Discrete Event Dyn Syst (2015) 25:203–226
DOI 10.1007/s10626-014-0194-6

New results on supervisor localization, with case studies

Kai Cai · W. M. Wonham

Received: 7 February 2013 / Accepted: 23 April 2014 / Published online: 31 May 2014
© Springer Science+Business Media New York 2014

Abstract Recently we developed supervisor localization, a top-down approach to distributed control of discrete-event systems in the Ramadge-Wonham supervisory control framework. Its essence is the allocation of monolithic (global) control action among the local control strategies of individual agents. In this paper, we start by presenting several refinements of our localization theory. First, we drop the original assumption that the event sets of component agents are pairwise disjoint. Second, we show that consistent marking information can be enforced by just one agent, which can be selected arbitrarily. Third, the event sets of localized controllers are explicitly defined, in general as proper subsets of the entire event set. For these generalizations, we again prove that the collective local controlled behavior is identical to the global optimal and nonblocking controlled behavior. Moreover, we provide a language interpretation of localization by relating the key concept of control cover/congruence on the supervisor's state set to a special right congruence on the supervisor's language.W e go on to apply the extended supervisor localization to solve a multi-agent formation problem. We introduce a suitable formulation of formation invariance as well as shortest paths to formation. Local strategies are synthesized for a group of agents to arrive at a pre-specified formation in shortest paths; then issues of information exchange and control logic are examined. We further demonstrate the extended localization on a large-scale

This work was supported in part by Program to Disseminate Tenure Tracking System, MEXT, Japan (Kai Cai).

This work was supported in part by the Natural Sciences and Engineering Research Council, Canada, Grant no. 7399 (W. M. Wonham).

K. Cai (✉)
Urban Research Plaza, Osaka City University, Osaka, Japan
e-mail: kai.cai@info.eng.osaka-cu.ac.jp

W. M. Wonham
Systems Control Group, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada
e-mail: wonham@control.utoronto.ca

Cluster Tool example. By first synthesizing a set of decentralized supervisors and coordinators by an efficient heterarchical approach, our localization yields a distributed control architecture with comprehensible local control/coordination logic.

Keywords Discrete-event systems · Supervisory control · Supervisor localization · Distributed control · Automata

1 Introduction

Recently we developed a top-down approach, called *supervisor localization* (Cai and Wonham 2010a, b), to the distributed control of discrete-event systems in the Ramadge-Wonham (RW) supervisory control framework (Ramadge and Wonham 1987; Wonham 2013b). We view a plant to be controlled as comprised of independent asynchronous agents which are coupled implicitly through control specifications. To make the agents 'smart' and semi-autonomous, our localization algorithm allocates *external* supervisory control action to individual agents as their *internal* control strategies, while preserving the optimality (maximal permissiveness) and nonblocking properties of the overall monolithic (global) controlled behavior. Under the localization scheme, each agent controls only its own events, although it may very well need to observe events originating in other (typically neighboring) agents. We call such a scheme *distributed control architecture*.

Related, though distinct, control architectures are decentralized, hierarchical, and heterarchical (for recent developments see e.g. Feng and Wonham 2008; Schmidt and Breindl 2011; Su et al. 2012). Both the distributed and the latter modular approaches aim to achieve efficient computation and transparent control logic, while realizing monolithic optimality and nonblocking. With modular supervision, global control action is typically allocated among specialized supervisors enforcing individual specifications. By contrast, with our distributed supervision it is allocated among the individual active agents; Cai and Wonham (2010a, b) provide further discussion of this distinction.

We note that in Seow et al. (2009) and Pham and Seow (2012) the authors proposed a multi-agent coordination scheme in the RW framework similar in general terms to the distributed control architecture of our supervisor localization. Their synthesis procedure is essentially, however, a combination of the existing standard RW supervisor synthesis with partial observation (Wonham 2013b [Chapter 6]) and supervisor reduction (Su and Wonham 2004); no approach is presented to handle large-scale systems. By contrast, our localization approach (combined with a heterarchical framework) has been successfully applied to benchmark large systems (Cai and Wonham 2010a, b).

In this paper and its conference precursor (Cai and Wonham 2012a), we extend our supervisor localization theory in the following three respects. First, we drop the original assumption that component agents have pairwise disjoint event sets. As will be seen in Section 2, a local controller is computed for each (controllable) event; when an event is shared by two or more agents, the implementation of its local controller will be discussed. Second, we separate the marking issue from the control issue, and synthesize a *local marker* dedicated to maintaining correct marking information. The local marker may be implemented by an arbitrarily selected agent. Third, the event sets of localized controllers are explicitly defined, in general as proper subsets of the entire event set. These event sets determine the information exchange (or communication) structure among local controllers, and in turn among component agents. As in Cai and Wonham (2010a), the communication structure is not specified a priori, but emerges as part of the solution for localization. For these

extensions, we again prove that the collective local controlled behavior is identical to the global optimal and nonblocking controlled behavior. Furthermore, we provide a language interpretation of localization by relating the key concept of *control cover/congruence* on the supervisor's state set to a special *right congruence* on the supervisor's language.

We go on to apply extended supervisor localization to solve a multi-agent formation problem. The problem is to control a team of agents to assume certain formations, e.g. line, triangle, or circle. This problem finds application in many multi-agent cooperative tasks, including exploring an area or guarding a territory, and is a research topic of current vitality in robotics and 'standard' control communities (for some multi-agent formation problems see e.g. Anderson et al. 2008; Smith et al. 2012). We first introduce a suitable discrete-event formulation of *formation invariance* as well as *shortest paths to formation*. Then by localization we derive distributed control strategies that drive a group of agents to arrive at an arbitrarily pre-specified formation in shortest paths. Compared to deriving a local controller for each agent as in Cai and Wonham (2010a), deriving a local controller for each controllable event leads to simpler logic and more flexible implementation. Moreover, issues of information exchange and control logic are discussed.

Our second demonstration of extended supervisor localization is a large-scale Cluster Tool, with total state size approximately 3.6×10^{11}. Cluster Tool is an integrated semiconductor manufacturing system used for wafer processing (e.g. Yi et al. 2007); our model is adapted from Su et al. (2010, 2012) which consists of an array of robots routing wafers through different processing chambers connected sequentially by buffers. Because of the large state size, we combine localization with an efficient heterarchical approach (Feng and Wonham 2008): first synthesize a set of decentralized supervisors and coordinators to achieve global optimal and nonblocking supervision, and then apply localization to decompose each decentralized supervisor/coordinator into local controllers/markers for the relevant controllable events. As a result, each controllable event may be associated with multiple local controllers/markers. Again localization with respect to each controllable event allows deriving simpler control logic than localization with respect to each agent in Cai and Wonham (2010a).

Finally, this paper differs from its conference precursor (Cai and Wonham 2012a) by (1) including proofs of all results, (2) a language interpretation of key concepts in supervisor localization, (3) treatment of multi-agent formation control and the corresponding localization results, and (4) a detailed demonstration of extended supervisor localization on a large-scale Cluster Tool.

The rest of the paper is organized as follows. In Section 2 we formulate the distributed control problem. Section 3 presents extended supervisor localization theory. Section 4 provides a language interpretation of key concepts in supervisor localization. Section 5 applies extended localization to solve a multi-agent formation problem. Section 6 demonstrates extended localization on a large-scale Cluster Tool. Finally in Section 7 we state conclusions.

2 Problem formulation

The plant to be controlled is modeled by a generator

$$\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m) \tag{1}$$

where Q is the state set; $q_0 \in Q$ is the initial state; $Q_m \subseteq Q$ 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; $\delta : Q \times \Sigma \to Q$ is the (partial) state transition function. In the usual way, δ is extended to $\delta : Q \times \Sigma^* \to Q$, and we write $\delta(q, s)!$ to mean that $\delta(q, s)$ is defined. The *closed behavior* of $\mathbf{G}$ is the language

$$L(\mathbf{G}) := \{s \in \Sigma^* | \delta(q_0, s)!\} \tag{2}$$

and the *marked behavior* is

$$L_m(\mathbf{G}) := \{s \in L(\mathbf{G}) | \delta(q_0, s) \in Q_m\} \subseteq L(\mathbf{G}). \tag{3}$$

We consider the plant $\mathbf{G}$ consisting of n component agents $\mathbf{G}_k = (Q_k, \Sigma_k, \delta_k, q_{0,k}, Q_{m,k})$, $k \in [1, n]$.[1] In terms of language, this means the following: 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 respectively

$$L(\mathbf{G}) = ||\{L_k | k \in [1, n]\} \text{ and } L_m(\mathbf{G}) = ||\{L_{m,k} | k \in [1, n]\},$$

where $||$ denotes *synchronous product* (Wonham 2013b). We say that $\mathbf{G}$ is *nonblocking* if $\overline{L_m(\mathbf{G})} = L(\mathbf{G})$, where $\overline{\cdot}$ denotes *prefix closure* (Wonham 2013b). We note that, compared to Cai and Wonham (2010a), here no requirement is imposed that the agents' event sets Σ_k be pairwise disjoint. Consequently $\mathbf{G}$ need not be nonblocking in general.

The component agents $\mathbf{G}_k$, $k \in [1, n]$, are implicitly coupled through a specification language $E \subseteq \Sigma^*$ that imposes behavioral constraints on $\mathbf{G}$. Recall (Wonham 2013b) that a language $F \subseteq \Sigma^*$ is *controllable* with respect to $\mathbf{G}$ if

$$\bar{F}\Sigma_u \cap L(\mathbf{G}) \subseteq \bar{F}. \tag{4}$$

Whether or not F is controllable, we denote by $\mathcal{C}(F)$ the set of all controllable sublanguages of F. Then $\mathcal{C}(F)$ is nonempty (since the empty language $\emptyset$ belongs), and contains a unique supremal element, denoted $\sup \mathcal{C}(F)$ (Wonham 2013b). Now for the plant $\mathbf{G}$ and the imposed specification E, let $\mathbf{SUP} = (X, \Sigma, \xi, x_0, X_m)$ be the corresponding monolithic supervisor that is optimal (i.e. maximally permissive) and nonblocking: the marked language of $\mathbf{SUP}$ is

$$L_m(\mathbf{SUP}) := \sup \mathcal{C}(E \cap L_m(\mathbf{G})) \subseteq \Sigma^*, \tag{5}$$

and the closed language is $L(\mathbf{SUP}) = \bar{L}_m(\mathbf{SUP})$. Throughout the paper we assume that $L_m(\mathbf{SUP}) \neq \emptyset$. For a subset $\Sigma_o \subseteq \Sigma$, the *natural projection* $P_o : \Sigma^* \to \Sigma_o^*$ is defined according to

$$\begin{aligned} P_o(\epsilon) &= \epsilon, \quad \epsilon \text{ is the empty string;} \\ P_o(\sigma) &= \begin{cases} \epsilon, & \text{if } \sigma \notin \Sigma_o, \\ \sigma, & \text{if } \sigma \in \Sigma_o; \end{cases} \\ P_o(s\sigma) &= P_o(s)P_o(\sigma), \quad s \in \Sigma^*, \sigma \in \Sigma. \end{aligned} \tag{6}$$

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

Let α be an arbitrary controllable event, i.e. $\alpha \in \Sigma_c$. We say that a generator $\mathbf{LOC}_\alpha = (Y_\alpha, \Sigma_\alpha, \zeta_\alpha, y_{0,\alpha}, Y_{m,\alpha})$, $\Sigma_\alpha \subseteq \Sigma$, is a *local controller* for α if $\mathbf{LOC}_\alpha$ can disable only α. Let $P_\alpha : \Sigma^* \to \Sigma_\alpha^*$ be the natural projection as in Eq. 6. Then in terms of language, the above condition means that for all $s \in \Sigma^*$ and $\sigma \in \Sigma$, there holds

$$s\sigma \in L(\mathbf{G}) \,\&\, s \in P_\alpha^{-1}L(\mathbf{LOC}_\alpha) \,\&\, s\sigma \notin P_\alpha^{-1}L(\mathbf{LOC}_\alpha) \Rightarrow \sigma = \alpha.$$

[1] In this paper the notation $[1, n]$ is used for the integer interval $\{1, ..., n\}$.

