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

K. Cai and W.M. Wonham *

* *Systems Control Group, Department of Electrical and Computer Engineering, University of Toronto, 10 King's College Road, Toronto, ON, M5S 3G4, Canada (e-mail: caikai, wonham@control.utoronto.ca).*

Abstract: A purely distributed control paradigm is proposed for discrete-event systems (DES). In contrast to control by one or more external supervisors, distributed control aims to design built-in strategies for individual agents. First a distributed optimal nonblocking control problem is formulated. To solve it, a top-down localization procedure is developed which systematically decomposes an external supervisor into local controllers while preserving optimality and nonblockingness. An efficient localization algorithm is provided to carry out the computation, and an automated guided vehicles (AGV) example presented for illustration. Finally, the 'easiest' and 'hardest' boundary cases of localization are discussed.

Keywords: discrete-event systems, distributed control, intelligent agents.

1. INTRODUCTION

NEtworked intelligent agents are of ever-increasing importance in control, robotics, and artificial intelligence because of their extensive application domains: AGV systems in manufacturing cells, multi-robot search teams, and software agents on the Internet. To govern this type of system, particular attention has been focused on *distributed control*: each agent has its own local built-in control strategies – but with no external supervisor, thus embodying individual autonomy. Little work has been reported on distributed control of DES in the framework of *supervisory control theory* (SCT) [1].

SCT was initiated by Ramadge and Wonham [2, 3], with cornerstone results of the field established for a monolithic architecture, an organization 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 architecture [4, 5, 6, 7, 8, 9], hierarchical architecture [10, 11], and heterarchical architecture [12, 13, 14]. 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. These architectures are not properly considered to be distributed control, namely a flat system organization where the global functions are performed by the individual agents and not by higher-level, external supervisors. With this in mind, we address the following question: *given a collection of 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 or modular control?*

Only recently has work on distributed control architecture addressing similar questions begun to appear [15, 16, 17]. None of this work, however, deals with both optimal and nonblocking control. The present paper fills that gap. 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 (e.g. [18]); in particular it may refer to decentralized architecture with communicating modular supervisors. 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 on distributed control of DES exploits a top-down approach: first build an external (monolithic or modular) optimal nonblocking supervisor; then decompose the external supervisor into local controllers for individual agents. We call this procedure *supervisor localization*, as displayed in Fig.1.

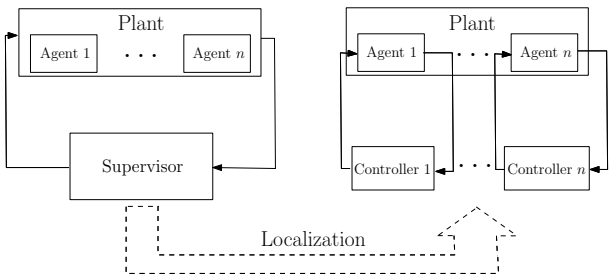

Fig. 1. Supervisor localization

* This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada, Grant No. 7399.

The goal of supervisor localization is first of all to preserve the optimality and nonblockingness of the external supervisor, namely to realize performance identical to that achieved by monolithic or modular control. It is also desired that each localized controller be as 'simple' as possible, so that individual strategies are readily comprehensible. Among diverse criteria of 'simplicity', we focus on the state size. Both goals are achieved by a suitable extension of *supervisor reduction* [19], of which the essence is to 'project' the plant model out of the supervisor model while preserving the controlled behavior. To localize an external 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, as displayed in Fig.1.

The rest of this paper is organized as follows. Section 2 formulates the distributed control problem; Section 3 presents the development and main results of supervisor localization; Section 4 proposes an efficient algorithm for computation; Section 5 illustrates supervisor localization with a familiar AGV example; Section 6 discusses boundary cases of localizability; and Section 7 states our conclusion.

2. PROBLEM FORMULATION

The plant to be controlled is modeled by a generator [1]

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

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* of $\mathbf{G}$ is the sublanguage

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

$\mathbf{G}$ is *nonblocking* if the prefix closure $\overline{L_m(\mathbf{G})} = L(\mathbf{G})$.

