Supervisor Localization: A Top-Down Approach to Distributed Control of Discrete-Event Systems

Kai Cai, *Member, IEEE*, and W. M. Wonham, *Life Fellow, IEEE*

***Abstract*—We study the design of distributed control for discrete-event systems (DES) in the framework of supervisory control theory. We view a DES as comprised of a group of agents, acting independently except for specifications on global (group) behavior. The central problem investigated is how to synthesize local controllers for individual agents such that the resultant controlled behavior is identical with that achieved by global supervision. In the case of small-scale DES, a supervisor localization algorithm is developed that solves the problem in a top-down fashion: first, compute a global supervisor, then decompose it to local controllers while preserving the global controlled behavior. In the case of large-scale DES where owing to state explosion a global supervisor might not be feasibly computable, a decomposition-aggregation solution procedure is developed that combines the supervisor localization algorithm with an efficient modular control theory.**

***Index Terms*—Discrete-event systems (DES), supervisor localization, supervisory control.**

I. Introduction

RAPID advances in communication networks and embedded computing technologies have made *distributed systems* pervasive in engineering practice: examples include multi-robot search teams, wireless sensor networks, and automated guided vehicles. By these are meant systems that consist of multiple autonomous agents locally interacting in pursuit of a global goal. To govern this type of system, attention has been focused on *distributed control*: each agent has its own local observation and control strategies—but with no external supervisor, thus embodying individual autonomy. Little work has been reported on distributed control of *discrete-event systems* (DES) in the framework of *supervisory control theory* (SCT) [1].

SCT was initiated by Ramadge and Wonham [2], [3], with cornerstone results established for a monolithic architecture, wherein all plant components are controlled by a single centralized supervisor. With this supervisor, the controlled behavior can be made *optimal* (i.e., minimally restrictive) with respect to imposed specifications, as well as *nonblocking*. Stimulated by the twin goals of improving understandability of control logic and reducing computational effort of the monolithic approach, subsequent literature has witnessed the emergence of alternative modular system architectures—decentralized [4]–[10], hierarchical [11], [12], and heterarchical [13]–[15]. The defining characteristic of these architectures is a supervisor-subordinate paradigm: a monolithic supervisor, or an organization of modular supervisors, monitors the behavior of subordinate agents and makes all decisions on their behalf, while the controlled agents themselves act blindly based on the commands they receive. Intuitively one could think of these supervisors as external to, rather than built into, the subordinate agents. From this perspective, monolithic and modular architectures are not, in our view, properly considered to be purely distributed. We take the view that distributed architecture is a flat system organization where global functions emerge through the collective actions of individual agents and are not, at least directly, guided by higher-level, external supervisors. With this in mind, we address the following question: *given a collection of independent agents as the plant and some desired collective behavior as the specification, what should individual agents do (in terms of sensing and decision making) so as to enforce the specification, and realize performance identical to that achieved by optimal and nonblocking monolithic control?*

Only recently has work on distributed architecture addressing similar questions begun to appear. Su and Thistle [16] present a distributed supervisor synthesis approach: first, local abstractions of the overall system are computed for individual agents based on *weak bisimulation*, then local controllers for each agent are synthesized from the agent and the abstraction models. Although in the latter step a *competing/cooperation policy* is defined that determines the restrictiveness of the resulting controlled behavior, optimal control is not considered. Mannani and Gohari [17] propose implementing monolithic supervisors through synthesizing local *guard formulas* and *updating functions* for individual agents modeled as extended finite state machines. But since all languages involved are assumed to be *prefix-closed*, nonblocking control is not addressed. Distributed implementation of monolithic supervisors is also investigated in [18]; the approach is first to convert a monolithic supervisor into a *bounded* and *distributable* Petri net, which in turn is converted into a set of automata as local controllers for individual agents. The former conversion can be applied, however, only to a strict subclass of monolithic supervisors. Finally we note that a recent paper [19] as well as its conference precursor [20], which had been unknown to us, proposed a scheme in the SCT setting similar in general terms to our own, but provided a control synthesis equivalent only to one of our simple boundary cases. The technical differences are explained in footnote 5, Section III, below.

Manuscript received December 07, 2008; revised March 07, 2009. First published January 22, 2010; current version published March 10, 2010. This work was supported in part by the Natural Sciences and Engineering Research Council (Canada) under Grant 7399. Recommended by Associate Editor S. Haar.

The authors are with the Systems Control Group, Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: caikai@control.utoronto.ca; wonham@control.utoronto.ca).

Digital Object Identifier 10.1109/TAC.2009.2039237

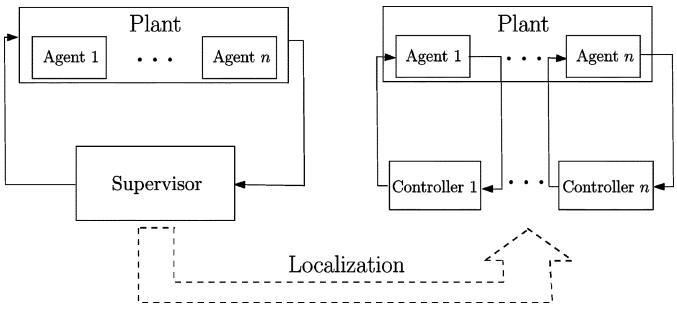


Fig. 1. Supervisor localization.

In this paper and its conference antecedent [21], we consider controlled behavior achieved by an arbitrary monolithic supervisor, and deal with both optimal and nonblocking control. Further, our approach can in principle handle large-scale systems, as will be demonstrated on a benchmark application, whereas only small-sized examples are given in the cited previous work.

We note that the term "distributed architecture" along with "distributed control" and "agent" has been used in the literature with different meanings. For instance, [22] formulates the distributed control problem as follows: "Consider a distributed networked DES modeled by automaton G. There are n agents observing the behavior of G using their own sets of sensors. The agents may be supervisors or diagnosers. The agents are able to communicate among each other... In this formulation, "distributed architecture" actually refers to decentralized architecture with communicating modular supervisors (or diagnosers). With decentralized supervision, the global control action is typically allocated among specialized supervisors enforcing individual specifications. By contrast, with distributed supervision (in our usage) it is allocated among the individual active agents.

Our investigation is carried out for both small- and large-scale DES. In the small-scale case, we assume that the monolithic supervisor of a given control problem can be computed. Given this assumption we propose a top-down approach: first, compute the monolithic supervisor, then decompose it into local controllers for individual agents. We call this approach supervisor localization, as displayed in Fig. 1.

The goal of supervisor localization is first of all to preserve the optimality and nonblockingness of the monolithic supervisor, namely to realize performance identical to that achieved by monolithic control. It is also desired that each localized controller be as simple as possible, so that individual strategies are more readily comprehensible; among diverse criteria of simplicity, we focus on state size. Both goals are achieved by a suitable extension of *supervisor reduction* [23], of which the essence is to project the plant model out of the supervisor model while preserving the controlled behavior. To localize the monolithic supervisor to a local controller for an individual agent, we carry the reduction idea one step further: in addition to projecting the plant model out of the supervisor, we also project out those transitions corresponding to the controls enforced by other agents. Namely, the localization procedure is conducted based solely on control information directly relevant to the target agent; we proceed this way for each agent in the plant, taken individually. The result is that each agent acquires its own local controller (see Fig. 1).

In large-scale systems, owing to state space explosion the monolithic supervisor might not be feasibly computable. Indeed, Gohari and Wonham [24] proved that monolithic supervisor synthesis is NP-hard, inasmuch as the state space size grows exponentially in the number of individual plant components and specifications. To manage such complexity, we propose combining supervisor localization with an efficient modular control theory [15]. This combination leads to a *decomposition-aggregation procedure* that systematically solves large-system problems in an alternative top-down manner: first, design an organization of modular supervisors that achieves optimal and nonblocking control, then decompose each of these modular supervisors into local controllers for the relevant agents.

The rest of the paper is organized as follows. Section II formulates the distributed control problem, for which solutions for small- and large-scale DES are presented in Sections III and IV, respectively. Section V states our conclusion.

II. Problem Formulation

The plant to be controlled is modeled by a (nonempty) generator

$$\mathbf{G} = (Y, \Sigma, \eta, y_0, Y_m)$$

where Y is the state set; $y_0 \in Y$ is the initial state; $Y_m \subseteq Y$ is the set of marker states; Σ is the finite event set, partitioned into Σ_c, the controllable event subset, and Σ_u, the uncontrollable subset; $\eta : Y \times \Sigma \to Y$ is the (partial) state transition function. In the usual way, η is extended to $\eta : Y \times \Sigma^* \to Y$ (pfn), and we write $\eta(y, s)!$ to mean that $\eta(y, s)$ is defined, where $y \in Y$ and $s \in \Sigma^*$. The *closed behavior* of $\mathbf{G}$ is the language

$$L(\mathbf{G}) := \{s \in \Sigma^* | \eta(y_0, s)!\}$$

and the *marked behavior* is

$$L_m(\mathbf{G}) := \{s \in L(\mathbf{G}) | \eta(y_0, s) \in Y_m\} \subseteq L(\mathbf{G}).$$

We focus on the case where $\mathbf{G}$ consists of component agents $\mathbf{G}^k$ defined over pairwise disjoint alphabets Σ^k ($k \in \mathcal{K}, \mathcal{K}$ an index set). So $\Sigma = \dot{\bigcup}\{\Sigma^k | k \in \mathcal{K}\}$. Let $L_k := L(\mathbf{G}^k)$ and $L_{m,k} := L_m(\mathbf{G}^k)$; then the closed and marked behaviors of $\mathbf{G}$ are

$$L(\mathbf{G}) = ||\{L_k | k \in \mathcal{K}\} \text{ and } L_m(\mathbf{G}) = ||\{L_{m,k} | k \in \mathcal{K}\}$$

where $||$ denotes *synchronous product* [1]. For simplicity we assume that for every $k \in \mathcal{K}$, $\mathbf{G}^k$ is *nonblocking* (i.e., $\bar{L}_{m,k} = L_k$)[1]. Then $\mathbf{G}$ is necessarily nonblocking (i.e., $\bar{L}_m(\mathbf{G}) = L(\mathbf{G})$).

With $\Sigma = \Sigma_c \dot{\cup} \Sigma_u$ we assign control structure to each agent

$$\Sigma_c^k = \Sigma^k \cap \Sigma_c, \quad \Sigma_u^k = \Sigma^k \cap \Sigma_u.$$

[1] $\bar{L}_{m,k}$ denotes the *(prefix) closure* [1] of $L_{m,k}$.

Let $k \in \mathcal{K}$. We say that a generator $\mathbf{LOC}^k$ (over Σ) is a *local controller* for agent $\mathbf{G}^k$ if $\mathbf{LOC}^k$ can disable only events in Σ_c^k. Precisely, for all $s \in \Sigma^*$ and $\sigma \in \Sigma$, there holds

$$s\sigma \in L(\mathbf{G}) \,\&\, s \in L(\mathbf{LOC}^k) \,\&\, s\sigma \notin L(\mathbf{LOC}^k) \Rightarrow \sigma \in \Sigma_c^k.$$

The observation scope of $\mathbf{LOC}^k$ is, however, neither confined within Σ^k nor fixed beforehand. In fact, it will be systematically determined to guarantee correct local control. Thus, while a local controller's control authority is strictly local, its observation scope need not, and generally will not, be. Whether or not the required capability of event observation on the part of an agent is feasible in practice will evidently be case-dependent, but need not be burdensome in many applications. An instance could be nearest-neighbor observation, as for motorists maneuvering through a congested intersection. With local controllers embedded, each agent acquires strictly local control and generally non-local observation strategies; the latter are critical to achieve useful synchronization with other agents, thereby ensuring correct local decisions[2].