The event set Σ_α of $\mathbf{LOC}_\alpha$ in general satisfies $\{\alpha\} \subseteq \Sigma_\alpha \subseteq \Sigma$. In typical cases, both subset containments are strict; in fact the events in $\Sigma_\alpha \setminus \{\alpha\}$ may be critical to achieve useful synchronization with the local controllers for other controllable events. It is worth emphasizing that Σ_α is not fixed *a priori*, but will be systematically determined, as part of our localization result, to guarantee correct local control. Below we shall precisely define Σ_α, which was not given explicitly in Cai and Wonham (2010a).

In addition to local controllers, we introduce a *local marker* $\mathbf{LOC}_M = (Z, \Sigma_M, \zeta_M, z_0, Z_m)$, $\Sigma_M \subseteq \Sigma$, which is also a generator such that

$$L(\mathbf{SUP}) \cap (L_m(\mathbf{G}) \cap P_M^{-1} L_m(\mathbf{LOC}_M)) = L_m(\mathbf{SUP}),$$

where $P_M : \Sigma^* \to \Sigma_M^*$ is the natural projection as in Eq. 6. One may think of $\mathbf{LOC}_M$ as monitoring the closed behavior of **SUP**, and sounding a 'beep' exactly when a string in $L_m(\mathbf{G}) \cap P_M^{-1} L_m(\mathbf{LOC}_M)$ is generated. On the other hand, $\mathbf{LOC}_M$ exercises no control action; namely, for all $s \in \Sigma^*$ and $\sigma \in \Sigma$, there holds

$$s\sigma \in L(\mathbf{G}) \;\&\; s \in P_M^{-1} L(\mathbf{LOC}_M) \;\Rightarrow s\sigma \in P_M^{-1} L(\mathbf{LOC}_M).$$

The event set $\Sigma_M \subseteq \Sigma$ of $\mathbf{LOC}_M$, just like Σ_α of $\mathbf{LOC}_\alpha$ above, will be generated as part of our localization result to ensure correct marking.

We are ready to formulate the *Distributed Control Problem*. Construct a set of local controllers $\{\mathbf{LOC}_\alpha | \alpha \in \Sigma_c\}$ and a local marker $\mathbf{LOC}_M$, with

$$L(\mathbf{LOC}) := \bigcap_{\alpha \in \Sigma_c} P_\alpha^{-1} L(\mathbf{LOC}_\alpha) \cap P_M^{-1} L(\mathbf{LOC}_M), \tag{7}$$

$$L_m(\mathbf{LOC}) := \bigcap_{\alpha \in \Sigma_c} P_\alpha^{-1} L_m(\mathbf{LOC}_\alpha) \cap P_M^{-1} L_m(\mathbf{LOC}_M), \tag{8}$$

such that the following two properties hold:

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

For the sake of easy implementation and comprehensibility, it would be desired in practice that the state sizes of local controllers and local marker be very much less than that of their parent monolithic supervisor. Inasmuch as this property is neither precise to state nor always achievable, it is omitted from the formal problem statement; in applications, nevertheless, it should be kept in mind.

Finally, compared to the localization in Cai and Wonham (2010a), which is with respect to individual agents $\mathbf{G}_k$, the localization scheme in this paper is with respect to each controllable event; and moreover the two issues, control and marking, are treated separately. This scheme allows diverse ways of allocating the local controllers $\mathbf{LOC}_\alpha$, $\alpha \in \Sigma_c$, and the local marker $\mathbf{LOC}_M$ among the set of component agents $\mathbf{G}_k$, $k \in [1, n]$. For example, if $\mathbf{G}_k$ and $\mathbf{G}_j$ share a controllable event α, i.e. $\alpha \in \Sigma_{c,k} \cap \Sigma_{c,j}$, then the local controller $\mathbf{LOC}_\alpha$ can be allocated to either agent or to both. Also, the local marker may be implemented by some arbitrarily chosen agent(s), whereas in Cai and Wonham (2010a) all agents need to

deal with marking. Among others, the following is a convenient allocation, in the sense that every $\mathbf{LOC}_\alpha$, and $\mathbf{LOC}_M$, is implemented by exactly one agent.

$$\begin{aligned}
&\mathbf{G}_1: \quad (\forall \alpha \in \Sigma_{c,1})\, \mathbf{LOC}_\alpha,\ \mathbf{LOC}_M \\
&\mathbf{G}_2: \quad (\forall \alpha \in \Sigma_{c,2} \setminus \Sigma_{c,1})\, \mathbf{LOC}_\alpha \\
&\vdots \\
&\mathbf{G}_n: \quad (\forall \alpha \in \Sigma_{c,n} \setminus (\Sigma_{c,n-1} \cup \cdots \cup \Sigma_{c,1}))\, \mathbf{LOC}_\alpha
\end{aligned} \tag{9}$$

3 Supervisor localization

We solve the Distributed Control Problem by developing a supervisor localization procedure similar to Cai and Wonham (2010a), with the difference that the issues of control and marking are dealt with separately. The main concepts were originally adapted from Su and Wonham (2004) for supervisor reduction.

3.1 Control localization

Fix a controllable event $\alpha \in \Sigma_c$. We will establish a *control cover* on X (the state set of **SUP**) based only on the control information pertaining to α, as captured by the following two functions. First define $E_\alpha : X \to \{1, 0\}$ according to

$$E_\alpha(x) = 1 \ \text{ iff } \ \xi(x, \alpha)!. \tag{10}$$

Thus $E_\alpha(x) = 1$ means that α is enabled at x. Next define $D_\alpha : X \to \{1, 0\}$ according to $D_\alpha(x) = 1$ iff

$$\neg\xi(x, \alpha)! \ \ \&\ \ (\exists s \in \Sigma^*)(\xi(x_0, s) = x \ \&\ \delta(q_0, s\alpha)!). \tag{11}$$

So $D_\alpha(x) = 1$ means that α must be disabled at x (i.e. α is not defined at x, but x is reached by a string s such that the string $s\alpha$ is in the closed behavior $L(\mathbf{G})$). Notice that if $\neg\xi(x, \alpha)!$ but $(\forall s \in \Sigma^*)\ \xi(x_0, s) = x \Rightarrow \neg\delta(\delta(q_0, s), \alpha)!$, then $E_\alpha(x) = D_\alpha(x) = 0$; we refer to such x as a 'don't care' state. Based on this enablement and disablement information of α, we define the following binary relation $\mathcal{R}_\alpha$ on X, called 'control consistency'.

Definition 1 A binary relation $\mathcal{R}_\alpha \subseteq X \times X$ is a *control consistency relation* with respect to α if for every $x, x' \in X$, $(x, x') \in \mathcal{R}_\alpha$ iff

$$E_\alpha(x) \cdot D_\alpha(x') = 0 = E_\alpha(x') \cdot D_\alpha(x).$$

Let $\mathcal{R}_\alpha$ be a control consistency relation. Then a pair of states (x, x') is *not* control consistent exactly when α is enabled at x and is disabled at x', or vice versa. Otherwise, $(x, x') \in \mathcal{R}_\alpha$. Similar to Cai and Wonham (2010a), $\mathcal{R}_\alpha$ is reflexive and symmetric, but 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 $\mathcal{C}_\alpha = \{X_i \subseteq X | i \in I_\alpha\}$ be a cover on X, with I_α a suitable index set. We say that $\mathcal{C}_\alpha$ is a *control cover* with respect to α if

$$\begin{aligned}
&(i)\ \ (\forall i \in I_\alpha, \forall x, x' \in X_i)\ (x, x') \in \mathcal{R}_\alpha,\\
&(ii)\ \ (\forall i \in I_\alpha, \forall \sigma \in \Sigma)\,[((\exists x \in X_i)\ \xi(x, \sigma)!) \Rightarrow\\
&\qquad \big((\exists j \in I_\alpha)(\forall x' \in X_i)\xi(x', \sigma)! \Rightarrow \xi(x', \sigma) \in X_j\big)\big].
\end{aligned}$$

A control cover $\mathcal{C}_\alpha$ lumps states of **SUP** into (possibly overlapping) cells $X_i, i \in I_\alpha$. According to (i) all states that reside in a cell X_i 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 by a one-step transition σ must be covered by the same cell X_j. Inductively, two states x, x' belong to a common cell of $\mathcal{C}_\alpha$ 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 a given string are again control consistent. We say that a control cover $\mathcal{C}_\alpha$ is a *control congruence* if $\mathcal{C}_\alpha$ happens to be a *partition* on X, namely its cells are pairwise disjoint. To compute a local controller for event α, the localization algorithm we proposed in Cai and Wonham (2010a) in fact searches for a control congruence to achieve computational efficiency. In Section 4, below, we will further explain this important concept in terms of equivalence relations on languages.

Having defined a control cover $\mathcal{C}_\alpha$ on X, we construct a local controller $\mathbf{LOC}_\alpha = (Y_\alpha, \Sigma_\alpha, \zeta_\alpha, y_{0,\alpha}, Y_{m,\alpha})$ for the event α as follows.

(S1) The state set is $Y_\alpha := I_\alpha$, with each state $y \in Y_\alpha$ being a cell X_i of the cover $\mathcal{C}_\alpha$. In particular, the initial state $y_{0,\alpha}$ is a cell X_{i_0} containing x_0, i.e. $x_0 \in X_{i_0}$, and the marker state set $Y_{m,\alpha} := \{i \in I_\alpha | X_i \cap X_m \neq \emptyset\}$.

(S2) For the event set Σ_α, define the transition function $\zeta'_\alpha : I_\alpha \times \Sigma \to I_\alpha$ over the entire event set Σ by $\zeta'_\alpha(i, \sigma) = j$ if $(\exists x \in X_i)\xi(x, \sigma) \in X_j$ and $(\forall x' \in X_i)\big[\xi(x', \sigma)! \Rightarrow \xi(x', \sigma) \in X_j\big]$. Choose Σ_α to be the union of $\{\alpha\}$ with other events which are *not* selfloop transitions of ζ'_α, i.e.

$$\Sigma_\alpha := \{\alpha\} \dot\cup \{\sigma \in \Sigma \setminus \{\alpha\} | (\exists i, j \in I_\alpha)\ i \neq j \& \zeta'_\alpha(i, \sigma) = j\}. \tag{12}$$

Clearly $\{\alpha\} \subseteq \Sigma_\alpha \subseteq \Sigma$.

(S3) Define the transition function ζ_α to be the restriction of ζ'_α to Σ_α; namely $\zeta_\alpha := \zeta'_\alpha|_{I_\alpha \times \Sigma_\alpha} : I_\alpha \times \Sigma_\alpha \to I_\alpha$.

In the above construction, owing to the possible overlapping of cells in the cover $\mathcal{C}_\alpha$, the choices of $y_{0,\alpha}$ and ζ_α may not be unique, and consequently $\mathbf{LOC}_\alpha$ may not be unique. In that case we pick an arbitrary instance of $\mathbf{LOC}_\alpha$. If $\mathcal{C}_\alpha$ happens to be a control congruence, however, then $\mathbf{LOC}_\alpha$ is determined uniquely. By the same procedure as above, we generate a set of local controllers $\mathbf{LOC}_\alpha$, one for each controllable event $\alpha \in \Sigma_c$.