We focus on the case where $\mathbf{G}$ consists of component agents $\mathbf{G}^k$ defined over disjoint alphabets Σ^k ($k \in K, K$ an index set):

$$\Sigma = \dot{\bigcup} \{\Sigma^k | k \in K\}$$

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

$$\Sigma_c^k = \Sigma^k \cap \Sigma_c \,,\; \Sigma_u^k = \Sigma^k \cap \Sigma_u$$

Let $k \in K$. 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 the correct local control. Thus, while a local controller's control authority is strictly local, its observation scope need not, and generally will not, be. With local controllers embedded, each agent acquires a strictly local control and generally a non-local observation strategy; the latter is critical to achieve useful synchronization with other agents, thereby ensuring correct local control decisions [1].

The independent components are implicitly coupled through an imposed specification language $E \subseteq \Sigma^*$ that (as usual) imposes a behavioral constraint on $\mathbf{G}$. Recall that a language $F \subseteq \Sigma^*$ is *controllable* (with respect to $\mathbf{G}$) if

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

Now let $\mathcal{C}(E)$ be the set of all controllable sublanguages of E. Then $\mathcal{C}(E)$ contains a (unique) supremal element, denoted by $sup\,\mathcal{C}(E)$ [3] . Let $\mathbf{SUP} = (X, \Sigma, \xi, x_0, X_m)$ be a generator that represents the language $sup\,\mathcal{C}(E \cap L_m(\mathbf{G}))$. $\mathbf{SUP}$ is the monolithic optimal nonblocking supervisor for $\mathbf{G}$ (with respect to E) [2].

Now we formulate the *Distributed Optimal Nonblocking Control Problem* ($\divideontimes$): Given $\mathbf{G}$ and $\mathbf{SUP}$ described above, construct a set of local controllers $\mathbf{LOC} = \{\mathbf{LOC}^k | k \in K\}$, one for each agent, with $L(\mathbf{LOC}) = \bigcap\{L(\mathbf{LOC}^k) | k \in K\}$ and $L_m(\mathbf{LOC}) = \bigcap\{L_m(\mathbf{LOC}^k) | k \in K\}$, such that the following two properties hold:

$$L(\mathbf{G}) \;\cap\; L(\mathbf{LOC}) = L(\mathbf{SUP}) \qquad (1a)$$

$$L_m(\mathbf{G}) \;\cap\; L_m(\mathbf{LOC}) = L_m(\mathbf{SUP}) \qquad (1b)$$

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

For the sake of easy implementation and transparent comprehensibility, it would be desired in practice that the state sizes of local supervisors be appreciably less than that of their 'parent' monolithic supervisor:

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

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

3. SUPERVISOR LOCALIZATION

We solve the distributed control problem ($\divideontimes$) by developing a supervisor localization procedure.

It follows from $\Sigma = \dot{\bigcup}\{\Sigma^k | k \in K\}$ that the set $\{\Sigma_c^k \subseteq \Sigma_c | k \in K\}$ forms a partition on Σ_c. Fix an element $k \in K$. Following [19], we first establish a *control cover* on X, the state space of $\mathbf{SUP}$, based only on control information pertaining to Σ_c^k, as captured by the following four functions. First define $E : X \to 2^\Sigma$ 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 $D^k : X \to 2^{\Sigma_c^k}$ according to

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

Thus $D^k(x)$ is the set of controllable events in Σ_c^k that must be disabled at x. Define $M : X \to \{1, 0\}$ according to

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

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

[2] Throughout the paper we assume that $\mathbf{SUP}$ is nonempty.

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 **G**. Note that for each $x \in X$, we have by $(1b)$ $T(x) = 0 \Rightarrow M(x) = 0$ and $M(x) = 1 \Rightarrow T(x) = 1$.

Definition 1. Let $x, x' \in X$. We say x and x' are *control consistent* (cf [19]) (with respect to Σ_c^k), and write $(x, x') \in \mathcal{R}^k \subseteq X \times X$ if

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

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 **G** (consistent marking information). It should be noted that $\mathcal{R}^k$ need not be transitive in general, and consequently need not be an equivalence relation. This fact leads to the following definition of control cover (with respect to Σ_c^k). First 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 $\mathcal{C}^k = \{X_{i^k}^k \subseteq X | i^k \in I^k\}$ be a cover on X. $\mathcal{C}^k$ is a *control cover* (cf [19, Definition 2.1]) on X (with respect to Σ_c^k) if

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

A control cover $\mathcal{C}^k$ lumps states of **SUP** into (possibly overlapping) cells $X_{i^k}^k (i^k \in I^k)$. 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 some 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 always obtain an induced generator $\mathbf{J}^k = (I^k, \Sigma, \kappa^k, i_0^k, I_m^k)$ by the following construction (cf [19]):

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

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 simply 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.