The component agents are implicitly coupled through an imposed specification language E that imposes a behavioral constraint on $\mathbf{G}$. As in [4, Section 4] and subsequent literature (e.g., [5]), assume that E is *decomposable* into component specifications $E_p \subseteq \Sigma_{e,p}^*$ ($p \in \mathcal{P}$, $\mathcal{P}$ an index set), where the $\Sigma_{e,p} \subseteq \Sigma$ need not be pairwise disjoint; namely

$$E = \|\{E_p | p \in \mathcal{P}\}.$$

Thus E is defined over the alphabet $\Sigma_e := \bigcup\{\Sigma_{e,p} | p \in \mathcal{P}\}$. Let $P_e : \Sigma^* \to \Sigma_e^*$ be the corresponding *natural projection*, defined according to

$$\begin{aligned} P_e(\epsilon) &= \epsilon \\ P_e(\sigma) &= \begin{cases} \epsilon, & \text{if } \sigma \notin \Sigma_e \\ \sigma, & \text{if } \sigma \in \Sigma_e \end{cases} \\ P_e(s\sigma) &= P_e(s)P_e(\sigma),\ s \in \Sigma^*, \sigma \in \Sigma. \end{aligned}$$

In the usual way, P_e is extended to $P_e : Pwr(\Sigma^*) \to Pwr(\Sigma_e^*)$. We write $P_e^{-1} : Pwr(\Sigma_e^*) \to Pwr(\Sigma^*)$ for the inverse-image function of P_e, where $Pwr(\cdot)$ denotes powerset.

Let $F \subseteq \Sigma^*$, and recall that F is *controllable* (with respect to $\mathbf{G}$) if

$$\bar{F}\Sigma_u \cap L(\mathbf{G}) \subseteq \bar{F}.$$

Whether or not F is controllable, we denote by $\mathcal{C}(F)$ the set of all controllable sublanguages of F. $\mathcal{C}(F)$ is nonempty because the empty language $\emptyset$ is trivially controllable, hence always belongs to $\mathcal{C}(F)$. Further, $\mathcal{C}(F)$ contains a (unique) supremal element, denoted $sup\,\mathcal{C}(F)$ [3].

For the plant $\mathbf{G}$ and the specification E described above, let $\mathbf{SUP} = (X, \Sigma, \xi, x_0, X_m)$ be the corresponding monolithic supervisor that is optimal and nonblocking. Throughout the paper we assume that **SUP** is nonempty. The marked language of **SUP** can be expressed algebraically as

$$sup\mathcal{C}\left(P_e^{-1}E \cap L_m(\mathbf{G})\right) \subseteq \Sigma^*$$

and may or may not be feasibly computable.

Now we formulate the *Optimal Nonblocking Distributed Control Problem* ($\divideontimes$):

Construct a set of local controllers $\mathbf{LOC} = \{\mathbf{LOC}^k | k \in \mathcal{K}\}$, one for each agent, with $L(\mathbf{LOC}) = \bigcap\{L(\mathbf{LOC}^k) | k \in \mathcal{K}\}$ and $L_m(\mathbf{LOC}) = \bigcap\{L_m(\mathbf{LOC}^k) | k \in \mathcal{K}\}$, such that the following two properties hold:

$$L(\mathbf{G}) \cap L(\mathbf{LOC}) = L(\mathbf{SUP}), \tag{1a}$$

$$L_m(\mathbf{G}) \cap L_m(\mathbf{LOC}) = L_m(\mathbf{SUP}). \tag{1b}$$

We say that $\mathbf{LOC}$, satisfying (1a) and (1b), is *control equivalent* to **SUP** with respect to $\mathbf{G}$.

For the sake of easy implementation and comprehensibility, it would be desired in practice that the state sizes of local supervisors be very much less than that of their "parent" monolithic supervisor

$$(\forall\, k \in \mathcal{K})\ |\mathbf{LOC}^k| \ll |\mathbf{SUP}|$$

where $|\cdot|$ denotes state size of the argument. Inasmuch as this property is neither precise to state nor always achievable, it must needs be omitted from the formal problem statement; in applications, nevertheless, it should be kept in mind.

III. Solution for Small-Scale Systems

A. Supervisor Localization

We solve the distributed control problem ($\divideontimes$) for small-scale systems by developing a *supervisor localization* approach.

It follows from $\Sigma = \dot{\bigcup}\{\Sigma^k | k \in \mathcal{K}\}$ that the set $\left\{\Sigma_c^k \subseteq \Sigma_c | k \in \mathcal{K}\right\}$ forms a partition on Σ_c. Fix an element $k \in \mathcal{K}$. Following [23], we first establish a *control cover* on X, the (nonempty) state space of **SUP**, based only on control information pertaining to Σ_c^k, as captured by the following four functions. First define $E : X \to Pwr(\Sigma)$ according to

$$E(x) = \{\sigma \in \Sigma | \xi(x, \sigma)!\}.$$

Thus $E(x)$ denotes the set of events that are enabled at x. Next define $D^k : X \to Pwr\left(\Sigma_c^k\right)$ according to

$$D^k(x) = \Big\{\sigma \in \Sigma_c^k | \neg\xi(x, \sigma)! \,\&\, (\exists\, s \in \Sigma^*) \left[\xi(x_0, s) = x \,\&\, \eta(y_0, s\sigma)!\right]\Big\}.$$

So $D^k(x)$ is the set of controllable events in Σ_c^k that must be disabled at x. Notice that if an event $\sigma \in \Sigma_c^k$ is not in $D^k(x)$, then either $\sigma \in E(x)$ or σ is not defined at any state in Y that corresponds to x. Define $M : X \to \{1, 0\}$ according to

$$M(x) = 1 \text{ iff } x \in X_m.$$

[2]For simplicity we assume in this paper that observation of an event is simultaneous with its occurrence.

Thus M is a predicate on X that determines if a state is marked in **SUP**. Finally define $T: X \to \{1,0\}$ according to

$$T(x) = 1 \text{ iff } (\exists\, s \in \Sigma^*)\xi(x_0, s) = x \;\&\; \eta(y_0, s) \in Y_m.$$

So T is a predicate on X that determines if some corresponding state is marked in $\mathbf{G}$. Note that for each $x \in X$, we have by $(1b)$ that $T(x) = 0 \Rightarrow M(x) = 0$ and $M(x) = 1 \Rightarrow T(x) = 1$.

Definition 1: We define a binary relation $\mathcal{R}^k$ on X as follows. For $x, x' \in X$ we say that x and x' are *control consistent* (with respect to Σ_c^k) (cf [23, Section 2.2]), and write $(x, x') \in \mathcal{R}^k$, if

(i) $E(x) \cap D^k(x') = \emptyset = E(x') \cap D^k(x)$,
(ii) $T(x) = T(x') \Rightarrow M(x) = M(x')$.

$\Diamond$

Informally, a pair of states (x, x') is in $\mathcal{R}^k$ if (i) there is no event in Σ_c^k that is enabled at x but is disabled at x', or vice versa (consistent disablement information); and (ii) x and x' are both marked or unmarked in **SUP** provided that they are both marked or unmarked in $\mathbf{G}$ (consistent marking information). While $\mathcal{R}^k$ is reflexive and symmetric, it need not be transitive, and consequently not an equivalence relation. This fact leads to the following definition of *control cover*. Recall that a *cover* on a set X is a family of nonempty subsets (or *cells*) of X whose union is X.

Definition 2: Let I^k be some index set, and $\mathcal{C}^k = \left\{X_{i^k}^k \subseteq X | i^k \in I^k\right\}$ a cover on X. $\mathcal{C}^k$ is a *control cover* (cf [23, Definition 2.1]) on X (with respect to Σ_c^k) if

(i) $(\forall\, i^k \in I^k)\left(\forall\, x, x' \in X_{i^k}^k\right)(x, x') \in \mathcal{R}^k$;
(ii) $\left(\forall\, i^k \in I^k, \sigma \in \Sigma\right)$
$\left[(\exists\, j^k \in I^k)\left(\forall\, x \in X_{i^k}^k\right)\xi(x, \sigma)! \Rightarrow \xi(x, \sigma) \in X_{j^k}^k\right]$.

$\Diamond$

A control cover $\mathcal{C}^k$ lumps states of **SUP** into (possibly overlapping) cells $X_{j^k}^k (i^k \in I^k)$. According to (i) all states that reside in a cell $X_{i^k}^k$ must be pairwise control consistent; and (ii) for every event $\sigma \in \Sigma$, all states that can be reached from any state in $X_{i^k}^k$ by a one-step transition σ must be covered by the same cell $X_{j^k}^k$. Recursively, two states x, x' belong to a common cell in $\mathcal{C}^k$ if and only if (1) x and x' are control consistent; and (2) two future states that can be reached respectively from x and x' by the same string are again control consistent. We say that a control cover $\mathcal{C}^k$ is a *control congruence* if $\mathcal{C}^k$ happens to be a partition on X, namely its cells are pairwise disjoint.

Having established a control cover $\mathcal{C}^k$ on X based only on the control information of Σ_c^k, we can then obtain an induced generator $\mathbf{J}^k = \left(I^k, \Sigma, \kappa^k, i_0^k, I_m^k\right)$ by the following construction (cf [23, Section 2.2]):

(i) $i_0^k \in I^k$ such that $x_0 \in X_{i_0^k}^k$;
(ii) $I_m^k = \left\{i^k \in I^k | X_{i^k}^k \cap X_m \neq \emptyset\right\}$;
(iii) $\kappa^k : I^k \times \Sigma \to I^k$ (pfn) with $\kappa^k(i^k, \sigma) = j^k$ if $\left(\exists\, x \in X_{i^k}^k\right)\xi(x, \sigma) \in X_{j^k}^k$ & $\left(\forall\, x' \in X_{i^k}^k\right)\left[\xi(x', \sigma)! \Rightarrow \xi(x', \sigma) \in X_{j^k}^k\right]$.

Note that, owing to overlapping, the choices of i_0^k and κ^k may not be unique, and consequently $\mathbf{J}^k$ may not be unique. In that case we pick an arbitrary instance of $\mathbf{J}^k$. Clearly if $\mathcal{C}^k$ happens to be a control congruence, then $\mathbf{J}^k$ is unique.

Returning to the partition $\left\{\Sigma_c^k \subseteq \Sigma_c | k \in \mathcal{K}\right\}$ on Σ_c, we can thus obtain a set of induced generators $\mathbf{J} := \{\mathbf{J}^k | k \in \mathcal{K}\}$. Let $L(\mathbf{J}) := \bigcap\{L(\mathbf{J}^k) | k \in \mathcal{K}\}$ and $L_m(\mathbf{J}) := \bigcap\left\{L_m(\mathbf{J}^k) | k \in \mathcal{K}\right\}$. Our first result states that $\mathbf{J}$ is a solution to the distributed control problem $(\divideontimes)$.