3.2 Marking localization

We deal now with marking information. Define $M : X \to \{1, 0\}$ according to

$$M(x) = 1 \ \text{ iff } x \in X_m. \tag{13}$$

Thus $M(x) = 1$ means that state x is marked in **SUP**. Also define $T : X \to \{1, 0\}$ according to

$$T(x) = 1 \ \text{ iff } (\exists s \in \Sigma^*)\xi(x_0, s) = x \ \&\ \delta(q_0, s) \in Q_m. \tag{14}$$

So $T(x) = 1$ means that there is a string that reaches x and also reaches some marked state in $\mathbf{G}$. Note that for each $x \in X$, it follows from $L_m(\mathbf{SUP}) \subseteq L_m(\mathbf{G})$ that $T(x) = 0 \Rightarrow M(x) = 0$ and $M(x) = 1 \Rightarrow T(x) = 1$. Based on the above marking information, we define the following binary relation $\mathcal{R}_M$ on X, called 'marking consistency'.

Definition 3 Let $\mathcal{R}_M \subseteq X \times X$. We say that $\mathcal{R}_M$ is a *marking consistency relation* if for every $x, x' \in X$, $(x, x') \in \mathcal{R}_M$ iff

$$T(x) = T(x') \Rightarrow M(x) = M(x').$$

Thus, a pair of states (x, x') is marking consistent if x and x' are both marked or both unmarked in **SUP**, provided either (i) there exist strings s and s' that reach x and x', respectively, and both reach some marked state(s) in $\mathbf{G}$, or (ii) no string that reaches both x and x', reaches any marked state in $\mathbf{G}$. Again it is verified that $\mathcal{R}_M$ is reflexive and symmetric, but need not be transitive, and consequently not an equivalence relation; this leads to the following definition of *marking cover*, analogous to the control cover above. The difference is that here the marking consistency relation $\mathcal{R}_M$ is used instead of the control consistency $\mathcal{R}_\alpha$.

Definition 4 Let I be some index set, and $\mathcal{C}_M = \{X_i \subseteq X | i \in I\}$ a cover on X. We say $\mathcal{C}_M$ is a *marking cover* if

$$\begin{aligned}
&(i) \quad (\forall i \in I, \forall x, x' \in X_i)\ (x, x') \in \mathcal{R}_M, \\
&(ii) \quad (\forall i \in I, \forall \sigma \in \Sigma)\,[((\exists x \in X_i)\ \xi(x, \sigma)!) \Rightarrow \\
&\qquad \left((\exists j \in I)(\forall x' \in X_i)\ \xi(x', \sigma)! \Rightarrow \xi(x', \sigma) \in X_j\right)].
\end{aligned}$$

Again, we say a marking cover $\mathcal{C}_M$ is a *marking congruence* if $\mathcal{C}_M$ happens to be a *partition* on X.

With the marking cover $\mathcal{C}_M$, we construct a (nonblocking) generator $\mathbf{LOC}_M = (Z, \Sigma_M, \zeta_M, z_0, Z_m)$ by the same three steps (S1)–(S3) as in the previous subsection, except for choosing (cf. (12))

$$\Sigma_M := \{\sigma \in \Sigma | (\exists i, j \in I)\ i \neq j\ \&\ \zeta'_M(i, \sigma) = j\}. \tag{15}$$

3.3 Main result

Now we present the main result of this section.

Theorem 5 *The set of local controllers* $\{\mathbf{LOC}_\alpha | \alpha \in \Sigma_c\}$ *and the local marker* $\mathbf{LOC}_M$ *constructed above solve the Distributed Control Problem; that is,*

$$L(\mathbf{G}) \cap L(\mathbf{LOC}) = L(\mathbf{SUP}) \tag{16}$$

$$L_m(\mathbf{G}) \cap L_m(\mathbf{LOC}) = L_m(\mathbf{SUP}) \tag{17}$$

where $L(\mathbf{LOC})$ *and* $L_m(\mathbf{LOC})$ *are as defined in* Eqs. 7 *and* 8*, respectively.*

Theorem 5 states that every set of control covers and every marking cover together generate a solution to the Distributed Control Problem. In particular, a set of *state-minimal* local controllers and a state-minimal local marker (possibly non-unique) can in principle be defined from a set of suitable control covers and a suitable marking cover. The minimal state problem, however, is known to be NP-hard (Su and Wonham 2004); for computation in this paper (Section 5 below), we shall resort to the polynomial-time localization algorithm designed

in Cai and Wonham (2010a) which generates control (resp. marking) congruences instead of covers.

We note that Theorem 5 refines the main result of Cai and Wonham (2010a) in the following respects. First, the plant component agents $\mathbf{G}_k, k \in [1, n]$, may share events. Second, the event sets Σ_α of local controllers and the event set Σ_M of the local marker are explicitly given in Eqs. 12 and 15, in general as proper subsets of Σ. Third, the marking issue is separated from the control issue, and is enforced by a single local marker $\mathbf{LOC}_M$.

We now prove Theorem 5, first the ($\supseteq$) part of Eqs. 16 and 17, and then the ($\subseteq$) part. Compared to the proof in Cai and Wonham (2010a) where the local event sets Σ_α and Σ_M are assumed to be equal to Σ, here we address the general case $\Sigma_\alpha \subseteq \Sigma$ and $\Sigma_M \subseteq \Sigma$ by using natural projections P_α, P_M, and their inverse-image functions.

Proof of Theorem 5 ($\supseteq$, Eq. 17) Since $L_m(\mathbf{SUP}) \subseteq L_m(\mathbf{G})$, it suffices to show that $L_m(\mathbf{SUP}) \subseteq L_m(\mathbf{LOC})$, i.e. $(\forall \alpha \in \Sigma_c)\ L_m(\mathbf{SUP}) \subseteq P_\alpha^{-1} L_m(\mathbf{LOC}_\alpha)$ and $L_m(\mathbf{SUP}) \subseteq P_M^{-1} L_m(\mathbf{LOC}_M)$ by Eq. 8. Let $s = \sigma_0 \sigma_1 \cdots \sigma_h \in L_m(\mathbf{SUP})$. Then $x_1 := \xi(x_0, \sigma_0) \in X, \ldots, x_h := \xi(x_0, \sigma_0 \cdots \sigma_{h-1}) \in X, x_{h+1} := \xi(x_0, s) \in X_m$. By the construction of $\mathbf{LOC}_\alpha$ ($\alpha \in \Sigma_c$ arbitrary), in particular the transition function ζ'_α over Σ, there exist $i_0, i_1, \ldots, i_{h+1}$ (with $i_0 = y_{0,\alpha}$) such that

$$\begin{aligned} x_0 \in X_{i_0} \ \ \& \ \ \zeta'_\alpha(i_0, \sigma_0) &= i_1, \\ x_1 \in X_{i_1} \ \ \& \ \ \zeta'_\alpha(i_1, \sigma_1) &= i_2, \\ &\vdots \\ x_{h+1} \in X_{i_{h+1}} \ \ \& \ \ \zeta'_\alpha(i_h, \sigma_h) &= i_{h+1}. \end{aligned} \tag{18}$$

So $\zeta'_\alpha(i_0, \sigma_0 \sigma_1 \cdots \sigma_h) = \zeta'_\alpha(i_0, s)!$, and belongs to $Y_{m,\alpha}$ because $X_{i_{h+1}} \cap X_m \neq \emptyset$ (x_{h+1} belongs). Moreover since any $\sigma \notin \Sigma_\alpha$ (defined in Eq. 12) is only a selfloop transition of ζ'_α, we derive $\zeta_\alpha(i_0, P_\alpha(s)) \in Y_{m,\alpha}$. Hence $P_\alpha(s) \in L_m(\mathbf{LOC}_\alpha)$, i.e. $s \in P_\alpha^{-1} L_m(\mathbf{LOC}_\alpha)$. A similar argument yields $s \in P_M^{-1} L_m(\mathbf{LOC}_M)$. ($\supseteq$, Eq. 16) This is an easy consequence of ($\supseteq$, Eq. 17):

$$\begin{aligned} L(\mathbf{SUP}) = \overline{L_m(\mathbf{SUP})} &\subseteq \overline{L_m(\mathbf{G}) \cap L_m(\mathbf{LOC})} \\ &\subseteq \overline{L_m(\mathbf{G})} \cap \overline{L_m(\mathbf{LOC})} \\ &\subseteq L(\mathbf{G}) \cap L(\mathbf{LOC}). \end{aligned}$$

($\subseteq$, Eq. 16) We show this by induction. First, the empty string ϵ belongs to $L(\mathbf{G})$, $L(\mathbf{LOC})$, and $L(\mathbf{SUP})$, because these (closed) languages are all nonempty. Now suppose $s \in L(\mathbf{G}) \cap L(\mathbf{LOC}) \Rightarrow s \in L(\mathbf{SUP})$, and $s\alpha \in L(\mathbf{G}) \cap L(\mathbf{LOC})$, $\alpha \in \Sigma$. It will be proved that $s\alpha \in L(\mathbf{SUP})$. This is the case when $\alpha \in \Sigma_u$, since $\mathbf{SUP}$ is controllable. Let $\alpha \in \Sigma_c$. Then by hypothesis and Eq. 7, $s, s\alpha \in P_\alpha^{-1}(L(\mathbf{LOC}_\alpha))$, i.e. $P_\alpha(s), P_\alpha(s)\alpha \in L(\mathbf{LOC}_\alpha)$. Write $i := \zeta_\alpha(y_{0,\alpha}, P_\alpha(s))$ and $j := \zeta_\alpha(i, \alpha)$. By the definition of ζ_α (and ζ'_α), there exist $x \in X_i$, $x' \in X_j$ such that $\xi(x, \alpha) = x'$; hence $E_\alpha(x) = 1$ (defined in Eq. 10). On the other hand, by hypothesis $s \in L(\mathbf{SUP})$, i.e. $\xi(x_0, s)!$. Let $s = \sigma_0 \sigma_1 \cdots \sigma_h$; since $\zeta_\alpha(y_{0,\alpha}, P_\alpha(s)) = i$, it follows from the definition of ζ'_α that $\zeta'_\alpha(y_{0,\alpha}, s) = \zeta'_\alpha(y_{0,\alpha}, \sigma_0 \sigma_1 \cdots \sigma_h) = i$. Hence, performing the same construction as in Eq. 18 above and identifying $i = i_{h+1}$, we derive $\xi(x_0, s) \in X_i$; and by the control cover Definition 2, it holds that $(x, \xi(x_0, s)) \in \mathcal{R}_\alpha$. It then follows from Definition 1 that $D_\alpha(\xi(x_0, s)) = 0$. Since $s\alpha \in L(\mathbf{G})$, i.e. $\delta(\delta(q_0, s), \alpha)!$, we conclude that $\xi(\xi(x_0, s), \alpha)!$, namely $s\alpha \in L(\mathbf{SUP})$.