Let $\mathbf{J} := \{\mathbf{J}^k | k \in K\}$ be the set of all induced generators for the partition $\{\Sigma_c^k \subseteq \Sigma_c | k \in K\}$, with $L(\mathbf{J}) := \bigcap\{L(\mathbf{J}^k) | k \in K\}$ and $L_m(\mathbf{J}) := \bigcap\{L_m(\mathbf{J}^k) | k \in K\}$. Our first result shows that $\mathbf{J}$ is a solution to $(\circledast)$.

Proposition 3. $\mathbf{J}$ is control equivalent to **SUP** with respect to **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. See [20]. ■

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

Definition 4. A generator $\mathbf{LOC} = (Z, \Sigma, \zeta, z_0, Z_m)$ is *normal* (with respect to **SUP**) [19, Definition 2.2] if

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

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

Definition 5. Given generators $\mathbf{LOC} = (Z, \Sigma, \zeta, z_0, Z_m)$ and $\mathbf{J} = (I, \Sigma, \kappa, i_o, I_m)$. **LOC** and **J** are *DES-isomorphic with isomorphism* θ [19, 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) $(\forall i \in I, \sigma \in \Sigma)\kappa(i, \sigma)! \Rightarrow [(\exists z \in Z)\zeta(z, \sigma)! \ \&\ \theta(z) = i]$

Under normality and DES-isomorphism, we have the following result in response to the converse question posed above.

Theorem 6. Let $\mathbf{LOC} := \{\mathbf{LOC}^k = (Z^k, \Sigma, \zeta^k, z_0^k, Z_m^k) | k \in K\}$ be a set of normal generators that is control equivalent to **SUP** with respect to **G**. Then there exists a set of control covers $\mathcal{C} := \{\mathcal{C}^k | k \in K\}$ on X with a corresponding set of induced generators $\mathbf{J} := \{\mathbf{J}^k | k \in K\}$ such that $(\forall k \in K)\ \mathbf{J}^k$ and $\mathbf{LOC}^k$ are DES-isomorphic.

Proof. See [20]. ■

To summarize, every set of control covers generates a solution to $(\circledast)$ (Proposition 3); and every solution to $(\circledast)$ can be induced from some set of control covers (Theorem 6). In particular, a set of *state-minimal* generators can be induced from some set of 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 [19].

4. LOCALIZATION ALGORITHM

It would be desirable to have an efficient algorithm that always computes a set of state-minimal generators, despite its non-uniqueness. Unfortunately, this minimal state problem is NP-hard [19], and consequently we cannot expect a polynomial-time algorithm that can compute a control cover which yields a state-minimal generator.

Nevertheless, a polynomial-time algorithm for supervisor reduction is known [19]. The algorithm generates a control congruence, rather than a control cover, and empirical evidence is given showing 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 **SUP** $= (X, \Sigma_c \dot{\cup} \Sigma_u, \_, \_, \_)$ 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., $\mathcal{C}^k_{init} = \{[x] \subseteq X | [x] = \{x\}\}$, where $[x]$ denotes the cell in $\mathcal{C}^k$ to which x belongs. Then LA merges $[x]$ and $[x']$ into one cell if x and x', as well as all their corresponding future states reachable by identical strings, are control consistent. This *mergibility* condition is checked by lines 14 and 19 in the pseudocode displayed below: line 14 checks control consistency for the current state pair (x, x') and line 19 recursively checks consistency for all their related future states. 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 18 – to eliminate redundant mergibility tests as well as element overlapping in $\mathcal{C}^k$. LA loops until all of the states are checked.

