3$, such that

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

Rather than verify the nonconflicting property among the three modular supervisors, the second step organizes the shuffle plant and local modular supervisors into the hierarchical structure in Fig. 4(b). Here the controlled systems represented by K_1 and K_2 are considered as two plant components which are further regulated by supervisor K_3. In addition, K_1, K_2 could interact through shared events, though this situation is not reflected in Fig. 4(b).

Using the model abstraction technique, step 3 checks if the shuffle plant supervised by the three local modular supervisors is nonblocking. Since each subsystem is only partially related with K_3, any events not shared with K_3 can be projected out of the models of K_1 and K_2. These projections are realized by natural observers Q_1 and Q_2, as specified in Proposition 6. Hence the verification of K_1, K_2, and K_3 can be achieved through a relatively simple computation on $Q_1(K_1)$, $Q_2(K_2)$, and K_3, as shown in Fig. 4(c).

Proposition 6: Let $\Upsilon_0 \supseteq \Upsilon_3 \cup (\Upsilon_1 \cap \Upsilon_2)$ and

$$Q_i : \Upsilon_i^* \to (\Upsilon_i \cap \Upsilon_0)^*, i = 1, 2.$$

Suppose that for $i = 1, 2, Q_i$ are K_i-observers and

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

where $K_i (i = 1, 2, 3)$ are defined in (3). Then K_1, K_2, K_3 are synchronously nonconflicting and the conjunction of the three local modular supervisors is the maximally permissive solution to the control problem, namely,

$$\begin{aligned} \overline{K_1||K_2||K_3} &= \overline{K_1}||\overline{K_2}||\overline{K_3}, \\ K_1||K_2||K_3 &= K. \end{aligned}$$

Proof: See [2]. ■

In case this verification fails, i.e., $Q_1(K_1), Q_2(K_2), K_3$ are conflicting, we proceed to step 4 to design a coordinator for the three supervisors. The coordination of subsystems has been addressed in Proposition 4. Using the same idea, we can design a coordinator for the three supervisors $K_i (i = 1, 2, 3)$.

In principle, the coordinator for the three supervisors is another supervisor for the new plant composed of K_3 and the abstractions of K_1 and K_2. The specification for this high level plant is simply Υ^*, as the only purpose of the coordinator is to resolve the conflict among the three supervisors. This coordination scheme is shown in Fig. 4(d).

The supervisor for the high level plant is then

$$K_C := \sup \mathcal{C}(Q_1(K_1)||Q_2(K_2)||K_3, Q_1(\overline{K_1})||Q_2(\overline{K_2})||\overline{K_3}) \quad (4)$$

Its supervisor reduction is a language C such that.

$$K_C = C||Q_1(K_1)||Q_2(K_2)||K_3 \quad (5)$$

$$\overline{K_C} = \overline{C}||Q_1(\overline{K_1})||Q_2(\overline{K_2})||\overline{K_3} \quad (6)$$

The following proposition provides a sufficient condition that a supervisor recognizing language C is the coordinator for the three modular supervisors.

Proposition 7: Let $\Upsilon_0 \supseteq \Upsilon_3 \cup (\Upsilon_1 \cap \Upsilon_2)$ and

$$P_0 : \Sigma^* \to \Upsilon_0^*.$$

Then $Q_i = P_0|\Upsilon_i^*, i = 1, 2$. If for $i = 1, 2, Q_i$ are K_i-observers and for $j \in \mathbf{n}$, $P_0|\Sigma_j^*$ are OCC for $\overline{M_j}$, then

language C defined by (4), (5), and (6) is a coordinator for supervisors $K_i (i = 1, 2, 3)$ defined by Equation (3), namely,

$$\overline{K_1||K_2||K_3||C} = \overline{K_1}||\overline{K_2}||\overline{K_3}||\overline{C}, \quad (7)$$

$$K_1||K_2||K_3||C = K. \quad (8)$$

Proof: Thanks to (5) and (6),

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

Furthermore, (4) 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

$$K_C \subseteq R^{-1}(K_3) \text{ and } \overline{K_C} \subseteq R^{-1}(\overline{K_3}),$$

where R is the natural projection $R : \Upsilon_0^* \to \Upsilon_3^*$. It follows immediately that

$$K_3||K_C = K_C \text{ and } \overline{K_3}||\overline{K_C} = \overline{K_C} \quad (9)$$

Finally we have

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

The proof of (7) 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 be confirmed by Proposition 4. The presumption already ensures that the Q_i are K_i-observers $(i = 1, 2)$, so we just need to show

$$\overline{Q_1(K_1)||Q_2(K_2)||K_C} = Q_1(\overline{K_1})||Q_2(\overline{K_2})||\overline{K_C}. \quad (10)$$

Considering (9), 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 (10) follows and (7) is proved.