($\subseteq$, Eq. 17) Let $s \in L_m(\mathbf{G}) \cap L_m(\mathbf{LOC})$; by Eq. 8, $s \in P_M^{-1}(L_m(\mathbf{LOC}_M))$, i.e. $P_M(s) \in L_m(\mathbf{LOC}_M)$. Write $i_m := \zeta_M(z_0, P_M(s))$. Then there exists $x \in X_{i_m} \cap X_m$ (here X_{i_m} is the

cell labeled by i_m); so that $M(x) = 1$ (defined in Eq. 13), which also implies $T(x) = 1$ (defined in Eq. 14). On the other hand, since $L_m(\mathbf{G}) \cap L_m(\mathbf{LOC}) \subseteq L(\mathbf{G}) \cap L(\mathbf{LOC}) = L(\mathbf{SUP})$ (the last equality has been shown above), we have $s \in L(\mathbf{SUP})$. That is, $\xi(x_0, s)!$. Again let $s = \sigma_0\sigma_1 \cdots \sigma_h$; since $\zeta_M(z_0, P_M(s)) = i_m$, it follows from the definition of ζ'_M (similar to ζ'_α) that $\zeta'_M(z_0, s) = \zeta'_M(z_0, \sigma_0\sigma_1 \cdots \sigma_h) = i_m$. Hence, performing the same construction as in Eq. 18 above and identifying $i_m = i_{h+1}$, we derive $\xi(x_0, s) \in X_{i_m}$; and by the marking cover Definition 4, it holds that $(x, \xi(x_0, s)) \in \mathcal{R}_M$. Since $s \in L_m(\mathbf{G})$, i.e. $\delta(q_0, s) \in Q_m$, we have $T(\xi(x_0, s)) = 1$. Therefore by Definition 3, $M(\xi(x_0, s)) = 1$, i.e. $s \in L_m(\mathbf{SUP})$. □

We remark that in proving ($\subseteq$, Eq. 16) only the properties of local controllers $\mathbf{LOC}_\alpha$ (control information) were used, and in proving ($\subseteq$, Eq. 17) only the properties of local marker $\mathbf{LOC}_M$ (marking information) were used. That is, the issues of control and marking are separated in the proof of Theorem 5. This fact also indicates that the local controllers collectively guarantee correct, closed controlled behavior; while the local marker guarantees correct, marked behavior.

4 Language interpretation of control/marking congruences

In light of the (Nerode) identification of "states" with the cells of a right congruence on a language, it is of theoretical interest to identify a control congruence on the state set X of the supervisor **SUP** with a corresponding construct on $L(\mathbf{SUP})$. In this way control congruence will be seen to arise "naturally" in the language framework of supervisory control theory.

In this section we relate control congruence on X to a special right congruence on $L(\mathbf{SUP})$. A similar relation can be established for marking congruence (Definition 4). Several concepts to be used in this section—equivalence relation, Nerode equivalence relation, canonical recognizer, and right congruence—are standard, and can be found in e.g. Wonham (2013b [Chapters 1,2]).

Let $\mathbf{SUP} = (X, \Sigma, \xi, x_0, X_m)$ be a supervisor, with closed language $L(\mathbf{SUP})$ and marked language $L_m(\mathbf{SUP})$. Let $\equiv_{L_m(\mathbf{SUP})}$ be the Nerode equivalence relation on $L(\mathbf{SUP})$ with respect to $L_m(\mathbf{SUP})$; that is, for arbitrary strings $s, t \in L(\mathbf{SUP})$, s and t are Nerode equivalent, written $(s, t) \in \equiv_{L_m(\mathbf{SUP})}$, if and only if

$$(\forall u \in \Sigma^*)\ su \in L_m(\mathbf{SUP}) \Leftrightarrow tu \in L_m(\mathbf{SUP}).$$

Write $| \equiv_{L_m(\mathbf{SUP})} |$ for the cardinality (number of cells) of $\equiv_{L_m(\mathbf{SUP})}$. Suppose $|X| = | \equiv_{L_m(\mathbf{SUP})} |$, i.e. **SUP** is the canonical recognizer of the language $L_m(\mathbf{SUP})$.

Now fix $\alpha \in \Sigma_c$ and let $\mathcal{C}_\alpha = \{X_i \subseteq X | i \in I_\alpha\}$ be a control congruence on X with pairwise disjoint cells. Thus $\mathcal{C}_\alpha$ corresponds to a partition $\mathcal{P}_\alpha$ on $L(\mathbf{SUP})$ as follows. For each state $x \in X$ let $[x] := \{s \in L(\mathbf{SUP}) | \xi(x_0, s) = x\}$, and for each cell $X_i \subseteq X$ let $[X_i] := \bigcup\{[x] | x \in X_i\}$. Then $\mathcal{P}_\alpha := \{[X_i] \subseteq L(\mathbf{SUP}) | i \in I_\alpha\}$ is the partition corresponding to $\mathcal{C}_\alpha$. The partition $\mathcal{P}_\alpha$ corresponds to an equivalence relation E_α on $L(\mathbf{SUP})$ according to $(s, t) \in E_\alpha$ if and only if

$$(\exists i \in I_\alpha)\ s \in [X_i]\ \&\ t \in [X_i].$$

The equivalence relation E_α on $L(\mathbf{SUP})$ thus corresponds to the control congruence $\mathcal{C}_\alpha$ on X. Our result is the following.

Theorem 6 *The equivalence relation E_α on $L(\mathbf{SUP})$ has the following properties:*

(i) *E_α is a right congruence on $L(\mathbf{SUP})$, i.e.*

$$(\forall s, t \in L(\mathbf{SUP}))(\forall u \in \Sigma^*)\ su, tu \in L(\mathbf{SUP})\ \&\ (s,t) \in E_\alpha \Rightarrow (su, tu) \in E_\alpha. \tag{19}$$

(ii) $(\forall s, t \in L(\mathbf{SUP}))\ (s,t) \in \equiv_{L_m(\mathbf{SUP})} \Rightarrow (s,t) \in E_\alpha$.

(iii) $(\forall s, t \in L(\mathbf{SUP}))\ (s,t) \in E_\alpha \Rightarrow (\xi(x_0, s), \xi(x_0, t)) \in \mathcal{R}_\alpha$.

Proof

(i) We argue by induction on the length of string $u \in \Sigma^*$. If $u = \epsilon$, then Eq. 19 holds trivially. Suppose $u = \sigma$, an arbitrary event, and let $s\sigma, t\sigma \in L(\mathbf{SUP})$, $(s,t) \in E_\alpha$. It will be shown that $(s\sigma, t\sigma) \in E_\alpha$. From $(s,t) \in E_\alpha$ we know that there exists $i \in I_\alpha$ such that $s, t \in [X_i]$. It follows that $(\exists x, x' \in X_i)\ \xi(x_0, s) = x,\ \xi(x_0, t) = x'$. Since $s\sigma, t\sigma \in L(\mathbf{SUP})$, i.e. $\xi(x, \sigma)!$ and $\xi(x', \sigma)!$, by Definition 2(ii) there exists $j \in I_\alpha$ such that $\xi(x, \sigma), \xi(x', \sigma) \in [X_j]$. That is, $s\sigma, t\sigma \in [X_j]$, and therefore $(s\sigma, t\sigma) \in E_\alpha$. Inductively, Eq. 19 holds for an arbitrary string $u \in \Sigma^*$.

(ii) Let $s, t \in L(\mathbf{SUP})$. Then

$$\begin{aligned}(s,t) \in \equiv_{L_m(\mathbf{SUP})} &\Rightarrow (\exists x \in X)\ s, t \in [x] \\ &\Rightarrow (\exists i \in I_\alpha)\ s, t \in [x] \subseteq [X_i] \\ &\Rightarrow (s,t) \in E_\alpha\end{aligned}$$

(iii) Let $s, t \in L(\mathbf{SUP})$. Then

$$\begin{aligned}(s,t) \in E_\alpha &\Rightarrow (\exists i \in I_\alpha)\ s, t \in [X_i] \\ &\Rightarrow (\exists x, x' \in X_i)\ \xi(x_0, s) = x,\ \xi(x_0, t) = x' \\ &\Rightarrow (\xi(x_0, s), \xi(x_0, t)) \in \mathcal{R}_\alpha\end{aligned}$$

□

We have thus established the correspondence of the control congruence $\mathcal{C}_\alpha$ on X to the special equivalence relation E_α on $L(\mathbf{SUP})$: E_α is a right congruence on $L(\mathbf{SUP})$ that is 'coarser' than the Nerode equivalence relation $\equiv_{L_m(\mathbf{SUP})}$ and 'finer' than the control consistency relation $\mathcal{R}_\alpha$ (Definition 1). With E_α and $L(\mathbf{SUP})$, a (finite-state) generator may be defined (see Wonham 2013b [Section 2.3]); owing to the properties of E_α, the generator is a valid local controller for event α.

5 Multi-agent formations

In this section, we apply extended supervisor localization to solve a multi-agent formation problem. New features of extended localization will be illustrated. The goal of formation control is to guide a team of agents, moving on a plane, to assume certain geometric formations, e.g. line, triangle, or circle. This problem finds application in many multi-agent cooperative tasks, including exploring an area or guarding a territory, and it is often desirable to design local control strategies for individual agents (e.g. Anderson et al. 2008; Smith et al. 2012). The discrete-event formulation of the formation problem allows us to address logical control specifications such as specific ordering and/or mutual exclusion of entering a location, and supervisor localization will yield local controllers whose collective behavior is identical to the global monolithic supervision.

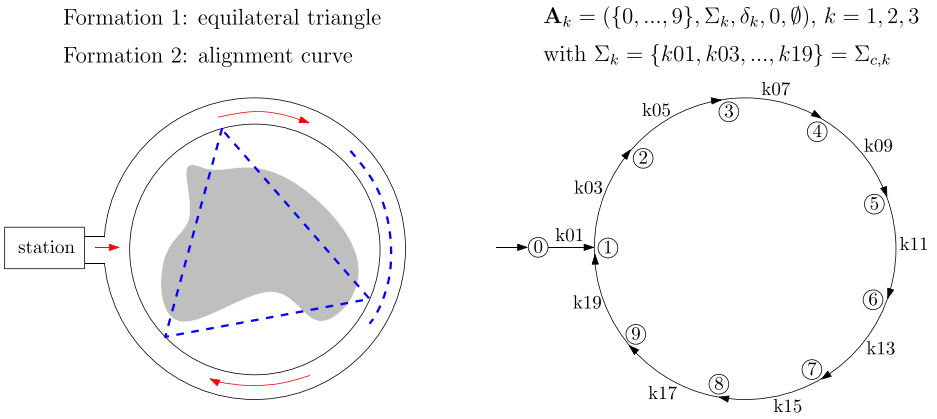


Fig. 1 Example: formations of three agents

Consider the specific example displayed in Fig. 1. Three agents are deployed from a station to cooperatively explore and map an area of interest (e.g. a space station carrying three mobile rovers lands on Mars and the rovers are dispatched to explore the terrain). We first introduce a discrete-event formulation of this formation problem, and then provide solutions by localization to (i) invariance of the desired formation and (ii) shortest paths to the desired formation.

Suppose that owing to the terrain's physical character, agents can traverse only clockwise around a circular route. We discretize the route to get an (identical) generator model $\mathbf{A}_k = (Q_k, \Sigma_k, \delta_k, q_{0,k}, Q_{m,k})$ for each agent $k \in [1, 3]$, displayed on the right of Fig. 1. Here the state set $Q_k = \{0, ..., 9\}$, the event set $\Sigma_k = \{k01, ..., k19\}$, the transition function δ_k is clear from Fig. 1, the initial state $q_{0,k} = 0$ (i.e. agent k in station), and the marker state set $Q_{m,k} = \emptyset$. The reason $Q_{m,k}$ is empty is that we will designate only desired formations, introduced below, as marker states in our plant model.