Localization Algorithm (LA) [3]

```
1  int main()
2  for i : 0 to n − 2 do
3      if i > min{m|x_m ∈ [x_i]} then continue;
4      for j : i + 1 to n − 1 do
5          if j > min{m|x_m ∈ [x_j]} then continue;
6          wl = ∅;
7          if Check_Mergibility(x_i, x_j, wl, i) = T then
           C^k = {[x] ∪
           ⋃_{x':{(x,x'),(x',x)}∩wl≠∅} [x'] | [x], [x'] ∈ C^k}
8      end
9  end
10 bool Check_Mergibility(x_i, x_j, wl, cnode)
11 for each x_p ∈ [x_i] ∪ ⋃_{x:{(x,x_i),(x_i,x)}∩wl≠∅} [x] do
12     for each x_q ∈ [x_j] ∪ ⋃_{x:{(x,x_j),(x_j,x)}∩wl≠∅} [x] do
13         if {(x_p, x_q), (x_q, x_p)} ∩ wl ≠ ∅ then continue;
14         if (x_p, x_q) ∉ R^k then return F;
15         wl = wl ∪ {(x_p, x_q)};
16         for each σ ∈ Σ with ξ(x_p, σ)!, ξ(x_q, σ)! do
17             if [ξ(x_p, σ)] = [ξ(x_q, σ)] ∨
               {(ξ(x_p, σ), ξ(x_q, σ)), (ξ(x_p, σ), ξ(x_q, σ))}∩
               wl ≠ ∅ then continue;
18             if min{m|x_m ∈ [ξ(x_p, σ)]} < cnode ∨
               min{m|x_m ∈ [ξ(x_q, σ)]} < cnode then
               return F;
19             if Check_Mergibility(ξ(x_p, σ), ξ(x_q, σ),
               wl, cnode) = F then return F;
20         end
21     end
22 end
23 return T;
```

[3] Notation: $X = \{x_0, \ldots, x_{n-1}\}$ is an ordering of states. $wl \subseteq X \times X$ is a list of state pairs whose mergibility is pending. T, F denote *true*, *false*.

Remark 7. LA preserves all computational properties of the reduction algorithm in [19] – LA terminates, generates a control congruence, and has time complexity $O(n^4)$, where n is the state size of **SUP**. For an example that illustrates LA see [20].

5. DISTRIBUTED CONTROL OF AN AGV SYSTEM

We apply the supervisor localization procedure to solve the distributed control problem of AGV serving a manufacturing workcell, taken from [1]. The results are computed by the proposed localization algorithm (implemented in a C++ program); the desired control equivalence between the set of local controllers and the optimal nonblocking supervisor is verified in TCT [21], by confirming

isomorph(**meet**($\{\mathbf{LOC}^k | k \in K\}$, **G**),**SUP**) = TRUE

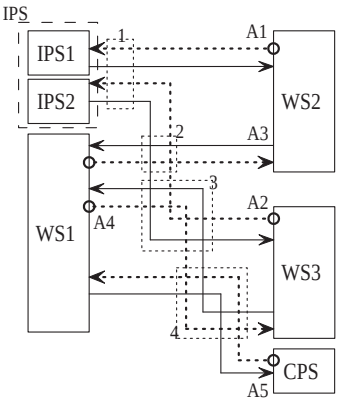

Fig. 2. AGV: system configuration

The manufacturing workcell (displayed in Fig. 2) consists of two input 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,...,AGV5 – travel in fixed interleaving 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 mutual exclusion of the shared loading area of the input stations. Readers are referred to [1, Section 4.7] for generator models of plant components and specifications, as well as the detailed description of events. The distributed control objective is to design for each AGV a set of local strategies – but with no external supervisors.

The monolithic approach generates a monolithic supervisor of 4406 states [1]. We localize this global supervisor with respect to each AGV: the resultant local controllers have 23, 44, 13, 20, 10 states respectively. However, since the computation complexity of our localization algorithm is $O(n^4)$, where n is the state size of the supervisor, it is inefficient to directly localize the central supervisor. In addition, with local controllers having the state sizes listed above, individual control logics remain hard to understand.

Instead, we combine our supervisor localization with decentralized control theory; namely, we localize decentralized supervisors, in general of smaller state size, to the relevant agents. The resultant local controllers can achieve control equivalence with the monolithic supervisor as long as the decentralized

supervisors do [4]. For this example, we employ the decentralized solution presented in [14]: decentralized supervisors, one corresponding to each control specification, are first synthesized by using the standard method; denote them by **Z1SUP**, **Z2SUP**, **Z3SUP**, **Z4SUP**, **WS13SUP**, **WS14SUP**, **WS2SUP**, **WS3SUP**, and **IPSUP**, respectively. Since this example is prone to deadlock, a coordinator **CO** must be designed; we do so using control flow nets and model abstraction by natural observer [14]. This approach is computationally efficient, and the solution has been verified to be control equivalent to the optimal nonblocking monolithic supervisor. More details are provided in Table 1.