Proposition 1: $\mathbf{J}$ is control equivalent to **SUP** with respect to $\mathbf{G}$, i.e.

$$L(\mathbf{G}) \cap L(\mathbf{J}) = L(\mathbf{SUP})$$
$$L_m(\mathbf{G}) \cap L_m(\mathbf{J}) = L_m(\mathbf{SUP}).$$

Proof: An argument applicable to $L_m(\mathbf{G}) \cap L_m(\mathbf{J}) = L_m(\mathbf{SUP})$ and $L(\mathbf{SUP}) \subseteq L(\mathbf{G}) \cap L(\mathbf{J})$ can be found in [23, Appendix P-1]. Here we need only show $L(\mathbf{G}) \cap L(\mathbf{J}) \subseteq L(\mathbf{SUP})$, and proceed by induction. For the base case, first note that none of $L(\mathbf{G}), L(\mathbf{J})$, and $L(\mathbf{SUP})$ is empty; thus the empty string ε belongs to all of them. For the inductive step, we suppose that $(s \in L(\mathbf{G}) \cap L(\mathbf{J})) \Rightarrow (s \in L(\mathbf{SUP}))$. Let $\sigma \in \Sigma$ and assume $s\sigma \in L(\mathbf{G}) \cap L(\mathbf{J})$; we must show that $s\sigma \in L(\mathbf{SUP})$. By hypothesis we have $s \in L(\mathbf{SUP})$. If $\sigma \in \Sigma_u$, then $s\sigma \in L(\mathbf{SUP})$ because $L(\mathbf{SUP})$ is controllable. Otherwise $\sigma \in \Sigma_c$, and there must exist $k \in \mathcal{K}$ such that $\sigma \in \Sigma_c^k$. It follows from $s\sigma \in L(\mathbf{J})$ that $s\sigma \in L(\mathbf{J}^k)$ and $s \in L(\mathbf{J}^k)$. So $\kappa^k\left(i_0^k, s\sigma\right)!$ and $\kappa^k\left(i_0^k, s\right)!$. Let $i_n^k := \kappa^k\left(i_0^k, s\right)$. By definition of κ^k we have $\left(\exists\, x \in X_{i_n^k}^k, \exists\, x' \in X\right)\xi(x, \sigma) = x'$, which then entails $\sigma \in E(x)$. Since $\xi(x_0, s) \in X_{i_n^k}^k$, we have $(x, \xi(x_0, s)) \in \mathcal{R}^k$. Hence $\sigma \notin D^k(\xi(x_0, s))$, which implies that either $\xi(\xi(x_0, s), \sigma)!$ or $(\forall\, t \in \Sigma^*)\xi(x_0, t) = \xi(x_0, s) \Rightarrow \neg\eta(y_0, t\sigma)!$ But the latter case does not arise, because letting $t = s$ we have $s\sigma \in L(\mathbf{G})$. Therefore we conclude $\xi(\xi(x_0, s), \sigma)!$, i.e., $s\sigma \in L(\mathbf{SUP})$. ■

Next we investigate if the converse is true: can a set of generators that is control equivalent to **SUP** always be induced from a set of suitable control covers on X? For this we bring in the following two definitions.

Definition 3: A generator $\mathbf{LOC} = (Z, \Sigma, \zeta, z_0, Z_m)$ is *normal* (with respect to **SUP**) [23, Definition 2.2] if

(i) $(\forall\, z \in Z)(\exists\, s \in L(\mathbf{SUP}))\ \zeta(z_0, s) = z$;
(ii) $(\forall z \in Z, \sigma \in \Sigma)\zeta(z, \sigma)! \Rightarrow (\exists\, s \in L(\mathbf{SUP}))[\zeta(z_0, s) = z \;\&\; s\sigma \in L(\mathbf{SUP})]$;
(iii) $(\forall\, z \in Z_m)(\exists\, s \in L_m(\mathbf{SUP}))\ \zeta(z_0, s) = z$.

$\Diamond$

Informally, a generator is normal (with respect to **SUP**) if (i) each of its states is reachable by at least one string in $L(\mathbf{SUP})$; (ii) a one-step transition σ is defined at a state z only if z is reached by a string s in $L(\mathbf{SUP})$ such that membership of $s\sigma$ in $L(\mathbf{SUP})$) is preserved; and (iii) each of its marked states is reachable by at least one string in $L_m(\mathbf{SUP})$.

Definition 4: Given two generators $\mathbf{LOC} = (Z, \Sigma, \zeta, z_0, Z_m)$ and $\mathbf{J} = (I, \Sigma, \kappa, i_o, I_m)$, we say that $\mathbf{LOC}$ and $\mathbf{J}$ are *DES-isomorphic with isomorphism* θ [23, Definition 2.3] if there exists a map $\theta: Z \to I$ such that

(i) $\theta: Z \to I$ is a bijection;
(ii) $\theta(z_0) = i_0 \;\&\; \theta(Z_m) = I_m$;
(iii) $(\forall\, z \in Z, \sigma \in \Sigma)\zeta(z, \sigma)! \Rightarrow [\kappa(\theta(z), \sigma)! \;\&\; \kappa(\theta(z), \sigma) = \theta(\zeta(z, \sigma))]$;
(iv) $(\forall\, i \in I, \sigma \in \Sigma)\kappa(i, \sigma)! \Rightarrow [(\exists\, z \in Z)\zeta(z, \sigma)! \;\&\theta(z) = i]$.

$\Diamond$

Under normality and DES-isomorphism, we have the following result.

Theorem 1: Let $\mathbf{LOC} := \left\{\mathbf{LOC}^k = \left(Z^k, \Sigma, \zeta^k, z_0^k, Z_m^k\right) | k \in \mathcal{K}\right\}$ be a set of normal generators that is control equivalent to **SUP** with respect to $\mathbf{G}$. Then there exists a set of control covers $\mathcal{C} := \{\mathcal{C}^k | k \in \mathcal{K}\}$ on X with a corresponding set of induced generators $\mathbf{J} := \{\mathbf{J}^k | k \in \mathcal{K}\}$ such that for every $k \in \mathcal{K}$, $\mathbf{J}^k$ and $\mathbf{LOC}^k$ are DES-isomorphic.

Proof: Let $k \in \mathcal{K}$. We must show that there exists a control cover $\mathcal{C}^k$ such that the induced generator $\mathbf{J}^k$ is DES-isomorphic to $\mathbf{LOC}^k$. Let $z^k \in Z^k$, and define

$$X(z^k) := \Big\{x \in X | (\exists\, s \in L(\mathbf{SUP}))\xi(x_0, s) = x \;\&\; \zeta^k\left(z_0^k, s\right) = z^k\Big\}.$$

Letting $\mathcal{C}^k := \{X(z^k) | z^k \in Z^k\}$, we claim that $\mathcal{C}^k$ is a control cover on X with respect to Σ_c^k. An argument proving this claim can be found in [23, Appendix P-2], except for the following:

$$\begin{aligned}(\forall\, z^k \in Z^k, \forall\, a, b \in X(z^k))\; & E(a) \cap D^k(b) \\ &= \emptyset \\ &= E(b) \cap D^k(a).\end{aligned}$$

Let $\sigma \in \Sigma$ and assume $\sigma \in E(a)$; it will be shown that $\sigma \notin D^k(b)$. If $\sigma \in \Sigma - \Sigma_c^k$, then by definition $\sigma \notin D^k(b)$. Otherwise $\sigma \in \Sigma_c^k$, and it follows from $\sigma \in E(a)$ that $(\exists\, s \in L(\mathbf{SUP}))\, \xi(x_0, s) = a \;\&\; \xi(a, \sigma)!$. So $s\sigma \in L(\mathbf{SUP})$. Since $\mathbf{LOC}$ is control equivalent to **SUP**, we obtain that $s\sigma \in L(\mathbf{LOC}^k)$, i.e., $\zeta^k\left(\zeta^k\left(z_0^k, s\right), \sigma\right)!$. It then follows from $a \in X(z^k)$ that $\zeta^k(z^k, \sigma)!$. Further, by $b \in X(z^k)$ we have $(\exists\, t \in L(\mathbf{SUP}))\, \xi(x_0, t) = b \;\&\; \zeta^k\left(z_0^k, t\right) = z^k$, which entails $t\sigma \in L(\mathbf{LOC}^k)$. If $t\sigma \notin L(\mathbf{G})$, then trivially $\sigma \notin D^k(b)$; for the case $t\sigma \in L(\mathbf{G})$, since only $\mathbf{LOC}^k$ has control authority on Σ_c^k, we have that

$$(\forall\, k' \in \mathcal{K})\; k' \neq k \Rightarrow \zeta^{k'}\left(z_0^{k'}, t\sigma\right)!, \text{ i.e., } t\sigma \in L(\mathbf{LOC}^{k'})$$

and thus $t\sigma \in L(\mathbf{G}) \cap L(\mathbf{LOC}) = L(\mathbf{SUP})$. Hence $\xi(b, \sigma)!$ and $\sigma \in E(b)$, i.e., $\sigma \notin D^k(b)$. Last, an argument proving the DES-isomorphism between $\mathbf{J}^k$ and $\mathbf{LOC}^k$ can be found in [23, Appendix P-2]. ∎

To summarize, every set of control covers generates a solution to the distributed control problem (⋇) (Proposition 1); and every normal solution to (⋇) must be induced from some set of control covers under the condition of DES-isomorphism (Theorem 1). In particular, a set of state-minimal normal generators can be induced from a set of suitable control covers. However, such a set is in general not unique, even up to DES-isomorphism. This conclusion accords with that for a state-minimal supervisor in supervisor reduction [23]. Notice that, while Theorem 1 is not required for the subsequent computation of control-equivalent generators, it provides important theoretical perspective (just as its precursor did in [23, Theorem 2.1]). Namely, it points to the universality of control covers as the mechanism of supervisor reduction/localization. And this underlines the fact that for computational tractability, as explained below, (general) covers are traded off for (simpler) congruences.

B. Localization Algorithm

It would be desirable to have an efficient algorithm that always computes a set of state-minimal normal generators, despite its non-uniqueness. Unfortunately, this minimal state problem is NP-hard [23, Section 3], and consequently we cannot expect a polynomial-time algorithm for it.

Nevertheless, a polynomial-time algorithm for supervisor reduction is known [23, Section 4.1]. The algorithm generates a control congruence, rather than a control cover, and empirical evidence was given to show that significant state size reduction can often be achieved. Therefore we employ this algorithm, suitably modified to work for supervisor localization, and call the altered version a *localization algorithm* (LA).

We sketch the idea of LA as follows. Given $\mathbf{SUP} = (X, \Sigma_c \dot{\cup} \Sigma_u, \_, \_, \_)$ with $X = \{x_0, \ldots, x_{n-1}\}$ and $\Sigma_c^k \subseteq \Sigma_c$, LA generates a control congruence $\mathcal{C}^k$ on X (with respect to Σ_c^k). LA initializes $\mathcal{C}^k$ to be the singleton partition on X, i.e.,

$$\mathcal{C}_{init}^k = \{[x] \subseteq X | [x] = \{x\}\}$$

where $[x]$ denotes the cell in $\mathcal{C}^k$ to which x belongs. Then LA merges two cells $[x_1]$ and $[x_2]$ into one if for every $x_1' \in [x_1]$ and $x_2' \in [x_2]$, x_1' and x_2', as well as all their corresponding future states reachable by identical strings, are control consistent. This *mergibility* condition is checked by lines 13 and 18 in the pseudocode below: line 13 checks control consistency for the current state pair (x_1', x_2') and line 18 recursively checks consistency for all their related future states. Throughout, in order to generate a control congruence, it is crucial to prevent states from being shared by more than one cell. This is achieved by inserting in LA three filters—at lines 3, 5, and 17—to eliminate redundant mergibility tests as well as element overlapping in $\mathcal{C}^k$. LA loops until all the states in X are checked.