For simplicity, all events are assumed to be controllable. We then compute[2] the synchronized generator $\mathbf{A} = ||_{k=1}^{3} \mathbf{A}_k = (Q, \Sigma, \delta, q_0, Q_m)$ for the collective behavior of $\mathbf{A}_k$, i.e. $L(\mathbf{A}) = ||_{k=1}^{3} L(\mathbf{A}_k)$ and $L_m(\mathbf{A}) = ||_{k=1}^{3} L_m(\mathbf{A}_k)$; we get

$$\mathbf{A} = (\{0, \ldots, 9\}^3, \{101, \ldots, 319\}, \delta_1 \times \delta_2 \times \delta_3, (0, 0, 0), \emptyset).$$

Now suppose there are two alternative desired formations (dashed patterns in Fig. 1): (i) an equilateral triangle, possibly for omnidirectional scan and mapping of the terrain; and (ii) an alignment curve, possibly for close examination of a hot spot. To respond to the current need, a team leader or a remote operator decides on one formation out of the two; then we will design local control strategies for the agents to reach the target formation and hold this formation hereafter. For formation (i), let

$$Q_1^f = \{(3, 6, 9), (6, 9, 3), (9, 3, 6)\} \subseteq Q,$$

and for formation (ii), let

$$Q_2^f = \{(2, 3, 4), (4, 5, 6), (6, 7, 8)\} \subseteq Q,$$

[2]Computation in this and next sections is by TCT software (Wonham 2013a).

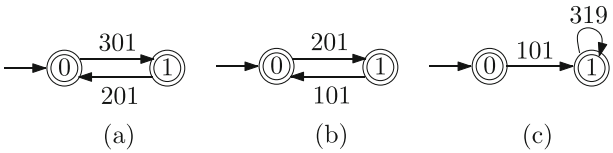


Fig. 2 Specification (i): agents depart from station in the order first $\mathbf{A}_3$, then $\mathbf{A}_2$, and finally $\mathbf{A}_1$. The order is enforced by generators (**a**) and (**b**). Generator (**c**) specifies that $\mathbf{A}_1$ must depart before $\mathbf{A}_3$ finishes its first cycle. All states of the generators are marked. A marked state is denoted by a double circle ◎ in this and all subsequent figures

Since the two formations represent different geometric shapes, the above two sets are disjoint, $Q_1^f \cap Q_2^f = \emptyset$. Moreover, we associate a distinct event $\tau_i \notin \Sigma$, $i = 1, 2$, to the formation set Q_i^f; execution of τ_i signals the leader/operator's decision of choosing formation type i. As will be seen below, event τ_i is used to 'lock' the three agents into the formation Q_i^f, so that they reach Q_i^f and remain there from then on. We let τ_1, τ_2 both be uncontrollable.

Now we define the plant $\mathbf{G}$ of the formation problem by revising $\mathbf{A}$ as follows:

$$\mathbf{G} = \left(\{0, \ldots, 9\}^3, \Sigma \dot{\cup} \{\tau_1, \tau_2\}, \delta \dot{\cup} \delta^f, (0, 0, 0), Q_1^f \dot{\cup} Q_2^f\right),$$

where $\delta^f := \{(q, \tau_1) \mapsto q | q \in Q_1^f\} \dot{\cup} \{(q, \tau_2) \mapsto q | q \in Q_2^f\}$ is a set of selfloop transitions τ_i at each state in Q_i^f, $i = 1, 2$. Note that the marker state set of $\mathbf{G}$ is the union of the two formation sets Q_1^f and Q_2^f.

5.1 Invariance of formation sets

Let $E \subseteq \Sigma^*$ be a specification language imposing behavioral constraints on the plant $\mathbf{G}$. Here we consider two constraints: (i) order of departure for the three agents from station; and (ii) mutual exclusion (i.e. no more than one agent can occupy any given location on the circular route). The generator models of these constraints are displayed in Figs. 2 and 3.

In addition to fulfilling the specification E, we require that the agents reach the subset $Q^f := Q_1^f \dot{\cup} Q_2^f$ of desired formations. For that, we introduce the generator $\mathbf{C}$ displayed in Fig. 4. $\mathbf{C}$ specifies that all strings in Σ^* may occur until an event τ_i, $i = 1, 2$, is executed; after that execution, no event may occur except for τ_i. Note that in the plant $\mathbf{G}$, event τ_i may occur only as a selfloop transition at a state in Q_i^f, corresponding to the desired formation type i. Thus $\mathbf{C}$ effectively renders each state in Q^f 'invariant', in the sense that once a state

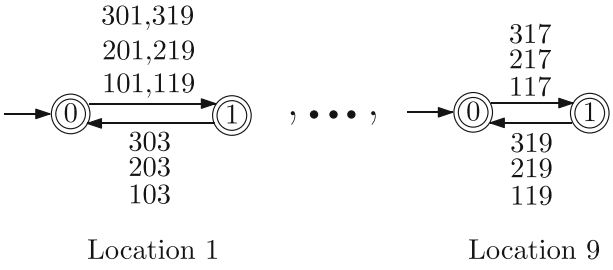


Fig. 3 Specification (ii): mutual exclusion at all 9 locations of the circular route

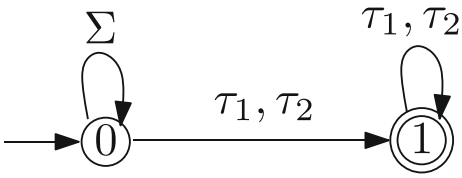


Fig. 4 Specification generator: invariance of desired formation sets

in Q^f is reached and an event τ_i executed (signaling that a desired formation is achieved), then the system will stay in that state thereafter (i.e. the team of agents is locked in that formation).

We compute the monolithic supervisor **SUP** to enforce both specifications E and $L_m(\mathbf{C})$, i.e.

$$L_m(\mathbf{SUP}) = \sup \mathcal{C}(P_e^{-1} E \cap L_m(\mathbf{C}) \cap L_m(\mathbf{G})), \tag{20}$$

where $P_e : (\Sigma \dot{\cup} \{\tau_1, \tau_2\})^* \to \Sigma^*$, and $L(\mathbf{SUP}) = \bar{L}_m(\mathbf{SUP})$. **SUP** has 304 states and 680 transitions; its state size may be reduced to 48 by the supervisor reduction algorithm in Su and Wonham (2004).

Now we apply supervisor localization to decompose **SUP** into a local marker $\mathbf{LOC}_M$ and local controllers $\mathbf{LOC}_\alpha$, one for each controllable event $\alpha \in \Sigma$. For the example, the local marker $\mathbf{LOC}_M$ is simply a generator with a single marked state and all 30 local controllers $\mathbf{LOC}_\alpha$ have state sizes between 2 and 4. Thus, localization achieves considerable

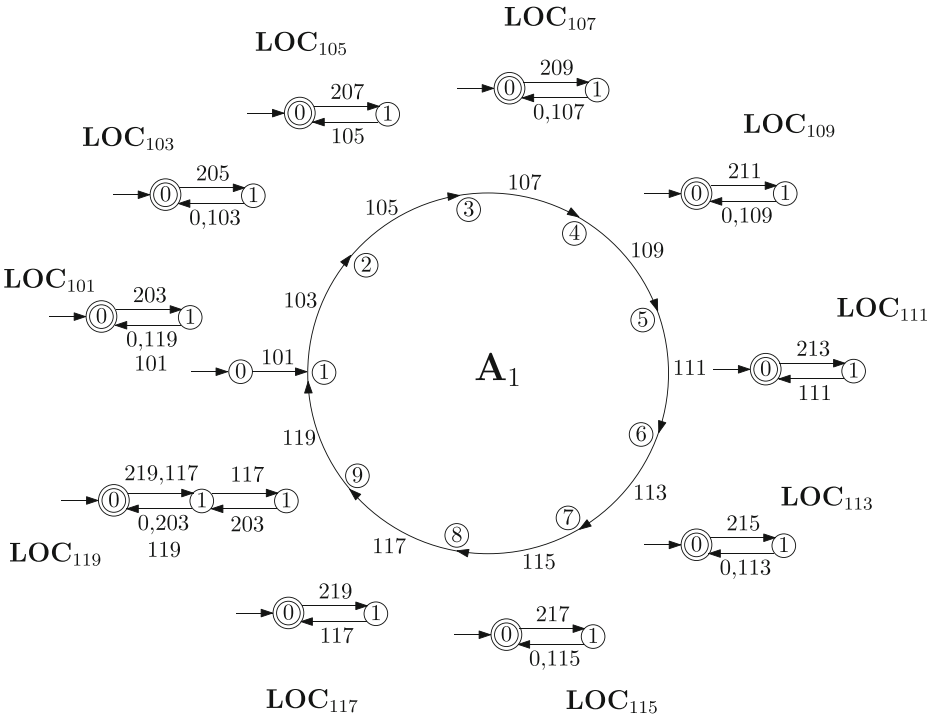


Fig. 5 Local controllers for the events of agent $\mathbf{A}_1$

further state reduction compared to Su and Wonham (2004). Moreover, the control logic of localized controllers becomes transparent. In Fig. 5, we display the local controllers for the events of agent $\mathbf{A}_1$; essentially, the logic of each controller is to ensure mutual exclusion at a corresponding location with agent $\mathbf{A}_2$. Observe that each controller needs certain event information only from agent $\mathbf{A}_2$ (no information is needed from $\mathbf{A}_3$); this is because $\mathbf{A}_1$ follows $\mathbf{A}_2$ in the circular route and can never overtake $\mathbf{A}_2$ because of the mutual exclusion constraint. We also note that when $\mathbf{A}_1$ travels around the route, it uses only one local controller at each step; this is a consequence of computing a local controller for each controllable event. This new feature leads to simpler control logic and further flexibility in controller implementation as compared to the agent-oriented localization in Cai and Wonham (2010a).

5.2 Shortest paths to formation

Having derived the maximally permissive behavior of keeping the invariance of the desired formations Q^f (subject also to specification E), one may aim to realize the shortest paths from the initial state to a formation. In our model, the transitions are not weighted; thus shortest paths refer to the least number of transitions.

Fix $i = 1, 2$. We find the shortest paths to formation type i using the generator $\mathbf{P}_r$, $r \geq 1$, displayed in Fig. 6. $\mathbf{P}_r$ specifies that τ_i is executed for the first time after r transitions of events in Σ. We work up from $r = 1$, to find the least r such that $P_{\tau_i}^{-1} L_m(\mathbf{P}_r) \cap L_m(\mathbf{SUP}) \neq \emptyset$, where $P_{\tau_i} : (\Sigma \dot{\cup} \{\tau_1, \tau_2\})^* \to (\Sigma \dot{\cup} \{\tau_i\})^*$. For that least r, we compute the corresponding monolithic supervisor **OPT** such that

$$L_m(\mathbf{OPT}) = \sup \mathcal{C}\left(P_{\tau_i}^{-1} L_m(\mathbf{P}_r) \cap L_m(\mathbf{SUP})\right), \tag{21}$$

and $L(\mathbf{OPT}) = \bar{L}_m(\mathbf{OPT})$. Thus every string in $L_m(\mathbf{OPT})$ is one of the shortest paths from the initial state to the formation type i.