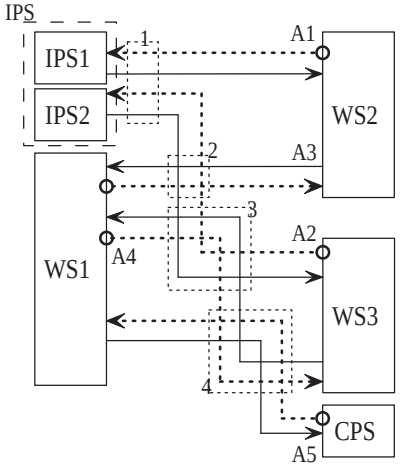
Since we have already shown that $K_1||K_2||K_3||C = K_1||K_2||K_3||K_C$, (8) is transformed to

$$K_1||K_2||K_3||K_C = \sup \mathcal{C}(E_1||E_2||E_3||L_m, L). \quad (11)$$

Then (11) follows immediately from Proposition 4.2.2 of [18], and the proof is complete. ■

V. An AGV Example

We apply the proposed control scheme to the coordination of a system of automatic guided vehicles (AGVs) serving a manufacturing work cell, adapted from [5]. The system consists of two input stations IPS1 and IPS2 for parts of types 1, 2; three workstations WS1, WS2, WS3; one completed-parts station CPS; and five AGVs (A1, ..., A5). The AGVs travel along fixed circular routes, on which they are loaded and unloaded by stations and machines, as shown in Fig. 5. The dashed rectangles are zones shared by the AGVs and each zone can be occupied by at most one AGV at a time. Further details are provided in Section 4.7 of [18].



⟶: Direction Loaded
$\cdots>$: Direction Unloaded
○: Parking Location

Fig. 5. AGV System

The plant model consists of five automata $\mathbf{A}_i (i = 1, \ldots, 5)$ corresponding to the five AGVs, with alphabets $\Sigma_i, i = 1, \ldots, 5$. There are totally eight automaton models for control specifications. Four ($\mathbf{Z}_i, i = 1, \ldots, 4$) stand for the zone restrictions, three ($\mathbf{WS}_i, i = 1, \ldots, 3$) for operational constraints in the workstations, and one (**IPS**) for a restriction on the common loading area between IPS1 and IPS2. Events and state transition diagrams of the automata can be found in [18] as cited.

The connecting relationships among these models of plant and control specifications are represented in Fig. 6, where a block is a plant component $\mathbf{A}_i (i = 1, \ldots, 5)$, and an oval a control specification. A line connects a plant component and a specification if and only if they share common events.

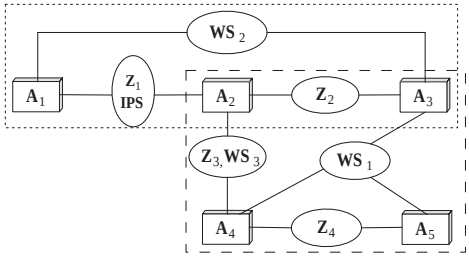


Fig. 6. Connecting Relationships and Decomposition

As prescribed in Section IV, we first design the local modular supervisors of the eight specifications. The sizes of these supervisors are listed in Table I.

In the second step, we decompose the whole AGV system into two subsystems, bounded by dashed lines in Fig. 6. Plant components $\mathbf{A}_1$ to $\mathbf{A}_3$ constitute subsystem 1 and components $\mathbf{A}_2$ to $\mathbf{A}_5$ subsystem 2. Combining the supervisors and the plant of each subsystem, we obtain automaton models for

TABLE I

LOCAL MODULAR SUPERVISORS

Spec	Sup (State#, Trans#)	Spec	Sup (State#, Trans#)
$\mathbf{Z}_1$	(2, 14)	$\mathbf{WS}_1$	(4, 40)
$\mathbf{Z}_2$	(2, 14)	$\mathbf{WS}_2$	(2, 12)
$\mathbf{Z}_3$	(2, 17)	$\mathbf{WS}_3$	(2, 21)
$\mathbf{Z}_4$	(2, 11)	**IPS**	(2, 16)

the controlled behaviors of the two subsystems, $\mathbf{S}_1$ and $\mathbf{S}_2$. They are both nonblocking.

$$\mathbf{S}_1\text{(State\#, Trans\#): } (88, 169)$$

$$\mathbf{S}_2\text{(State\#, Trans\#): } (1702, 4584)$$

In step 3, we verify the nonconflicting property between them through abstractions based on a natural observer for the two subsystems. Set the observable event subset of the natural projection to be

$$\Upsilon := \Sigma_2 \cup \Sigma_3 \cup \{11, 43\}.$$

Then natural projection $P : \Sigma^* \rightarrow \Upsilon^*$ meets the conditions in Proposition 7.

$$Project(\mathbf{S}_1)\text{(State\#, Trans\#) : } (54, 94)$$

$$Project(\mathbf{S}_2)\text{(State\#, Trans\#) : } (231, 486)$$

The synchronous product of the two projections is a *blocking* automaton with 396 states and 734 transitions. Consequently, $\mathbf{S}_1$ and $\mathbf{S}_2$ are synchronously conflicting.