1: **procedure** MAIN()

2: **for** $i := 0$ to $n - 2$ **do**

3: **if** $i > \min\{m | x_m \in [x_i]\}$ **then continue**;

4: **for** $j := i + 1$ to $n - 1$ **do**

5: **if** $j > \min\{m | x_m \in [x_j]\}$ **then continue**;

6: wait $= \emptyset$;

7: **if** $\text{Check\_Merge}(x_i, x_j, \text{wait}, i) = true$ **then**

8: $\mathcal{C}^k := \left\{[x] \cup \bigcup_{x' : \{(x,x'),(x',x)\} \cap \text{wait} \neq \emptyset} [x'] \mid [x], [x'] \in \mathcal{C}^k\right\}$;

9: **function** CHECK_MERGE(x_i, x_j, wait, cnode)

10: **for** each $x_p \in [x_i] \cup \bigcup_{x : \{(x,x_i),(x_i,x)\} \cap \text{wait} \neq \emptyset} [x]$ **do**

11: **for** each $x_q \in [x_j] \cup \bigcup_{x : \{(x,x_j),(x_j,x)\} \cap \text{wait} \neq \emptyset} [x]$ **do**

12: **if** $\{(x_p, x_q), (x_q, x_p)\} \cap \text{wait} \neq \emptyset$ **then continue**;

13: **if** $(x_p, x_q) \notin \mathcal{R}^k$ **then return** *false*;

14: wait $=$ wait $\cup \{(x_p, x_q)\}$;

15: **for** each $\sigma \in \Sigma$ with $\xi(x_p, \sigma)!$, $\xi(x_q, \sigma)!$

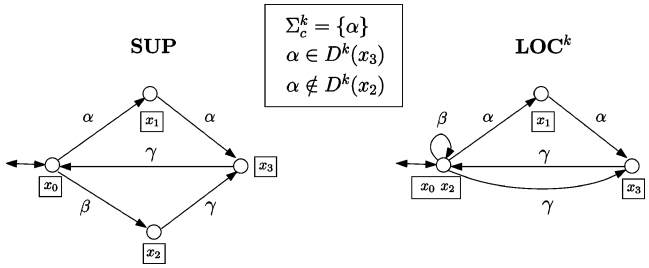


Fig. 2. Example: localization algorithm.

16: **if** $[\xi(x_p,\sigma)] = [\xi(x_q,\sigma)]$ **or** $\{(\xi(x_p,\sigma),\xi(x_q,\sigma)),(\xi(x_p,\sigma),\xi(x_q,\sigma))\} \cap \text{wait} \neq \emptyset$ **then continue**;

17: **if** $\min\{m|x_m \in [\xi(x_p,\sigma)]\} < \text{cnode}$ **or** $\min\{m|x_m \in [\xi(x_q,\sigma)]\} < \text{cnode}$ **then return** *false*;

18: **if** $\text{Check\_Merge}(\xi(x_p,\sigma),\xi(x_q,\sigma),\text{wait},\text{cnode}) = false$ **then return** *false*;

19: **return** *true*;

Remark 1: LA preserves all computational properties of the reduction algorithm in [23]. Namely, LA terminates, generates a control congruence, and has time complexity $O(n^4)$, where n is the state size of **SUP**.

Example 1: This example, displayed in Fig. 2, is a simple illustration of LA.

Notation: $\mathcal{C}_{init}^k$ denotes the initial control cover with respect to Σ_c^k, and $\mathcal{C}_i^k$ $(i = 1, 2, 3)$ the resulting control cover of the ith iteration of main().

(1) Initially, $\mathcal{C}_{init}^k = \{[x_0],[x_1],[x_2],[x_3]\}$

(2) (x_0, x_1) cannot be merged: they pass line 13 because $(x_0, x_1) \in \mathcal{R}^k$, but they fail at line 18 for $(\xi(x_0,\alpha),\xi(x_1,\alpha)) \notin \mathcal{R}^k$; (x_0, x_2) can be merged: they pass line 13 because $(x_0, x_2) \in \mathcal{R}^k$, and they trivially pass line 18 since there is no common event defined on them, so that no further control consistency needs to be verified; (x_0, x_3) cannot be merged: they fail at line 13, for $(x_0, x_3) \notin \mathcal{R}^k$.
So, $\mathcal{C}_1^k = \{[x_0, x_2],[x_1],[x_3]\}$.

(3) (x_1, x_2) cannot be merged: they cannot pass line 5, because x_2 and x_0 are now in the same cell and $2 > 0$; (x_1, x_3) cannot be merged: they failed at line 13, since $(x_1, x_3) \notin \mathcal{R}^k$.
Thus, $\mathcal{C}_2^k = \{[x_0, x_2],[x_1],[x_3]\}$.

(4) (x_2, x_3) cannot be merged: they failed at line 3 for, again, x_2 and x_0 are now in the same cell and $2 > 0$.
Finally, $\mathcal{C}_3^k = \{[x_0, x_2],[x_1],[x_3]\}$, and the induced generator $\mathbf{LOC}^k$ (unique in this case) is displayed on the right of Fig. 2.

♦

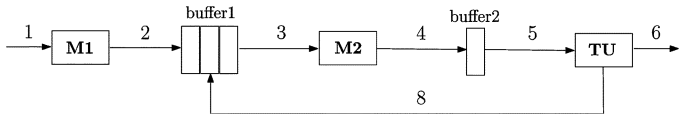


Fig. 3. Transfer line: system configuration.

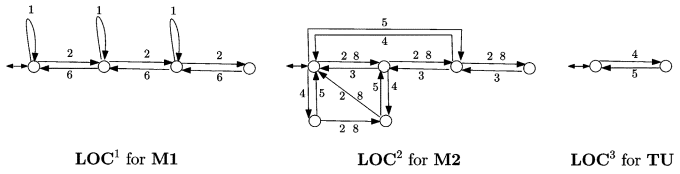


Fig. 4. Transfer line: local controllers.

C. Example: Distributed Control of Transfer Line

The transfer line system [1, Section 4.6], displayed in Fig. 3[3], consists of two machines **M1, M2** followed by a test unit **TU**; these agents are linked by two buffers with capacities of three slots and one slot, respectively. We model **M1, M2**, and **TU** as the plant to be controlled; the (safety) specification is to protect the two buffers against overflow and underflow. The distributed control objective is to design for each agent a local controller—but with no external supervisor.

We first build the monolithic optimal nonblocking supervisor, which has 28 states. Then we employ the localization algorithm to compute for each agent a local controller from the monolithic supervisor. The resultant controllers are displayed in Fig. 4 (for clarity extraneous selfloops are omitted), having 4, 6, and 2 states, respectively. The desired control equivalence between these local controllers and the monolithic supervisor is verified in TCT [25], by confirming $\mathbf{isomorph}\,(\mathbf{meet}(\mathbf{LOC}^1, \mathbf{LOC}^2, \mathbf{LOC}^3, \mathbf{G}), \mathbf{SUP}) = true$.[4] With these individual controllers, we can account for the local strategies of each agent. Machine **M1** with $\mathbf{LOC}^1$, controlling event 1, ensures that no more than three workpieces can be processed simultaneously in the system, i.e., prevents choking in the material feedback loop; but to achieve this goal, an external event 6 from **TU** has to be observed. Machine **M2** with $\mathbf{LOC}^2$, controlling event 3, guarantees the safety of both buffers at the same time; again, external events 2, 5, and 8 have to be observed. Notice that the observed event 5, being a controllable event of **TU**, cannot be disabled by **M2**. Lastly, **TU** with $\mathbf{LOC}^3$, controlling event 5 and observing an external event 4, is responsible only for the safety of buffer2.

D. Boundary Cases

We identify two boundary cases of supervisor localization which indicate, as a property of the localization problem itself, an extreme degree of easiness or hardness, respectively.

[3]Throughout this paper we label controllable events by odd numbers, and uncontrollable by even.

[4]For TCT procedures **meet** and **isomorph**, see [1, p. xii and p. xv respectively]. Essentially **meet** computes the reachable product of DES, and **isomorph** tests the identity of two DES under suitable recoding of states.

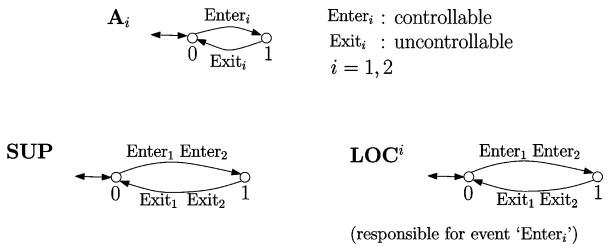


Fig. 5. Example: non-localizable.

1) Fully-Localizable: This case is the easy situation where component agents are completely decoupled: each agent works independently without any interaction through shared events.

Given a plant $\mathbf{G}$ (over Σ) composed of agents over disjoint alphabets Σ^k, define natural projections $P_k : \Sigma^* \to (\Sigma^k)^*$ ($k \in \mathcal{K}$). For an imposed specification $E = \|\{E_p | p \in \mathcal{P}\}$ let **SUP** be the corresponding monolithic supervisor.

Definition 5: $\mathbf{SUP}$ is *fully-localizable* if there exists a set of local controllers $\{\mathbf{LOC}^k | k \in \mathcal{K}\}$ which is control equivalent to **SUP** such that for every $k \in \mathcal{K}$, $L(\mathbf{LOC}^k) = P_k^{-1}(L^k)$ for some $L^k \subseteq (\Sigma^k)^*$. ◊

A sufficient condition that ensures full-localizability is the following.

Proposition 2: If for all $p \in \mathcal{P}$ there is $k \in \mathcal{K}$ such that $E_p \subseteq (\Sigma^k)^*$, then **SUP** is fully-localizable.

Proof: Follows from the assumption that $\Sigma^k (k \in \mathcal{K})$ are pairwise disjoint and Definition 5. ■

The assumption of Proposition 2 says that every component specification is imposed exclusively on some component agent. In that case, local controllers can be obtained locally without going through the top-down localization procedure. Similar results in the modular control context can be found in the literature (e.g., [5]).

2) Non-Localizable: The other extreme of the localization problem is the hard case where component agents are coupled so tightly that each one has to be globally aware.

Example 2: In Fig. 5, two agents $\mathbf{A}_i$ $(i = 1, 2)$ share a common resource that is not allowed to be occupied simultaneously. It is easy to see that $\mathbf{SUP}$ is a monolithic supervisor which enforces the mutual exclusion specification. Then by applying the localization algorithm to $\mathbf{SUP}$, we generate for agent $\mathbf{A}_i$ a local controller $\mathbf{LOC}^i$. However, both local controllers are nothing but the same as $\mathbf{SUP}$; namely, our supervisor localization accomplished nothing useful. ♦

In general, we aim to find conditions that can identify the situation where localization fails to achieve a truly local result. In that case we need only make copies of $\mathbf{SUP}$ for the relevant agents.

Definition 6: Let $\mathbf{MLOC}^k$ be a state-minimal local controller for agent $\mathbf{G}^k$ (defined over $\Sigma^k \subseteq \Sigma$). $\mathbf{SUP}$ is *non-localizable* (with respect to Σ_c^k) if $|\mathbf{SUP}| = |\mathbf{MLOC}^k|$. ◊

First note that $|\mathbf{SUP}| = |\mathbf{MLOC}^k|$ implies that $\mathbf{SUP} = \mathbf{MLOC}^k$. This is because if **SUP** is already state-minimal, then no more pairs of states in **SUP** can be merged, which in turn implies that the transition structure will remain the same.