Using the above method we find that the shortest paths to an equilateral triangle (actually the state $(3, 6, 9) \in Q_1^f$) are of 18 steps (suitable combinations of events 101, 103, 105, 201, ..., 211, 301, ..., 317); and the shortest paths to an alignment curve (actually the state $(2, 3, 4) \in Q_2^f$) are of 9 steps (suitable combinations of events 101, 103, 201, 203, 205, 301, 303, 305, 317). We then apply localization to the monolithic supervisor **OPT** for shortest paths to an alignment curve; the resulting local marker $\mathbf{LOC}_M$ and local controllers $\mathbf{LOC}_\alpha$ are displayed in Fig. 7. We see that (i) the local marker $\mathbf{LOC}_M$ is simply a generator with a single marked state; (ii) agent $\mathbf{A}_3$ moves, without constraint, four steps to location 4 in the circular route, and disables all remaining transitions; (iii) agent $\mathbf{A}_2$ follows agent $\mathbf{A}_3$, moves three steps to location 3, and disables all remaining transitions; and (iv) agent $\mathbf{A}_1$ follows agent $\mathbf{A}_2$, moves two steps to location 2, and disables all remaining

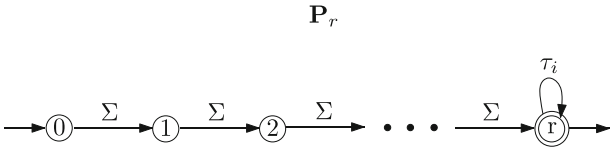


Fig. 6 Specification for finding shortest paths to a desired type of formation

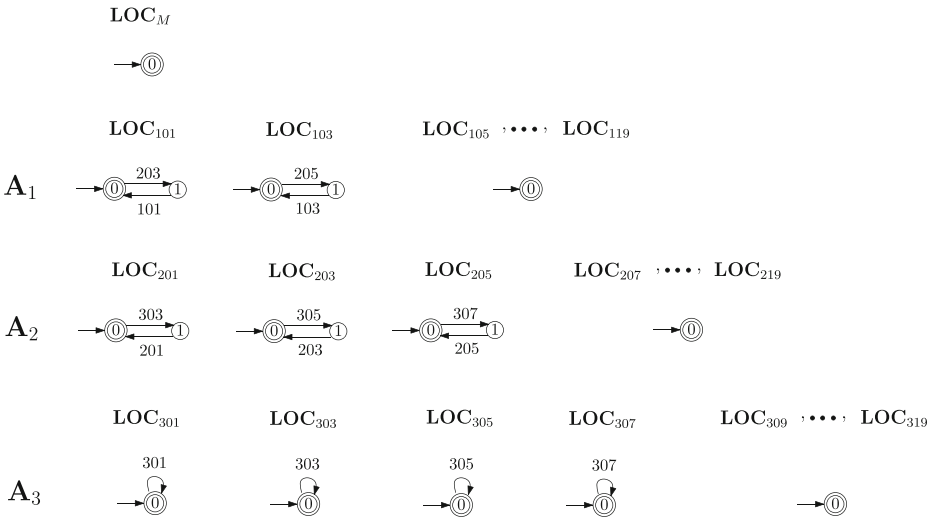


Fig. 7 Local marker and local controllers decomposed from the monolithic supervisor **OPT** for shortest paths to an alignment curve

transitions. Therefore, the agents reach the alignment curve $(2, 3, 4) \in Q_2^f$ after a total of 9 steps, which is the shortest path.

Finally we note that a problem called "convergence" (in shortest paths) has been studied in Brave and Heymann (1990; 1993); Kumar et al. (1993) using concepts of *attraction* and *stability*. Hence the supervisors **SUP** in Eq. 20 and **OPT** in Eq. 21 may be computed using methods reported in those references. The focus of this case study is nevertheless on localized results of these supervisors, and in particular the demonstration of several new features of extended supervisor localization.

6 Cluster tool

In this section, we demonstrate extended supervisor localization on a large-scale system, Cluster Tool. Cluster Tool is an integrated semiconductor manufacturing system used for wafer processing (e.g. Yi et al. 2007); our model is adapted from Su et al. (2010, 2012), with total state size approximately 3.6×10^{11}. Owing to the large scale, we combine localization with an efficient heterarchical approach (Feng and Wonham 2008) in two steps: (1) synthesize a set of decentralized supervisors and coordinators to achieve global optimal and nonblocking supervision; (2) apply localization to decompose each decentralized supervisor/coordinator into local controllers/markers for the relevant controllable events. This combined approach has been demonstrated successfully on benchmark examples in Cai and Wonham (2010a, b); alternative heterarchical methods are e.g. Mohajerani et al. (2011), Schmidt and Breindl (2011), and Su et al. (2012) with which our localization may also be combined.

As displayed in Fig. 8, Cluster Tool consists of (i) two loading docks ($\mathbf{L}_{in}, \mathbf{L}_{out}$) for wafers entering and leaving the system, (ii) eleven vacuum chambers ($\mathbf{C}_{11}, \mathbf{C}_{12}, \ldots, \mathbf{C}_{52}$) where wafers are processed, (iii) four buffers ($\mathbf{B}_1, \ldots, \mathbf{B}_4$) where wafers are temporarily

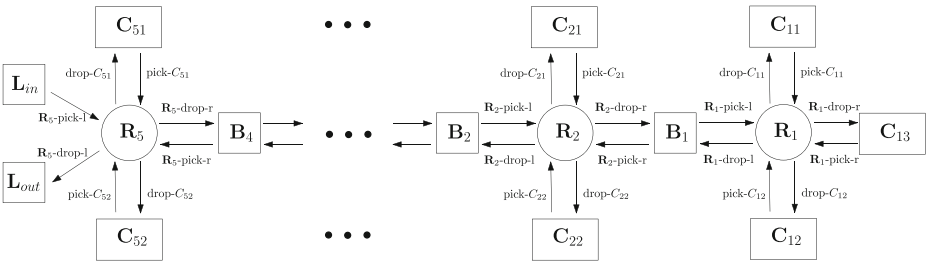


Fig. 8 Cluster Tool: an integrated semiconductor manufacturing system used for wafer processing

stored, and (iv) five robots ($\mathbf{R}_1, \ldots, \mathbf{R}_5$) which transport wafers in the system according to the following production sequence:

$$\mathbf{L}_{in} \rightarrow \mathbf{C}_{51} \rightarrow \mathbf{B}_4 \rightarrow \cdots \rightarrow \mathbf{B}_2 \rightarrow \mathbf{C}_{21} \rightarrow \mathbf{B}_1 \rightarrow \mathbf{C}_{11} \downarrow \mathbf{C}_{13}$$
$$\mathbf{L}_{out} \leftarrow \mathbf{C}_{52} \leftarrow \mathbf{B}_4 \leftarrow \cdots \leftarrow \mathbf{B}_2 \leftarrow \mathbf{C}_{21} \leftarrow \mathbf{B}_1 \leftarrow \mathbf{C}_{12} \downarrow .$$

The five robots are the plant component agents; their generator models are displayed in Fig. 9. Each robot $\mathbf{R}_i$ has 8 events, all assumed controllable; the robots have pairwise disjoint alphabets. The plant is then the synchronous product of the five robot generators.

Next, we describe control specifications for Cluster Tool. (1) Figure 10(a): at each chamber $\mathbf{C}_{ij}$ a wafer is first dropped in by robot $\mathbf{R}_i$, then processed (at state 1 of $\mathbf{C}_{ij}$), and finally picked up again by $\mathbf{R}_i$. Thus a chamber behaves essentially like a one-slot buffer; our first control specification is to protect each $\mathbf{C}_{ij}$ against overflow and underflow. (2) Figure 10(b): each buffer $\mathbf{B}_i$ has capacity one, and may be incremented by $\mathbf{R}_i$ from the right (resp. $\mathbf{R}_{i+1}$ from the left) and then decremented by $\mathbf{R}_{i+1}$ from the left (resp. $\mathbf{R}_i$ from the right). Our second control specification is to protect all buffers against overflow and underflow. Thus far we have described specifications related to physical units–chambers and buffers; the final requirement, denoted by $\mathbf{D}_i$ ($i \in [1, 3]$), is purely logical, and coordinates the operations between neighboring robots. (3) Figure 10(c): once robot $\mathbf{R}_i$

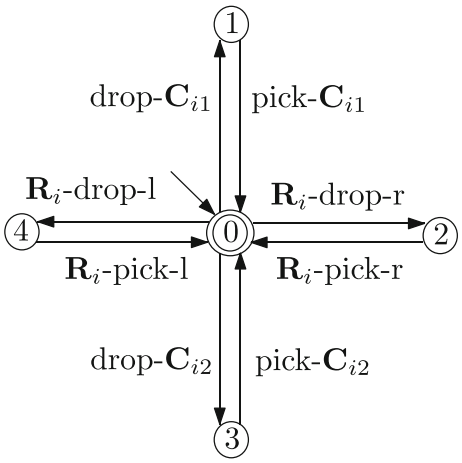


Fig. 9 Plant components

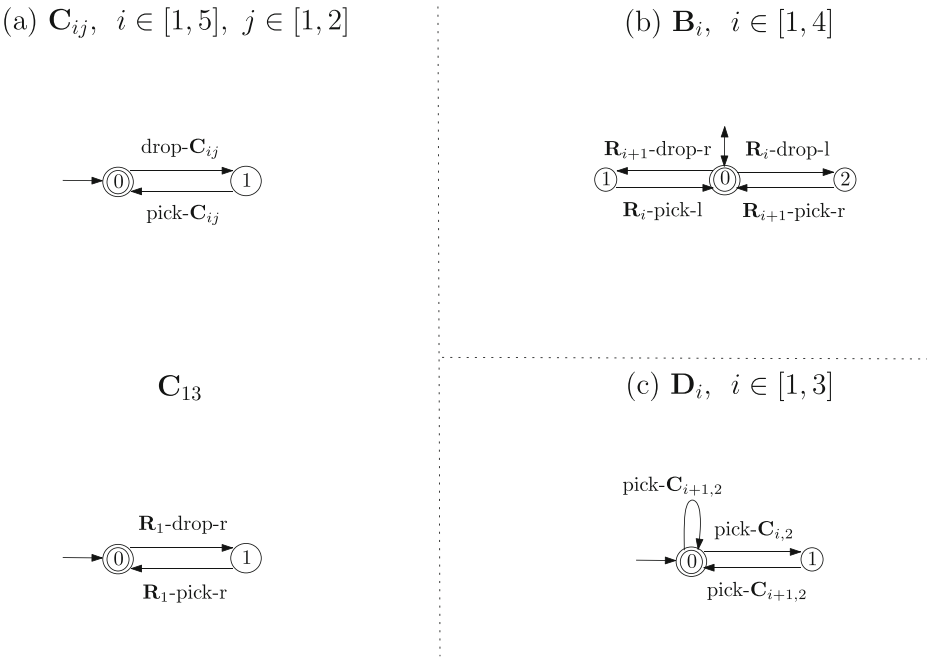


Fig. 10 Control specifications

($i \in [1,3]$) picks up a wafer from chamber $\mathbf{C}_{i,2}$, it may not do so again until robot $\mathbf{R}_{i+1}$ empties chamber $\mathbf{C}_{i+1,2}$. The rationale for imposing this specification is as follows (refer to Fig. 8): once a wafer is picked up by $\mathbf{R}_i$ ($i \in [1,3]$) it needs to be transported through $\mathbf{B}_i \rightarrow \mathbf{R}_{i+1} \rightarrow \mathbf{C}_{i+1,2} \rightarrow \mathbf{R}_{i+1} \rightarrow \mathbf{B}_{i+1}$; here buffers $\mathbf{B}_i, \mathbf{B}_{i+1}$ and robot $\mathbf{R}_{i+1}$ can be viewed as shared resources, and if chamber $\mathbf{C}_{i+1,2}$ is full, then the above wafer transportation may cause blocking. For example, if a wafer gets held by robot $\mathbf{R}_{i+1}$ while chamber $\mathbf{C}_{i+1,2}$ is full, then neither can the wafer be dropped to $\mathbf{C}_{i+1,2}$ nor can $\mathbf{R}_{i+1}$ empty $\mathbf{C}_{i+1,2}$—a deadlock situation. Hence a reasonable requirement to avoid system deadlock is to guarantee an empty slot in $\mathbf{C}_{i+1,2}$ before $\mathbf{R}_i$ initiates the wafer transportation. Note that we do not impose the same specification between $\mathbf{R}_4$ and $\mathbf{R}_5$, because $\mathbf{R}_5$ can drop wafers out of the system without capacity constraint.