To resolve this conflict we synthesize a coordinator by Proposition 7, using the projection models as the plant and language Υ^* as the control specification. The coordinator is just the modular supervisor that prunes the projection automaton to make it controllable and nonblocking.

$$\text{Coordinator (State\#, Trans\#):} (27, 241)$$

In [18] the coordinator presented is an automaton with 29 states and 64 transitions. That coordinator was derived as a natural projection of the monolithic supervisor for the AGV system, which has 4406 states and 11338 transitions. In contrast, the new structured approach of this paper completely avoids the computation of the monolithic supervisor. The most costly computation is for $\mathbf{S}_2$, which requires only 1702 states and 4584 transitions. The saving in computational effort is evident.

We can obtain a simpler solution to this example with even less computation, if we use the approach presented in [3]. The new solution includes the identical group of local modular supervisors and a smaller coordinator of only 7 states.

VI. CONCLUSIONS

Although the synthesis of an optimal and nonblocking supervisor generally demands exponential time, one may often avoid the worst case and design the supervisor using modularity and model abstraction techniques. A discrete-event system should, if possible, be divided and organized into subsystems based on the dependency relationships among plant components and control specifications. The decomposition will impose a hierarchical structure on the shuffle system. The supervisors of the low level subsystems are computed independently without regard to their mutual conflict. Subsequently coordination is realized by high level supervision of these controlled subsystems. To reduce computational complexity, the high level supervisors are computed 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.

REFERENCES

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

[2] Feng, L. *Computationally Efficient Supervisor Design*. Ph.D. Thesis, In Preparation, Electrical and Computer Engineering, University of Toronto.

[3] Feng, L. and Wonham, W.M. 2006. Computationally Efficient Supervisor Design: Control Flow Decomposition. *WODES 2006*.

[4] Gohari, P. and Wonham, W.M. 2000. On the Complexity of Supervisory Control Design in the RW Framework. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, 30(5): 643-652.

[5] Holloway, L.E. and Krogh, B.H. 1990. Synthesis of Feedback Logic Control for A Class of Controlled Petri Nets. *IEEE Transactions on Automatic Control*, 35(5): 514-523.

[6] Leduc, R.J, Brandin, B.A., Lawford, M., and Wonham, W.M. 2005. Hierarchical Interface-Based Supervisory Control-Part I: Serial Case. *IEEE Transactions on Automatic Control*, 50(9): 1322-1335.

[7] Leduc, R.J., Lawford, M, and Wonham, W.M. 2005. Hierarchical Interface-Based Supervisory Control-Part II: Parallel Case. *IEEE Transactions on Automatic Control*, 50(9): 1336-1348.

[8] Lee, S.H. and Wong, K.C. 2002. Structural Decentralized Control of Concurrent Discrete-Event Systems. *European Journal of Control*, 8(5): 477-491.

[9] de Queiroz, M.H. and Cury, J.E.R. 2000. Modular Control of Composed Systems. *Proceedings of the American Control Conference*, Chicago, Illinois, 4051-4055.

[10] Rohloff, K. and Lafortune, S. 2002. On the Computational Complexity of the Verification of Modular Discrete-Event Systems. *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada.

[11] Schmit, K., Reger, J., and Moor, T. 2004. Hierarchical Control for Structural Decentralized DES. *WODES04*, 289-294.

[12] Schmit, K., Moor, T., and Perk, S. 2005. A Hierarchical Architecture for Nonblocking Control of Decentralized Discrete Event Systems. *Proceedings of the 13th Mediterranean Conference on Control and Automation*, 902-907.

[13] Su, R. and Wonham, W.M. 2004. Supervisor Reduction for Discrete-Event Systems. *Discrete Event Dynamic Systems: Theory and Application*, 14(1): 31-53.

[14] Willner, Y. and Heymann, M. 1991. On Supervisory Control of Concurrent Discrete-Event Systems. *International Journal of Control*, 54(5): 1143-1169.

[15] Wong, K.C. 1998. On the Complexity of Projections of Discrete-Event Systems, *Proceedings of the Fourth Workshop on Discrete Event Systems, WODES98*, 201-206.

[16] Wong, K.C. and Wonham, W.M. 1996. Hierarchical Control of Discrete-Event Systems. *Discrete Event Dynamic Systems: Theory and Application*, 6(3): 241-273.

[17] Wong, K.C. and Wonham, W.M. 1998. Modular Control and Coordination of Discrete-Event Systems. *Discrete Event Dynamic Systems: Theory and Application*, 8(3): 247-297.

[18] Wonham, W.M. 2005. *Supervisory Control of Discrete-Event Systems*, Department of Electrical and Computer Engineering, University of Toronto, http://www.control.toronto.edu/DES.

[19] Zhong, H. and Wonham, W.M. 1990. On the Consistency of Hierarchical Supervision in Discrete-Event Systems. *IEEE Transactions on Automatic Control*, 35(10): 1125-1134.