By Theorem 1, $\mathbf{MLOC}^k$ is induced from some control cover, denoted $\mathcal{C}^k$. We proceed to determine the number of cells in $\mathcal{C}^k$. Given $\mathbf{SUP} = (X, \Sigma, \xi, x_0, X_m)$, by the definition of control cover two states $x, x' \in X$ that belong to an identical cell must satisfy both conditions

(1) $(x, x') \in \mathcal{R}^k$;
(2) $(\forall\, s \in \Sigma^*)\ \xi(x, s)!\ \&\ \xi(x', s)! \Rightarrow (\xi(x, s), \xi(x', s)) \in \mathcal{R}^k$.

Negating (1) and (2), we get

(1) $(x, x') \notin \mathcal{R}^k$;
(2) $(\exists\, s \in \Sigma^*)\ \xi(x, s)!\ \&\ \xi(x', s)!\ \&\ (\xi(x, s), \xi(x', s)) \notin \mathcal{R}^k$.

Hence, two states x, x' belong to different cells of $\mathcal{C}^k$ if and only if either (3) or (4) holds. Let

$$\Omega_{\mathcal{C}^k} := \max\Big\{n \big| (\exists\, X' \subseteq X)\, |X'| = n\ \&\ (\forall\, x, x' \in X') x \neq x' \Rightarrow (3) \text{ or } (4)\Big\}.$$

The above discussion has proved the following fact.

Proposition 3: $|\mathbf{MLOC}^k| = \Omega_{\mathcal{C}^k}$. ■

Now a necessary and sufficient condition for non-localizability is immediate.

Proposition 4: **SUP** is non-localizable (with respect to $\Sigma_c^k \subseteq \Sigma$) if and only if $|\mathbf{SUP}| = \Omega_{\mathcal{C}^k}$.

Proof: Follows directly from Definition 6 and Proposition 3. ■

In fact the above condition is hardly more than a restatement of the definition of non-localizability. We have still said nothing about how to check whether or not the condition holds. Nevertheless, a slight modification of $\Omega_{\mathcal{C}^k}$ will lead to a computationally verifiable sufficient condition for non-localizability.

Consider

$$\Omega_k := \max\{n | (\exists\, X' \subseteq X)\, |X'| = n\ \&\ (\forall\, x, x' \in X') x \neq x' \Rightarrow (x, x') \notin \mathcal{R}^k\}.$$

That is, we disregard those cases where control inconsistency is caused by related future states. It should be obvious that $\Omega_k \leq \Omega_{\mathcal{C}^k}$. More importantly, if we construct an undirected graph $G = (V, E)$ with $V = X$ and $E = \{(x, x') | (x, x') \notin \mathcal{R}^k\}$, then calculating Ω_k amounts to finding the maximum clique in G. Although the maximum clique problem is a well-known NP-complete problem, there exist efficient algorithms that compute suboptimal solutions [26]. In particular, the implemented polynomial-time algorithm that computes *lower bound estimate (lbe))* in [23, Section 4.2] can be directly employed for our purpose. Let us denote by lbe_k the outcome of the suboptimal algorithm with respect to $\mathcal{R}^k$. Thus we have $lbe_k \leq \Omega_k \leq \Omega_{\mathcal{C}^k} \leq |\mathbf{SUP}|$, which gives rise to the following result.

Proposition 5: If $|\mathbf{SUP}| = lbe_k$, then **SUP** is non-localizable (with respect to $\Sigma_c^k \subseteq \Sigma$).

Proof: $|\mathbf{SUP}| = lbe_k$ implies that $|\mathbf{SUP}| = \Omega_{\mathcal{C}^k}$, and consequently $|\mathbf{SUP}| = |\mathbf{MLOC}^k|$ by Proposition 3.

This condition is not necessary for non-localizability. If we obtain $|\mathbf{SUP}| > lbe_k$, lbe_k tells us little about localizability and can only serve as a conservative lower bound estimate indicating how much localization might (conceivably) be achieved. If, however, $|\mathbf{SUP}| = lbe_k$ does hold, then the problem instance admits no useful solution, and we can avoid wasting further computational effort. Continuing Example 2, and applying the

adopted algorithm from [23], we obtain $lbe_i = 2 = |\mathbf{SUP}|$ $(i = 1, 2)$. Hence **SUP** is non-localizable for either of the two agents, and we then simply assign the agents with copies of **SUP** as their local controllers.[5]

IV. Solution for Large-Scale Systems

We move on to solve the distributed control problem $(\divideontimes)$ for large-scale systems; our approach combines supervisor localization with an efficient modular control theory [15], [27].

A. Preliminaries

We introduce two concepts that are essential to guarantee optimality and nonblockingness of the modular control design we adopt. Consider a generator $\mathbf{G}$ defined over Σ, and a natural projection $P : \Sigma^* \to \Sigma_o^*$ for some observable event subset $\Sigma_o \subseteq \Sigma$. We say P is *output control consistent* (OCC) [15] for $L(\mathbf{G})$ if for every string $s \in L(\mathbf{G})$ of the form

$$s = s'\sigma_1 \cdots \sigma_k, \quad k \geq 1$$

where s' is either the empty string or terminates with an event in Σ_o, the following holds:

$$\begin{aligned}&((\forall\, i \in [1, k-1])\, \sigma_i \in \Sigma - \Sigma_o)\ \&\ \sigma_k \in \Sigma_o \cap \Sigma_u \\ &\quad \Rightarrow ((\forall\, j \in [1, k])\, \sigma_j \in \Sigma_u)\,.\end{aligned}$$

Informally, whenever σ_k is observable and uncontrollable, its immediately preceding unobservable events must all be uncontrollable. By this definition, ensuring that P is OCC amounts to ensuring that for each uncontrollable event σ in Σ_o, σ's nearest upstream controllable events are also in Σ_o.

The second concept is that of *natural observer*: P is called an $L_m(\mathbf{G})$-*observer* [15] if

$$\begin{aligned}&(\forall\, s \in L(\mathbf{G}), \forall\, t_o \in \Sigma_o^*)\ (P_o s)t_o \in P_o L_m(\mathbf{G}) \\ &\quad \Rightarrow (\exists\, t \in \Sigma^*)\, P_o t = t_o\ \&\ st \in L_m(\mathbf{G}).\end{aligned}$$

Informally, whenever Ps can be extended to $PL_m(\mathbf{G})$ by an observable string t_o, the underlying string s can be extended to $L_m(\mathbf{G})$ by a string t with $Pt = t_o$. In case P does not enjoy the observer property, we consider adding a minimal number of events to Σ_o so that the augmented observable subset does define an $L_m(\mathbf{G})$-observer. This is the *minimal extension problem* addressed in [27, Chapter 5], [28]. There, it was proved that a unique extension through adding a minimal number of events generally does not exist for $L_m(\mathbf{G})$-observers, and even finding some minimal extension is in fact NP-hard. Nevertheless, a polynomial-time algorithm is presented which accomplishes a reasonable extension that achieves the observer property; of course this extension need not always be minimal. Henceforth we refer to this algorithm as the *minimal extension* (MX) algorithm.

[5]Recently the paper [19] came to our attention, addressing "multi-agent coordination planning" in the SCT framework; the objective is to synthesize "coordination modules" for individual agents. The proposed synthesis procedure, however, amounts simply to making copies of the (reduced) monolithic supervisor for each agent, with some corresponding, extraneous selfloops removed. By contrast, we do so only when the supervisor is identified as non-localizable; otherwise, our localization procedure exploits supervisor reduction and achieves a truly local result. Furthermore, [19] does not present an effective approach to large systems, while we propose in Section IV a solution procedure that combines localization with modular control theories.

B. Decomposition-Aggregation Procedure

We present a *decomposition-aggregation procedure* (DAP) as a solution to the distributed control problem $(\divideontimes)$ for large-scale systems. Recall that we start with the following:

- A plant $\mathbf{G}$ (defined over Σ) consisting of agents $\mathbf{G}^k$ defined over disjoint Σ^k $(k \in \mathcal{K})$; the closed and marked behaviors of $\mathbf{G}^k$ are L_k and $L_{m,k}$, respectively.
- A specification E decomposable into $E_p \subseteq \Sigma_{e,p}^*$ $(p \in \mathcal{P})$. So E is defined over $\Sigma_e := \bigcup\{\Sigma_{e,p} | p \in \mathcal{P}\}$, with the corresponding natural projection $P_e : \Sigma^* \to \Sigma_e^*$.

1) Plant Model Abstraction: Part of the plant dynamics that is unrelated to the imposed specification may be concealed. By hiding irrelevant transitions, we can simplify the models of component agents. The procedure is as follows [27, Ch. 4].

(i) For every $k \in \mathcal{K}$, check if $P_e|_{(\Sigma^k)^*} : (\Sigma^k)^* \to \left(\Sigma_e \cap \Sigma^k\right)^*$ is OCC for L_k; if not, for each $\sigma \in \Sigma_e \cap \Sigma_u^k$ add its nearest upstream controllable events to Σ_e. Denote the augmented alphabet by Σ_e', and let $P_e' : \Sigma^* \to (\Sigma_e')^*$.

(ii) For every $k \in \mathcal{K}$, check if $P_e'|_{(\Sigma^k)^*} : (\Sigma^k)^* \to \left(\Sigma_e' \cap \Sigma^k\right)^*$ is an $L_{m,k}$-observer. If yes, go to (iii); otherwise, employ the MX algorithm to compute a reasonable extension of $\Sigma_e' \cap \Sigma^k$ that does define an $L_{m,k}$-observer. Denote the extended alphabet again by Σ_e', and the corresponding natural projection again by P_e'. Return to (i).

(iii) Compute model abstractions for each agent, denoted by $(\mathbf{G}^k)'$, with closed and marked languages

$$L_k' := P_e'|_{(\Sigma^k)^*}(L_k) \text{ and } L_{m,k}' := P_e'|_{(\Sigma^k)^*}(L_{m,k}).$$

Note that abstractions $(\mathbf{G}^k)'$ are defined over disjoint alphabets $(\Sigma^k)' := \Sigma_e' \cap \Sigma^k$.

2) Decentralized Supervisor Synthesis: The system now consists of agent model abstractions $(\mathbf{G}^k)'$ $(k \in \mathcal{K})$ and component specifications $E_p \subseteq \Sigma_{e,p}^*$ $(p \in \mathcal{P})$. Since each specification E_p may impose constraints on only a subset of agent abstractions, a decentralized supervisor with respect to E_p, denoted by $\mathbf{SUP}_p$, can be obtained with only those relevant abstractions. Specifically, we associate with each E_p its *event-coupled* agent abstractions: those sharing events with E_p (i.e., $(\Sigma^k)' \cap \Sigma_{e,p} \neq \emptyset$); and then, we synthesize a corresponding optimal and nonblocking decentralized supervisor $\mathbf{SUP}_p$ based on [15, Theorem 2], [27, Theorem 4.2].

3) Subsystem Decomposition and Coordination: After synthesizing decentralized supervisors, we view the whole system as comprised of a set of modules $\{\mathcal{M}_p \mid p \in \mathcal{P}\}$, each $\mathcal{M}_p$ consisting of a decentralized supervisor $\mathbf{SUP}_p$ with associated agent model abstractions. In this step, we decompose the overall system into small-scale subsystems, through grouping these modules based on their interconnection dependencies (e.g., event-coupling). If the modules admit certain special structures, *control-flow net* [29] and *process communication graph* [30] are two effective approaches for subsystem decomposition.

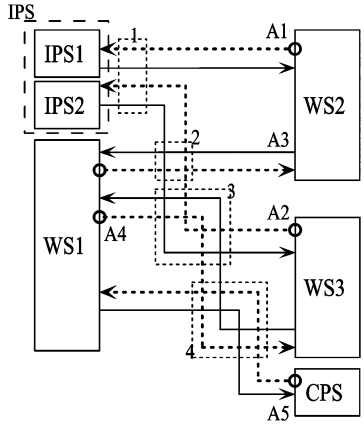


Fig. 6. AGV: system configuration.