On synchronizing plant components (Fig. 9) and control specifications (Fig. 10), the uncontrolled state size is approximately 3.6×10^{11}. Moveover, apart from satisfying all imposed control specifications, the system will require nontrivial coordination to prevent deadlocks caused by conflicts in using multiple shared buffers. Consequently, the overall optimal and nonblocking control of Cluster Tool is a challenging design exercise.

We tackle the problem by using an efficient heterarchical approach (Feng and Wonham 2008) to synthesize a set of decentralized supervisors and coordinators which collectively achieve global optimal and nonblocking supervision. In outline, the steps of this approach are as follows (for details refer to Cai and Wonham 2010a, b; Feng and Wonham 2008). (1) Synthesize an optimal nonblocking decentralized supervisor for each specification with the relevant plant components (based on event sharing); (2) group the decentralized supervisors into subsystems (an effective method is *control-flow net* (Feng and Wonham 2008) if certain special structures are admitted), and design a coordinator for each subsystem if the latter

is blocking; (3) compute an abstraction for each subsystem based on natural projections satisfying *observer* and *output control consistency* properties (Feng and Wonham 2008); (4) group the abstractions into high-level subsystems, and design a coordinator for each high-level subsystem if the latter is blocking; (5) repeat Steps (3) and (4) until a single higher-level subsystem remains in Step (4).

For Cluster Tool, the above procedure yields 18 decentralized supervisors (one computed for each specification in Step (1)) and 4 coordinators, as displayed in Fig. 11. In Step (2) the 18 decentralized supervisors are grouped into 5 subsystems, and in Step (3) one abstraction is computed for each subsystem. These five abstractions are conflicting; in Step (4) we exploit the special structure of Cluster Tool and design four coordinators to solve the conflict. The coordination logic will be explained below. By the heterarchical architectural theory in Feng and Wonham (2008), the decentralized supervisors and coordinators collectively achieve global optimal and nonblocking supervision. We now apply extended supervisor localization to decompose each of the decentralized supervisors into local controllers/markers with respect to the relevant controllable events.

Localizing decentralized supervisors $\mathbf{S}_{\mathbf{C}_{ij}}$ For chamber specifications $\mathbf{C}_{ij}$, we obtain a set of local controllers and (identical) local markers, as displayed in Fig. 12. For each $\mathbf{S}_{\mathbf{C}_{ij}}$ there are two events requiring control action. We explain control logic for the case where $i \in [1, 5]$ and $j = 1$; the other cases are similar. One such event is pick-$\mathbf{C}_{ij}$, which must be disabled ($\mathbf{R}_i$ may not pick up a wafer from chamber $\mathbf{C}_{ij}$) if $\mathbf{C}_{ij}$ is empty; this is to protect chamber $\mathbf{C}_{ij}$ against underflow. The other event requiring control action is $\mathbf{R}_i$-pick-l, which must be disabled ($\mathbf{R}_i$ may not pick up a wafer from left) if chamber $\mathbf{C}_{ij}$ is full. This rule prevents a deadlock situation: if $\mathbf{R}_i$ took a wafer and $\mathbf{C}_{ij}$ were full, then $\mathbf{R}_i$ could neither drop the wafer to $\mathbf{C}_{ij}$ nor pick up a wafer from $\mathbf{C}_{ij}$. The rule at the same time prevents chamber $\mathbf{C}_{ij}$ from overflow. Note that controlling just event drop-$\mathbf{C}_{ij}$ suffices to prevent overflow, but cannot prevent deadlock.

Localizing decentralized supervisors $\mathbf{S}_{\mathbf{B}_i}$ For buffer specifications $\mathbf{B}_i$, we obtain a set of local controllers and (identical) local markers, as displayed in Fig. 13. For each $\mathbf{S}_{\mathbf{B}_i}$ there are six events requiring control action. Events $\mathbf{R}_i$-drop-l and $\mathbf{R}_{i+1}$-drop-r must be disabled ($\mathbf{R}_i$ or $\mathbf{R}_{i+1}$ may not drop a wafer into buffer $\mathbf{B}_i$) when $\mathbf{B}_i$ is full—this is to prevent buffer overflow. On the other hand, events $\mathbf{R}_i$-pick-l and $\mathbf{R}_{i+1}$-pick-r must be disabled ($\mathbf{R}_i$ or $\mathbf{R}_{i+1}$ may not pick up a wafer from buffer $\mathbf{B}_i$) when $\mathbf{B}_i$ is empty—this is to prevent buffer underflow. In addition to preventing buffer overflow and underflow, event pick-$\mathbf{C}_{i2}$ must be

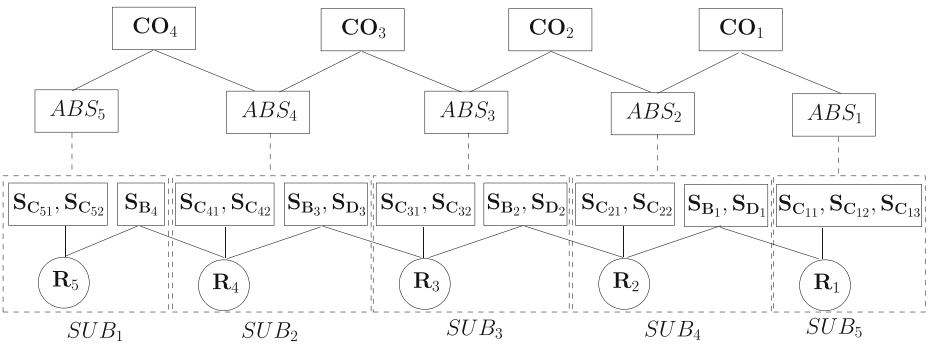


Fig. 11 Heterarchical supervisor synthesis of Cluster Tool

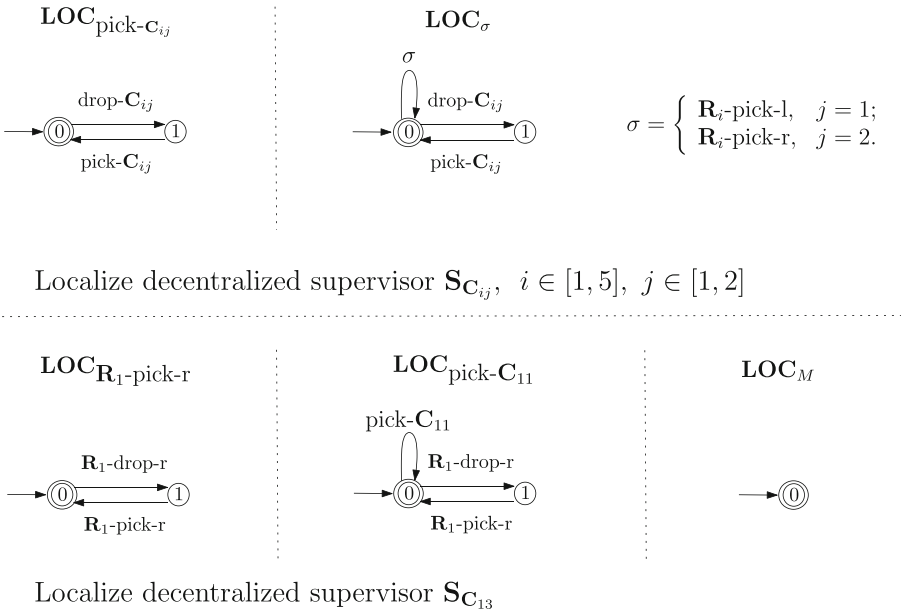


Fig. 12 Local controllers and local marker obtained by localizing decentralized supervisors $\mathbf{S}_{\mathbf{C}_{ij}}$

disabled ($\mathbf{R}_i$ may not pick up a wafer from chamber $\mathbf{C}_{i2}$) unless there is no wafer on the path $\mathbf{R}_i - \mathbf{B}_i - \mathbf{R}_{i+1}$. This logic is to prevent the deadlock situation where both $\mathbf{R}_i$ and $\mathbf{R}_{i+1}$ pick up a wafer to transport through $\mathbf{B}_i$, but neither can do so because the buffer has capacity of only one. For the same reason, event pick-$\mathbf{C}_{i+1,1}$ must be disabled ($\mathbf{R}_{i+1}$ may not pick up a wafer from chamber $\mathbf{C}_{i+1,1}$) unless there is no wafer on the path $\mathbf{R}_i - \mathbf{B}_i - \mathbf{R}_{i+1}$.

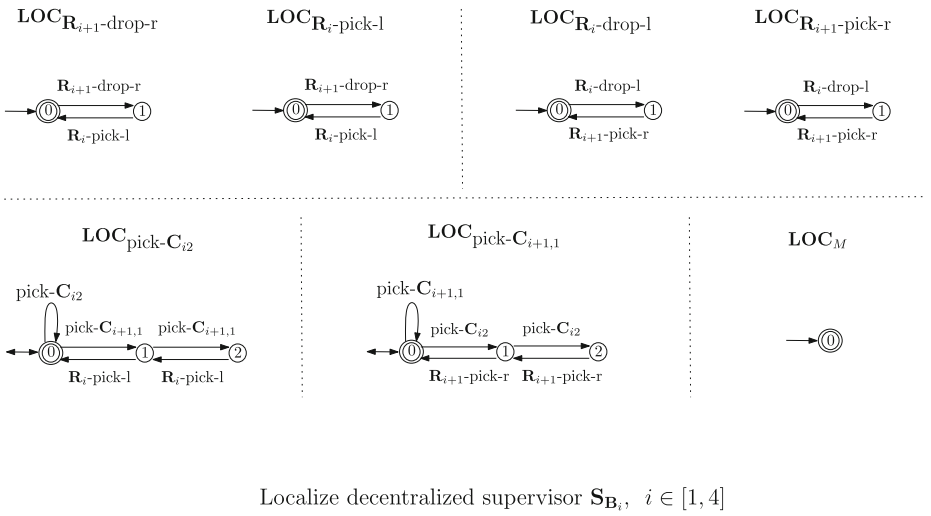


Fig. 13 Local controllers and local marker obtained by localizing decentralized supervisors $\mathbf{S}_{\mathbf{B}_i}$

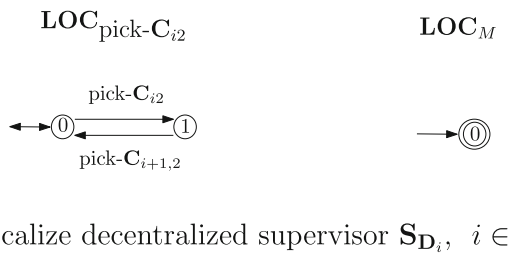


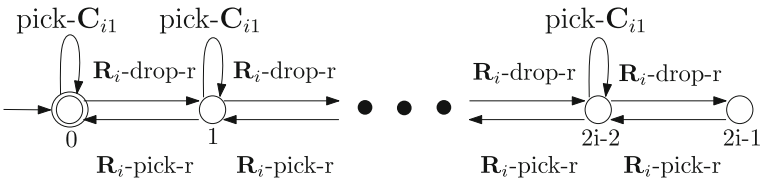
Fig. 14 Local controllers and local marker obtained by localizing decentralized supervisors $\mathbf{S}_{\mathbf{D}_i}$

Localizing decentralized supervisors $\mathbf{S}_{\mathbf{D}_i}$ For logical specifications $\mathbf{D}_i$, we obtain a set of local controllers and (identical) local markers, as displayed in Fig. 14. For each $\mathbf{S}_{\mathbf{D}_i}$ only the event pick-$\mathbf{C}_{i2}$ requires control action: it must be disabled ($\mathbf{R}_i$ may not pick up a wafer from chamber $\mathbf{C}_{i2}$) if the neighboring chamber $\mathbf{C}_{i+1,2}$ is full. This logic is to prevent blocking while wafers are transported from right to left in the system, as explained above when the specifications were imposed.

Coordinators $\mathbf{CO}_i$ The designed coordinators are displayed in Fig. 15. We designed these four coordinators by analyzing the structure of Cluster Tool and the wafer transportation route (Fig. 8); the coordination logic is as follows. Observe in Fig. 8 that once a wafer is picked up from chamber $\mathbf{C}_{i1}$ (i.e. pick-$\mathbf{C}_{i1}$ occurs), it will be transported by robot $\mathbf{R}_i$ to the right, and so all the way to $\mathbf{R}_1$ and then back to the left to $\mathbf{R}_i$—a looping route. For example, when $\mathbf{R}_3$ takes a wafer from $\mathbf{C}_{31}$, the loop is:

$$\mathbf{C}_{31} \xrightarrow{\mathbf{R}_3} \mathbf{B}_2 \xrightarrow{\mathbf{R}_2} \mathbf{C}_{21} \xrightarrow{\mathbf{R}_2} \mathbf{B}_1 \xrightarrow{\mathbf{R}_1} \mathbf{C}_{11} \downarrow \mathbf{R}_1 \quad \mathbf{C}_{13}$$
$$\mathbf{C}_{32} \xleftarrow{\mathbf{R}_3} \mathbf{B}_2 \xleftarrow{\mathbf{R}_2} \mathbf{C}_{21} \xleftarrow{\mathbf{R}_2} \mathbf{B}_1 \xleftarrow{\mathbf{R}_1} \mathbf{C}_{12} \downarrow \mathbf{R}_1.$$

Since the loop has limited capacity to hold wafers, control is needed at the entrance and exit of the loop to prevent 'choking' the loop with too many wafers. The logic of the coordinators in Fig. 15 specifies that event pick-$\mathbf{C}_{i1}$ must be disabled if the number of wafers input exceeds wafers output by $2i - 1$. Note that the loop capacity $2i - 1$ is exactly the number of chambers in the loop; this is because robots and buffers are shared resources, and if all the chambers are full, inputting one more wafer to the loop will clearly cause deadlock.



Coordinators $\mathbf{CO}_i$, $i \in [2, 5]$

Fig. 15 Coordinators $\mathbf{CO}_i$ ($i \in [2, 5]$) each having $2i$ states

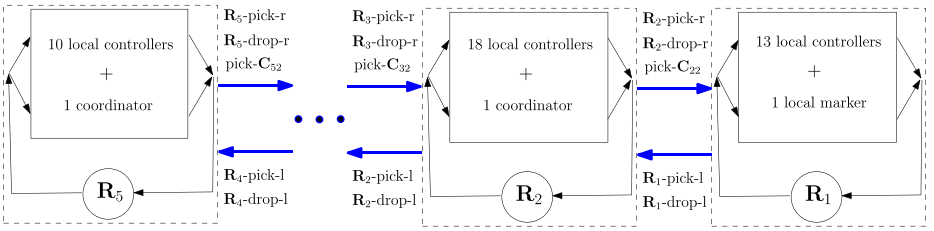


Fig. 16 Distributed control implementation for Cluster Tool: each robot is supervised by its own set of local controllers, as well as interacting (e.g. through event communication) with its immediate left and right neighbors. Robots R_3 and R_4 both have 18 local controllers and 1 coordinator, the same as R_2

We remark that this coordination rule requires global knowledge: for example, to disable event pick-$\mathbf{C}_{51}$, the coordinator $\mathbf{CO}_5$ needs to know a priori the capacity of the whole loop on the right. By knowledge of the loop capacity, however, each coordinator may be implemented locally because it suffices just to count the numbers of wafers input and output to the corresponding loop.

By our main Theorem 5, the collective controlled behavior of the derived local controllers and local markers in Figs. 12–14 is identical to that of the 18 decentralized supervisors; jointly with the 4 coordinators, the collective behavior is therefore identical to the global optimal nonblocking supervision. For control implementation, since the component robots $\mathbf{R}_i$, $i \in [1, 5]$, do not share events, a natural way is to group the local controllers with the robot that owns the corresponding controllable events. For the local marker, which is simply one marked state in all localized results, we associate it with an arbitrary robot, say $\mathbf{R}_1$. Finally, one coordinator $\mathbf{CO}_i$ is associated with one robot $\mathbf{R}_i$, $i \in [2, 5]$, because each $\mathbf{CO}_i$ disables only event pick-$\mathbf{C}_{i1}$. This grouping of local controllers/marker and coordinators yields the distributed control architecture displayed in Fig. 16, where each robot interacts only with its nearest neighbor(s) with the communication events identified. Compared to the agent-oriented localization scheme in Cai and Wonham (2010a) where each robot acquires a single local controller, our new scheme allows further modularization in control allocation, thereby yielding more transparent control logic.

Finally, we remark that in the work of Su et al. (2010, 2012) on Cluster Tool, the primary focus was on reducing computational complexity in achieving global optimal and nonblocking control. There the authors proposed an efficient "distributed supervisor" synthesis, based on abstraction and coordination techniques, which solves the Cluster Tool problem by involving state sizes of order only 10^2 in the computations. It is not clear, however, what the resulting control and coordination rules are. Engineers, on the other hand, demand comprehensible rules for easy implementation and safe management, especially when the plant itself has an intelligible structure; in this case, the system components are connected in a loop. Our supervisor localization, built on an heterarchical control synthesis approach, indeed results in transparent control/coordination rules.

7 Conclusions

We have extended supervisor localization, as developed in Cai and Wonham (2010a), to allow that (i) component agents may share events, (ii) the marking issue is separated from the control issue and can be enforced by an arbitrarily selected agent, and (iii) the event

sets of localized controllers are explicitly defined in general as proper subsets of the entire event set. Moreover, we have demonstrated extended supervisor localization on two case studies: multi-agent formation and Cluster Tool. Our results illustrate that the new features of localization can yield more transparent local control logic and more flexible control implementation.

In future research we aim to study further applications of multi-agent formation. Several deeper problems may be posed: (i) when there are a large number of agents resulting in a large monolithic state set, resort to smart computations, e.g. state tree structures and binary decision diagrams (Cai and Wonham 2012b; Ma and Wonham 2005), to derive local controllers and local marker; (ii) find (if possible) the relation between specified goal formations and the communication structure among agents; and (iii) find (if possible) the tradeoffs between path length to formation and amount of information exchange among agents.

References

Anderson BDO, Yu C, Fidan B, Hendrickx JM (2008) Rigid graph control architectures for autonomous formations. IEEE Control Syst Mag 28(6):48–63

Brave Y, Heymann M (1990) Stabilization of discrete-event processes. Int J Control 51(5):1101–1117

Brave Y, Heymann M (1993) On optimal attraction in discrete-event processes. Inf Sci 67(3):245–276

Cai K, Wonham WM (2010a) Supervisor localization: a top-down approach to distributed control of discrete-event systems. IEEE Trans Autom Control 55(3):605–618

Cai K, Wonham WM (2010b) Supervisor localization for large discrete-event systems—case study production cell. Int J Adv Manuf Technol 50(9–12):1189–1202

Cai K, Wonham WM (2012a) New results on supervisor localization, with application to multi-agent formations. In: Proceedings of the workshop on discrete-event systems. Guadalajara, pp 233–238

Cai K, Wonham WM (2012b) Supervisor localization of discrete-event systems based on state tree structures. In: Proceedings of the 51st IEEE conference on decision and control. Maui, pp 5822–5827

Feng L, Wonham WM (2008) Supervisory control architecture for discrete-event systems. IEEE Trans Autom Control 53(6):1449–1461

Kumar R, Garg V, Marcus SI (1993) Language stability and stabilizability of discrete event dynamical systems. SIAM J Control Optim 31(5):1294–1320

Ma C, Wonham WM (2005) Nonblocking supervisory control of state tree structures. Springer

Mohajerani S, Malik R, Ware S, Fabian M (2011) On the use of observation equivalence in synthesis abstraction. In: Proceedings of the international workshop on dependable control of discrete systems. Saarbrucken, pp 84–89

Pham MT, Seow KT (2012) Discrete-event coordination design for distributed agents. IEEE Trans Autom Sci Eng 9(1):70–82

Ramadge PJ, Wonham WM (1987) Supervisory control of a class of discrete event processes. SIAM J Control Optim 25(1):206–230

Schmidt K, Breindl C (2011) Maximally permissive hierarchical control of decentralized discrete event systems. IEEE Trans Autom Control 56(4):723–737

Seow KT, Pham MT, Ma C, Yokoo M (2009) Coordination planning: applying control synthesis methods for a class of distributed agents. IEEE Trans Control Syst Technol 17(2):405–415

Smith SL, Schwager M, Rus D (2012) Persistent robotic tasks: monitoring and sweeping in changing environments. IEEE Trans Robot 28(2):410–426

Su R, Wonham WM (2004) Supervisor reduction for discrete-event systems. Discrete Event Dyn Syst 14(1):31–53

Su R, van Schuppen JH, Rooda JE (2010) Aggregative synthesis of distributed supervisors based on automaton abstraction. IEEE Trans Autom Control 55(7):1627–1640

Su R, van Schuppen JH, Rooda JE (2012) Maximum permissive coordinated distributed supervisory control of nondeterministic discrete-event systems. Automatica 48(7):1237–1247

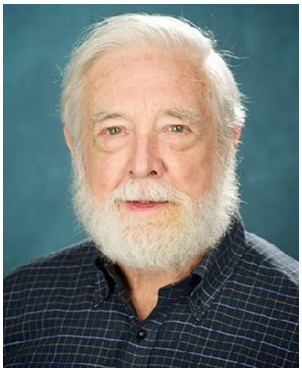
Wonham WM (2013a) Design software: TCT. Systems Control Group, ECE Dept, University of Toronto. Available online at http://www.control.toronto.edu/DES Accessed July 1 2013

Wonham WM (2013b) Supervisory control of discrete-event systems. Systems Control Group, ECE Dept, University of Toronto. Available online at http://www.control.toronto.edu/DES. Accessed July 1 2013

Yi J, Ding S, Zhang MT, van der Meulen P (2007) Throughput analysis of linear cluster tools. In: Proceedings of the 3rd IEEE international conference on automation science and engineering. Scottsdale, pp 1063–1068

Kai Cai received the B. Eng. degree in Electrical Engineering from Zhejiang University (China) in 2006, the M.A.Sc. degree in Electrical and Computer Engineering from the University of Toronto (Canada) in 2008, and the Ph.D. degree in Systems Science from Tokyo Institute of Technology (Japan) in 2011.

From 2011 to 2013 he was a postdoctoral fellow in the University of Toronto, and from 2013 to 2014 an assistant professor in the University of Tokyo. Since April 2014 he joined as an Associate Professor the Urban Research Plaza, Osaka City University. His research interests are distributed control of multi-agent systems, distributed control of discrete-event systems, and theory of control architecture.



W. M. Wonham received the B. Eng. degree in engineering physics from McGill University in 1956, and the Ph.D. in control engineering from the University of 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 of the University of Toronto. Wonham's research interests have included stochastic control and filtering, geometric multivariable control, and discrete-event systems.

He is the author of "Linear Multivariable Control: A Geometric Approach" (Springer-Verlag: 3rd ed. 1985) and co-author (with C. Ma) of "Nonblocking Supervisory Control of State Tree Structures" (Springer-Verlag: 2005).

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