Having obtained a group of small subsystems, we verify the nonblocking property for each of them[6]. If a subsystem happens to be blocking, we design a *coordinator*[7] to resolve the conflict by employing a method proposed in [15, Proposition 7 and Theorem 4], [27, Proposition 4.7 and Theorem 4.4].

4) Subsystem Model Abstraction: After ensuring nonblockingness within each subsystem, we need to verify the nonconflicting property among these subsystems. Directly verifying this property requires expensive computation; instead, we again bring in the model abstraction technique to simplify every subsystem, and check the nonconflictingness on the abstracted level. The procedure is analogous to that of Step 1) Plant Model Abstraction, above.

(i) Determine the shared event set, denoted by Σ_{sub}, of these subsystems. Let $P_{sub} : (\Sigma_e')^* \rightarrow \Sigma_{sub}^*$ be the corresponding natural projection.

(ii) For every subsystem check if the corresponding restriction of P_{sub} is OCC; if not, for each $\sigma \in \Sigma_{sub} \cap \Sigma_u$ add its nearest upstream controllable events to Σ_{sub}. Denote the augmented alphabet by Σ'_{sub}, and let $P'_{sub} : (\Sigma_e')^* \rightarrow (\Sigma'_{sub})^*$.

(iii) For every subsystem check if the corresponding restriction of P'_{sub} is an observer. If yes, go to (iv); otherwise, employ the MX algorithm to compute a reasonable extension of Σ'_{sub} that does define an observer for every subsystem. Denote the extended alphabet again by Σ'_{sub}, and the corresponding natural projection again by P'_{sub}. Return to (ii).

(iv) Compute model abstractions for each subsystem with P'_{sub}.

5) Abstracted Subsystem Decomposition and Coordination: This step is analogous to Step 3, but for subsystem model abstractions instead of modules. Concretely, we organize subsystem abstractions into groups according to their interconnection dependencies (e.g., event-coupling). Again, control-flow net and process communication graph may be effective tools if certain special structure is present. Then for each group, we check if the included subsystem abstractions are nonconflicting; and if not, design a coordinator to resolve the conflict.

6) Higher-Level Abstraction: Repeat Steps 4 and 5 until there remains a single group of subsystem abstractions in Step 5.

The modular supervisory control design terminates at Step 6; we have obtained a hierarchy of decentralized supervisors and coordinators. Specifically, Step 2 gives a set of decentralized supervisors $\{\mathbf{SUP}_p \mid p \in \mathcal{P}\}$; and Steps 3 to 6 iteratively generate a set of coordinators, denoted by $\{\mathbf{CO}_q \mid q \in \mathcal{Q}\}$ ($\mathcal{Q}$ an index set).

7) Decentralized Supervisors and Coordinators Localization: We now apply the supervisor localization algorithm to localize each of these decentralized supervisors $\mathbf{SUP}_p$ $(p \in \mathcal{P})$ and coordinators $\mathbf{CO}_q$ $(q \in \mathcal{Q})$ to local controllers for their relevant agents. To this end, we bring in a criterion to determine if an agent $\mathbf{G}^k$ $(k \in \mathcal{K})$ is related to $\mathbf{SUP}_p$ or $\mathbf{CO}_q$. For $\mathbf{SUP}_p = (X_p, \_, \_, \_, \_)$ recall that the function D^k associates each state $x \in X_p$ with the subset of controllable events of $\mathbf{G}^k$ that must be disabled at x. We say $\mathbf{G}^k$ is *control-coupled* to $\mathbf{SUP}_p$ if

$$(\exists\, x \in X_p)\; D^k(x) \neq \emptyset.$$

In other words, $\mathbf{SUP}_p$ disables some controllable events of $\mathbf{G}^k$[8]. Thus the set of agents that are control-coupled to $\mathbf{SUP}_p$ is $\{\mathbf{G}^k \mid (\exists\, x \in X_p)\; D^k(x) \neq \emptyset,\; k \in \mathcal{K}\}$; and we localize $\mathbf{SUP}_p$ to local controllers only for this set. Similarly, we localize $\mathbf{CO}_q = (X_q, \_, \_, \_, \_)$ to local controllers only for the set $\{\mathbf{G}^k \mid (\exists\, x \in X_q)\; D^k(x) \neq \emptyset,\; k \in \mathcal{K}\}$.

Theorem 2: DAP solves the distributed control problem $(\divideontimes)$.

Proof: The first six steps of DAP generate a hierarchy of decentralized supervisors $\{\mathbf{SUP}_p \mid p \in \mathcal{P}\}$ and coordinators $\{\mathbf{CO}_q \mid q \in \mathcal{Q}\}$. It follows from [15, Theorem 4], [27, Theorem 4.4] that

$$\begin{aligned} &L(\|\{\mathbf{SUP}_p \mid p \in \mathcal{P}\}) \,||\, L(\|\{\mathbf{CO}_q \mid q \in \mathcal{Q}\}) \\ &\quad = L(\mathbf{SUP}), \\ &L_m(\|\{\mathbf{SUP}_p \mid p \in \mathcal{P}\}) \,||\, L_m(\{\|\mathbf{CO}_q \mid q \in \mathcal{Q}\}) \\ &\quad = L_m(\mathbf{SUP}). \end{aligned}$$

In Step 7, each decentralized supervisor $\mathbf{SUP}_p = (X_p, \_, \_, \_, \_)$ $(p \in \mathcal{P})$ is decomposed into a set of local controllers, one for each agent in $\{\mathbf{G}^k \mid (\exists\, x \in X_p)\; D^k(x) \neq \emptyset, k \in \mathcal{K}\}$. We denote this set of local controllers by $\left\{\mathbf{LOC}_p^k \mid (\exists\, x \in X_p)\; D^k(x) \neq \emptyset, k \in \mathcal{K}\right\}$. Let

$$L(\mathbf{LOC}_p) = \bigcap \left\{ L\left(\mathbf{LOC}_p^k\right) \mid (\exists\, x \in X_p)\; D^k(x) \neq \emptyset,\; k \in \mathcal{K} \right\}$$

$$L_m(\mathbf{LOC}_p) = \bigcap \left\{ L_m\left(\mathbf{LOC}_p^k\right) \mid (\exists\, x \in X_p)\; D^k(x) \neq \emptyset,\; k \in \mathcal{K} \right\}.$$

[6]We use TCT [25] procedure **nonconflicting** for this verification.

[7]A coordinator is an automaton that does not directly enforce a safety specification, but only resolves conflict among decentralized supervisors. In other words, a coordinator enforces only a nonblocking specification.

[8]The control coupling relation can be determined by inspecting the control data table generated by the TCT procedure **condat** [25].

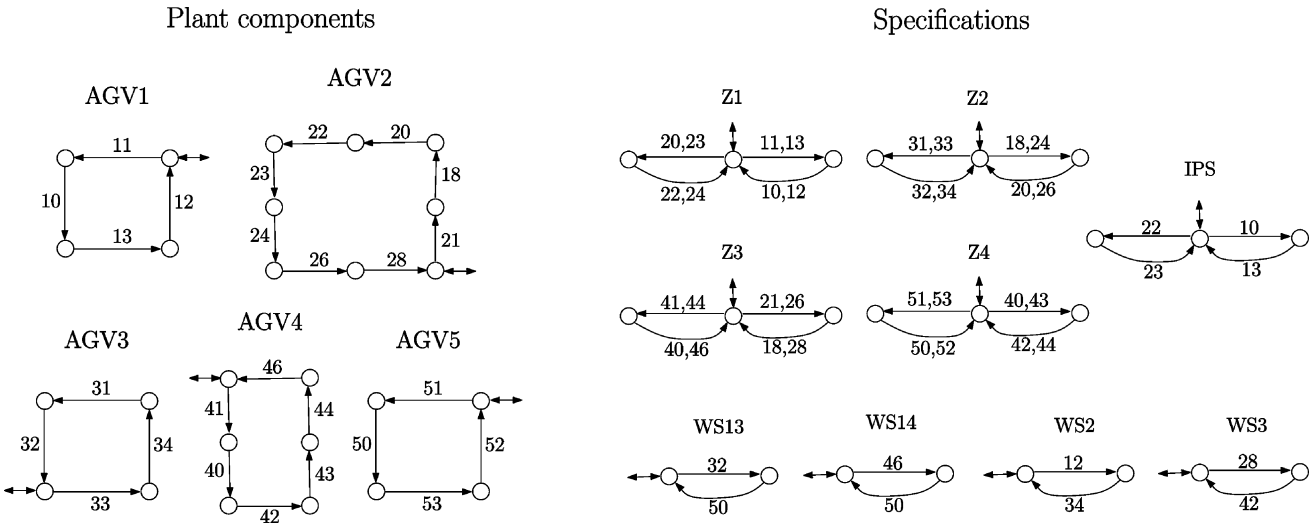


Fig. 7. Generators of plant components and specifications.

Then by Proposition 1 we have

$$L(\mathbf{LOC}_p) \cap L(\mathbf{G}) = L(\mathbf{SUP}_p)$$
$$L_m(\mathbf{LOC}_p) \cap L_m(\mathbf{G}) = L_m(\mathbf{SUP}_p).$$

Thus

$$L(||\{\mathbf{LOC}_p \mid p \in \mathcal{P}\}) \cap L(\mathbf{G}) = L(||\{\mathbf{SUP}_p \mid p \in \mathcal{P}\})$$
$$L_m(||\{\mathbf{LOC}_p \mid p \in \mathcal{P}\}) \cap L_m(\mathbf{G}) = L_m(||\{\mathbf{SUP}_p \mid p \in \mathcal{P}\}).$$

Also for each coordinator $\mathbf{CO}_q$ $(q \in \mathcal{Q})$, we have by the same derivation above

$$L(||\{\mathbf{LOC}_q \mid q \in \mathcal{Q}\}) \cap L(\mathbf{G}) = L(||\{\mathbf{CO}_q \mid q \in \mathcal{Q}\})$$
$$L_m(||\{\mathbf{LOC}_q \mid q \in \mathcal{Q}\}) \cap L_m(\mathbf{G}) = L_m(||\{\mathbf{CO}_q \mid q \in \mathcal{Q}\}).$$

Finally, letting

$$L(\mathbf{LOC}) = L(||\{\mathbf{LOC}_p \mid p \in \mathcal{P}\}) \,||\, L(||\{\mathbf{LOC}_q \mid q \in \mathcal{Q}\})$$
$$L_m(\mathbf{LOC}) = L_m(||\{\mathbf{LOC}_p \mid p \in \mathcal{P}\}) \,||\, L_m(||\{\mathbf{LOC}_q \mid q \in \mathcal{Q}\}).$$

we conclude

$$L(\mathbf{LOC}) \cap L(\mathbf{G}) = L(||\{\mathbf{SUP}_p | p \in \mathcal{P}\}) || L(\{||\mathbf{CO}_q | q \in \mathcal{Q}\}) = L(\mathbf{SUP})$$
$$L_m(\mathbf{LOC}) \cap L_m(\mathbf{G}) = L_m(||\{\mathbf{SUP}_p | p \in \mathcal{P}\}) || L_m(||\{\mathbf{CO}_q | q \in \mathcal{Q}\}) = L_m(\mathbf{SUP}).$$

■

C. Example: Distributed Control of an AGV System

We apply the decomposition-aggregation procedure to solve the distributed control problem of automated guided vehicles (AGVs) serving a manufacturing workcell, in the version of [1, Section 4.7], originally adapted from [31].

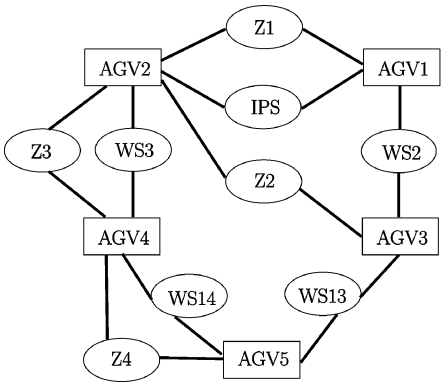
As displayed in Fig. 6, the workcell consists of two input parts stations IPS1, IPS2 for parts of types 1, 2; three workstations WS1, WS2, WS3; and one completed parts station CPS. A team of five independent $AGVs - AGV1, \ldots, AGV5$—travel in fixed criss-crossing routes, loading/unloading and transporting parts in the cell. We model the AGV system as the plant to be controlled, on which three types of control specifications are imposed: the mutual exclusion (i.e., single occupancy) of shared zones, the capacity limit of workstations, and the mutual exclusion of the shared loading area of the input stations. The generator models of plant components and specifications are displayed in Fig. 7; readers are referred to [1, Section 4.7] for the detailed interpretation of events. While the centralized approach generates a monolithic optimal nonblocking supervisor of 4406 states [1], our distributed control objective is to design for each AGV a set of local strategies which as a whole realize performance identical to that achieved by the monolithic supervisor.

1) Plant Model Abstraction: Let Σ and Σ_e denote the alphabets on which the overall plant and the overall specification are defined. One can verify that $\Sigma = \Sigma_e$ in this example. Namely, all of the plant dynamics are related to the subsequent synthesis, and therefore no plant model can be simplified in this step[9].

2) Decentralized Supervisor Synthesis: We group for each specification its event-coupled AGVs. The grouping is displayed on the left of Fig. 8, with solid lines denoting event-coupling. Then we synthesize a decentralized supervisor with respect to each specification, as shown on the right of Fig. 8. The state sizes of these supervisors are listed in Table I, where State # and Reduced State # are the results of TCT procedures **supcon** and **supreduce** [25], respectively.

3) Subsystem Decomposition and Coordination: We have nine decentralized supervisors, thus nine modules. The interconnection structure of these modules can be simplified by applying *control-flow net*. Specifically, the decentralized supervisors for the four zones—$\mathbf{Z1SUP}, \ldots, \mathbf{Z4SUP}$—are *harmless*

[9]See [32] for an example where $\Sigma \supsetneq \Sigma_e$ and part of the plant dynamics is concealed.



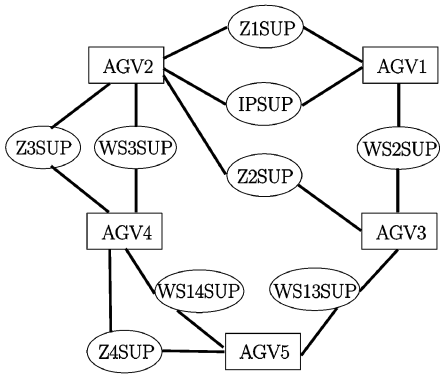


Fig. 8. Decentralized supervisor synthesis (ovals denote linking specifications).

TABLE I
STATE SIZES OF DECENTRALIZED SUPERVISORS

Supervisor	State #	Reduced State #
Z1SUP	24	2
Z2SUP	24	2
Z3SUP	36	2
Z4SUP	18	2
WS13SUP	24	2
WS14SUP	34	2
WS2SUP	24	2
WS3SUP	62	2
IPSUP	24	2

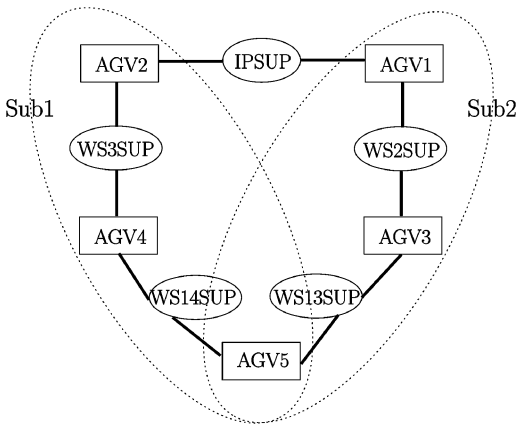


Fig. 9. Subsystem decomposition and coordination.

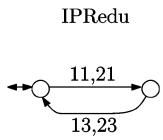


Fig. 10. Reduced generator model of decentralized supervisor $\mathbf{IPSUP}$.

TABLE II
STATE SIZES OF MODEL ABSTRACTIONS

	Sub1	**Sub1′**	**Sub2**	**Sub2′**
State #	140	30	330	30

to the overall nonblocking property, and hence can be safely removed from the interconnection structure [29], [32].

The simplified interconnection structure is displayed in Fig. 9. There are two paths—**AGV1**, **WS2SUP**, **AGV3**, **WS13SUP** on the right and **AGV2**, **WS3SUP**, **AGV4**, **WS14SUP** on the left—that process workpieces of types 1 and 2, respectively. Thus the overall system is naturally decomposed into two subsystems, as shown by dotted ovals in Fig. 9. It is further verified that both subsystems are nonblocking on their own.

4) Subsystem Model Abstraction: We must now verify the nonconflicting property among the two subsystems $\mathbf{Sub1}$, $\mathbf{Sub2}$, and the decentralized supervisor $\mathbf{IPSUP}$. First, we determine their shared event set, denoted by Σ_{sub}. Subsystems $\mathbf{Sub1}$ and $\mathbf{Sub2}$ share all of the events in $\mathbf{AGV5}$: 50, 51, 52, and 53. For the decentralized supervisor $\mathbf{IPSUP}$, we consider its reduced generator model (i.e., the result of the **supreduce** procedure), as displayed in Fig. 10. By inspection, $\mathbf{IPRedu}$ shares events 11, 13 with $\mathbf{Sub1}$, and 21, 23 with $\mathbf{Sub2}$. Thus we set $\Sigma_{sub} = \{11, 13, 21, 23, 50, 51, 52, 53\}$. It can then be verified that the corresponding natural projection $P_{sub} : \Sigma^* \to \Sigma^*_{sub}$ does enjoy both the OCC and observer properties. So with P_{sub}, we can compute the subsystem model abstractions, denoted by $\mathbf{Sub1}'$ and $\mathbf{Sub2}'$, with state sizes listed in Table II.

5) Abstracted Subsystem Decomposition and Coordination: As displayed in Fig. 11, we treat $\mathbf{Sub1}'$, $\mathbf{Sub2}'$, and $\mathbf{IPRedu}$ as a single group, and directly check the nonblocking property. This group turns out to be blocking; a coordinator **CO** is then designed to resolve the conflict, with its state size shown in Table III.

6) Higher-Level Abstraction: The modular supervisory control design terminates with the previous Step 5.

7) Decentralized Supervisors and Coordinators Localization: So far we have obtained a hierarchy of nine decentralized supervisors and one coordinator; their synchronized behavior is identical to the controlled behavior achieved by the monolithic supervisor [27]. In this last step we first determine the control-coupling relation between supervisors/coordinator and AGV agents; we do this by inspecting the corresponding **condat** tables. The result is displayed in Fig. 12, with dashed lines denoting the control-coupling. Notice that the coordinator **CO** is control-coupled only to $\mathbf{AGV1}$ and $\mathbf{AGV2}$. Along the

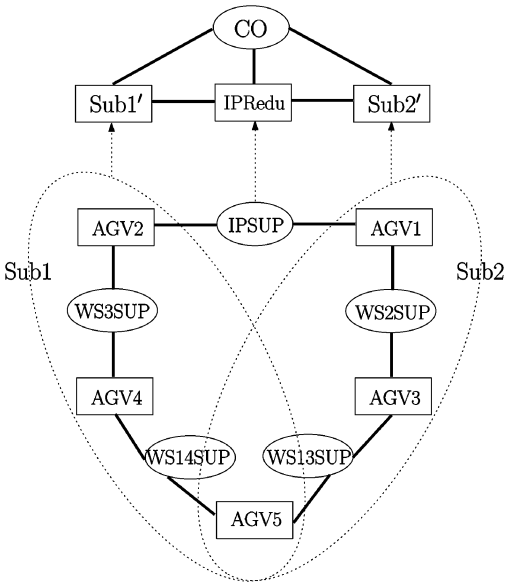


Fig. 11. Abstracted subsystem decomposition and coordination.

TABLE III
STATE SIZE OF COORDINATOR

	State #	Reduced State #
CO	165	7

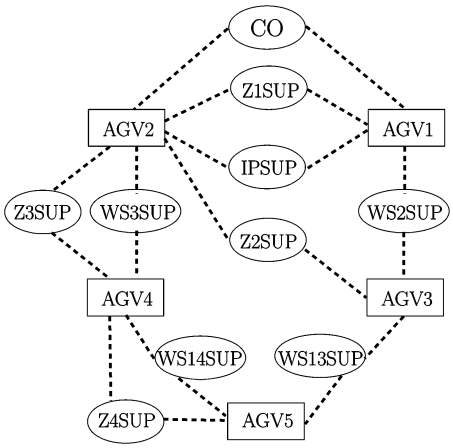


Fig. 12. Control-coupling relation between supervisors/coordinator and AGV agents.

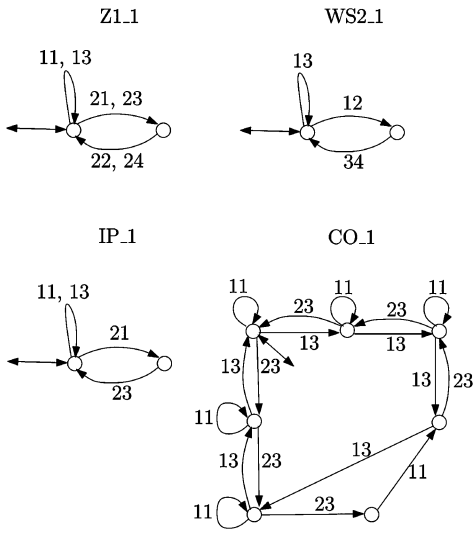


Fig. 13. Local controllers for AGV1.

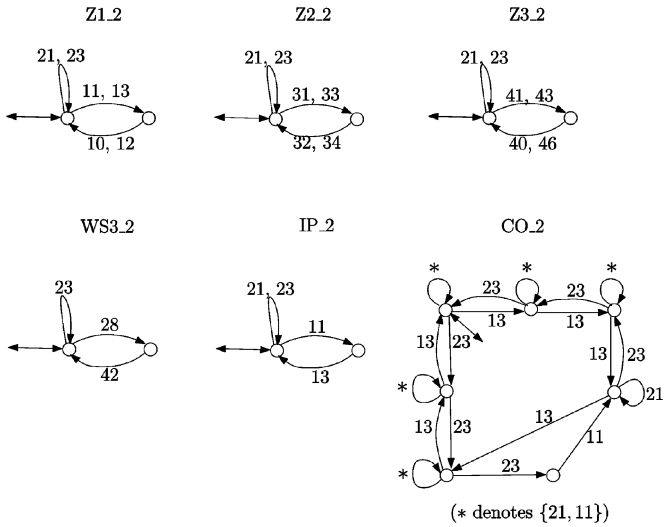


Fig. 14. Local controllers for AGV2.

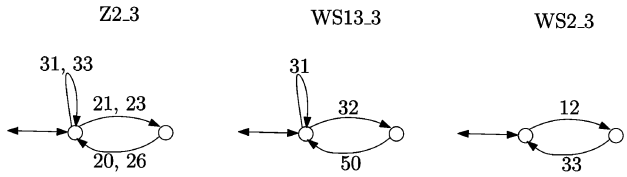


Fig. 15. Local controllers for AGV3.

dashed lines, we apply the supervisor localization algorithm. The state sizes of the resultant local controllers are listed in Table IV, and the generator models of each controller displayed in Figs. 13–17 (for clarity extraneous selfloops are omitted), grouped with respect to individual AGVs. Thus we have established a purely distributed control architecture, wherein each of the AGV robots pursues its independent lifestyle, while being coordinated implicitly with its fellows through their local shared observable events.

V. CONCLUSION

This paper has studied distributed control design for DES in the SCT framework. The central problem investigated is how to synthesize local controllers for individual agents such that these local controllers collectively realize controlled behavior identical to that achieved by optimal and nonblocking monolithic control. In the case of small-scale systems, a supervisor localization algorithm has been established that directly decomposes a monolithic supervisor into local controllers while preserving optimality and nonblockingness. For large systems a decomposition-aggregation procedure has been developed that first designs modular supervisors to achieve optimal nonblocking control, and then decomposes each modular supervisor into local controllers for the relevant agents.

TABLE IV
STATE SIZES OF LOCAL CONTROLLERS

Supervisor/Coordinator	AGV1 (#)	AGV2 (#)	AGV3 (#)	AGV4 (#)	AGV5 (#)
Z1SUP	Z1_1 (2)	Z1_2 (2)			
Z2SUP		Z2_2 (2)	Z2_3 (2)		
Z3SUP		Z3_2 (2)		Z3_4 (2)	
Z4SUP				Z4_4 (2)	Z4_5 (2)
WS13SUP			WS13_3 (2)		WS13_5 (2)
WS14SUP				WS14_4 (2)	WS14_5 (2)
WS2SUP	WS2_1 (2)		WS2_3 (2)		
WS3SUP		WS3_2 (2)		WS3_4 (2)	
IPSUP	IP_1 (2)	IP_2 (2)			
CO	CO_1 (7)	CO_2 (7)			

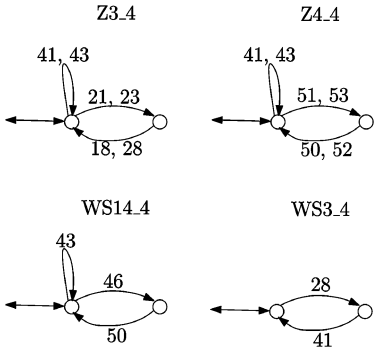


Fig. 16. Local controllers for AGV4.

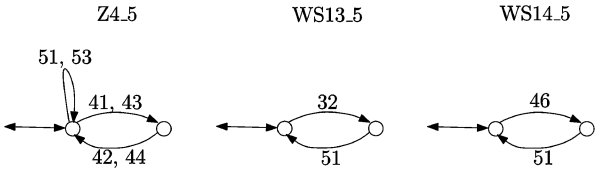


Fig. 17. Local controllers for AGV5.

Our investigation of distributed control design for DES has added "purely distributed" architecture to the family consisting of "monolithic" and "modular" architectures. This result gives rise to an interesting question: Given a specific system with a particular task, how to analyze quantitatively the tradeoffs among these three architectures, in such a way that one could decide which architecture was best suited to the task at hand? We consider such a "theory of architecture" to be an ultimate objective of SCT.

REFERENCES

[1] W. M. Wonham, "Supervisory Control of Discrete-Event Systems," Syst. Control Group, ECE Dept, Univ. Toronto, Toronto, ON, Canada, 2009 [Online]. Available: http://www.control.toronto.edu/DES

[2] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, 1987.

[3] W. M. Wonham and P. J. Ramadge, "On the supremal controllable sublanguage of a given language," *SIAM J. Control Optim.*, vol. 25, no. 3, pp. 637–659, 1987.

[4] F. Lin and W. M. Wonham, "Decentralized supervisory control of discrete-event systems," *Inform. Sci.*, vol. 44, pp. 199–224, 1988.

[5] Y. Willner and M. Heymann, "Supervisory control of concurrent discrete-event systems," *Int. J. Control*, vol. 54, no. 5, pp. 1143–1169, 1991.

[6] K. Rudie and W. M. Wonham, "Think globally, act locally: Decentralized supervisory control," *IEEE Trans. Autom. Control*, vol. 37, no. 11, pp. 1692–1708, Nov. 1992.

[7] T. S. Yoo and S. Lafortune, "A general architecture for decentralized supervisory control of discrete-event systems," *Discrete Event Dyna. Syst.: Theory Appl.*, vol. 12, no. 3, pp. 335–377, 2002.

[8] K. C. Wong and S. Lee, "Structural decentralized control of concurrent discrete-event systems," *Eur. J. Control*, vol. 8, pp. 477–491, 2002.

[9] K. Rudie, S. Lafortune, and F. Lin, "Minimal communication in a distributed discrete-event system," *IEEE Trans. Autom. Control*, vol. 48, no. 6, pp. 957–975, Jun. 2003.

[10] J. Komenda and J. H. van Schuppen, "Modular control of discrete-event systems with coalgebra," *IEEE Trans. Autom. Control*, vol. 53, no. 2, pp. 447–460, Mar. 2008.

[11] H. Zhong and W. M. Wonham, "On the consistency of hierarchical supervision in discrete-event systems," *IEEE Trans. Autom. Control*, vol. 35, no. 10, pp. 1125–1134, Oct. 1990.

[12] K. C. Wong and W. M. Wonham, "Hierarchical control of discrete-event systems," *Discrete Event Dyna. Syst.: Theory Appl.*, vol. 6, no. 3, pp. 241–273, 1996.

[13] K. C. Wong and W. M. Wonham, "Modular control and coordination of discrete-event systems," *Discrete Event Dyna. Syst.: Theory Appl.*, vol. 8, no. 3, pp. 247–297, 1998.

[14] K. Schmidt, T. Moor, and S. Perk, "Nonblocking hierarchical control of decentralized discrete event systems," *IEEE Trans. Autom. Control*, vol. 53, no. 10, pp. 2252–2265, Nov. 2008.

[15] L. Feng and W. M. Wonham, "Supervisory control architecture for discrete-event systems," *IEEE Trans. Autom. Control*, vol. 53, no. 6, pp. 1449–1461, Jul. 2008.

[16] R. Su and J. G. Thistle, "A distributed supervisor synthesis approach based on weak bisimulation," in *Proc. Int. Workshop Discrete Event Syst. (WODES06)*, Ann Arbor, MI, Jul. 2006, pp. 64–69.

[17] A. Mannani and P. Gohari, "Decentralized supervisory control of discrete-event systems over communication networks," *IEEE Trans. Autom. Control*, vol. 53, no. 2, pp. 547–559, Mar. 2008.

[18] P. Darondeau, "Distributed implementation of Ramadge-Wonham supervisory control with Petri nets," in *Proc. 44th IEEE Conf. Decision Control*, Seville, Spain, Dec. 2005, pp. 2107–2112.

[19] K. T. Seow, M. T. Pham, C. Ma, and M. Yokoo, "Coordination planning: Applying control synthesis methods for a class of distributed agents," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 2, pp. 405–415, Mar. 2009.

[20] K. T. Seow, C. Ma, and M. Yokoo, "Multiagent planning as control synthesis," in *Proc. 3rd Int. Joint Conf. AAMAS*, New York, NY, Jul. 2004, pp. 972–979.

[21] K. Cai and W. M. Wonham, "Supervisor localization: A top-down approach to distributed control of discrete-event systems," in *Proc. 2nd Med. Conf. Intelligent Syst. Autom. (CISA'09)*, Zarzis, Tunisia, Mar. 2009, pp. 302–308.

[22] S. Lafortune, "On decentralized and distributed control of partially-observed discrete event systems," in *Advances in Control Theory and Applications*. Berlin, Germany: Springer, 2007, vol. 353, pp. 171–184.

[23] R. Su and W. M. Wonham, "Supervisor reduction for discrete-event systems," *Discrete Event Dyn. Syst.*, vol. 14, no. 1, pp. 31–53, Jan. 2004.

[24] P. Gohari and W. M. Wonham, "On the complexity of supervisory control design in the RW framework," *IEEE Trans. Syst., Man, Cybern., Special Issue DES*, vol. 30, no. 5, pp. 643–652, Oct. 2000.
[25] W. M. Wonham, Design Software: XPTCT Systems Control Group, ECE Dept, Univ. Toronto, Tech. Rep., 2008 [Online]. Available: http://www.control.toronto.edu/DES
[26] P. M. Pardalos and J. Xue, "The maximum clique problem," *Global Optim.*, vol. 4, no. 3, pp. 301–328, Apr. 1994.
[27] L. Feng, "Computationally Efficient Supervisory Design for Discrete-Event Systems" Ph.D. dissertation, ECE Dept., Univ. Toronto, Toronto, ON, Canada, 2007.
[28] L. Feng and W. Wonham, "On the computation of natural observers in discrete-event systems," *Discrete Event Dyn. Syst.* [Online]. Available: www.springerlink.com
[29] L. Feng and W. M. Wonham, "Computationally efficient supervisory design: Control flow decomposition," in *Proc. Int. Workshop Discrete Event Syst. (WODES'06)*, Ann Arbor, MI, Jul. 2006, pp. 9–14.
[30] A. Vahidi, M. Fabian, and B. Lennartson, "Efficient supervisory synthesis of large systems," *Control Eng. Prac.*, vol. 14, no. 10, pp. 1157–1167, 2006.
[31] L. E. Holloway and B. H. Krogh, "Synthesis of feedback logic control for a class of controlled Petri nets," *IEEE Trans. Autom. Control*, vol. 35, no. 5, pp. 514–523, May 1990.
[32] L. Feng, K. Cai, and W. M. Wonham, "A structural approach to the nonblocking supervisory control of discrete-event systems," *Int. J. Adv. Manufact. Technol.*, vol. 41, no. 11–12, pp. 1152–1168, Apr. 2009.

Kai Cai (S'08–M'08) received the B.Eng. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 2006, the M.A.Sc. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2008, and is currently pursuing the Ph.D. degree at the Tokyo Institute of Technology, Tokyo, Japan.

His research interest is distributed control of multi-agent systems.

W. M. Wonham (M'64–SM'76–F'77–LF'00) received the B.Eng. degree in engineering physics from McGill University, Montreal, QC, Canada, in 1956, and the Ph.D. in control engineering from the University of Cambridge, Cambridge, U.K., in 1961.

From 1961 to 1969, he was associated with several U.S. research groups in control. Since 1970, he has been a faculty member in Systems Control, with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. He is the author of *Linear Multivariable Control: A Geometric Approach* (Berlin, Germany: Springer-Verlag, 1985, 3rd ed) and co-author of *Nonblocking Supervisory Control of State Tree Structures* (Berlin, Germany: Springer-Verlag, 2005). His research interests have included stochastic control and filtering, geometric multivariable control, and discrete-event systems.

Dr. Wonham is a Fellow of the Royal Society of Canada and a Foreign Associate of the (U.S.) National Academy of Engineering. He received the IEEE Control Systems Science and Engineering Award in 1987 and was the Brouwer Medallist of the Netherlands Mathematical Society in 1990. In 1996, he was appointed University Professor in the University of Toronto, and in 2000 University Professor Emeritus.