Table 1. Decentralized solution in [14]

	AGVs related	State #	Reduced State #
Z1SUP	1, 2	24	2
Z2SUP	2, 3	24	2
Z3SUP	2, 4	36	2
Z4SUP	4, 5	18	2
WS13SUP	3, 5	24	2
WS14SUP	4, 5	34	2
WS2SUP	1, 3	24	2
WS3SUP	2, 4	62	2
IPSUP	1, 2	24	2
CO	1, 2	165	7

Next we localize each supervisor to its associated AGVs, respectively. The state transition diagrams of resultant local controllers are displayed in Figs. 3–7 (for clarity extraneous self-loops 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.

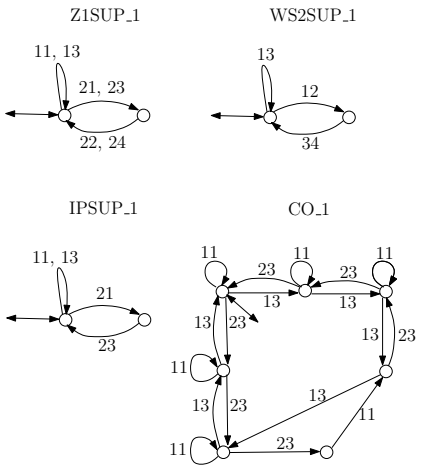

Fig. 3. Local controllers for AGV1

[4] This statement is proved in [20].

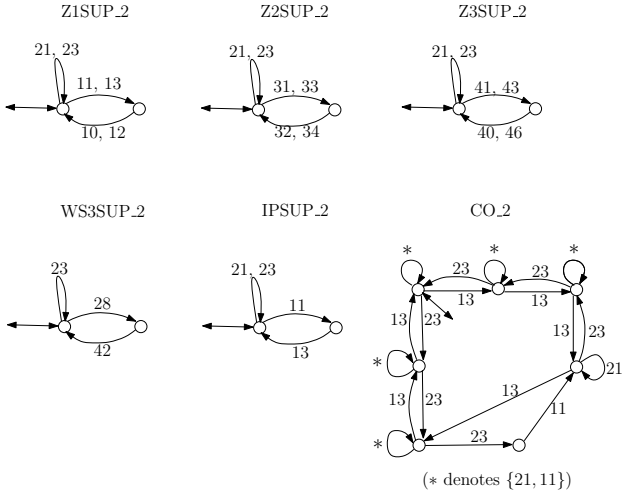

Fig. 4. Local controllers for AGV2

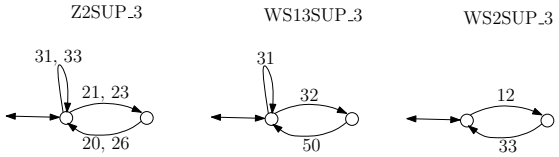

Fig. 5. Local controllers for AGV3

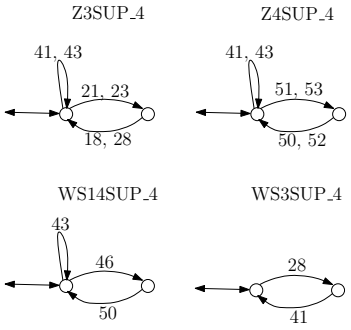

Fig. 6. Local controllers for AGV4

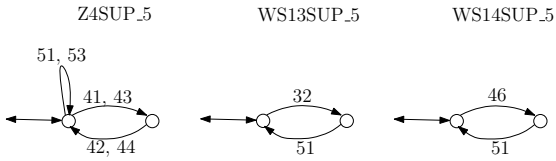

Fig. 7. Local controllers for AGV5

6. 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.

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

Definition 8. **SUP** is *fully-localizable* if there exists a set of local controllers $\{\mathbf{LOC}^k | k \in K\}$ that is control equivalent to **SUP** such that for every $k \in K$, $L(\mathbf{LOC}^k) = P_k^{-1}(L^k)$ for some $L^k \subseteq (\Sigma^k)^*$.

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

Proposition 9. If for all $p \in P$ there is $k \in K$ such that $E_p \subseteq (\Sigma^k)^*$, then **SUP** is fully-localizable.

Proof. Follows from the assumption that $\Sigma^k (k \in K)$ are pairwise disjoint and Definition 8. ■

The assumption of Proposition 9 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. [22]).

6.2 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 10. In Fig. 8, 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 **SUP** is a monolithic supervisor which enforces the mutual exclusion specification. Then by applying the localization algorithm to **SUP**, we generate for agent $\mathbf{A}_i$ a local controller $\mathbf{LOC}^i$. However, both local controllers are nothing but the same as **SUP**; namely, our supervisor localization accomplished nothing useful.

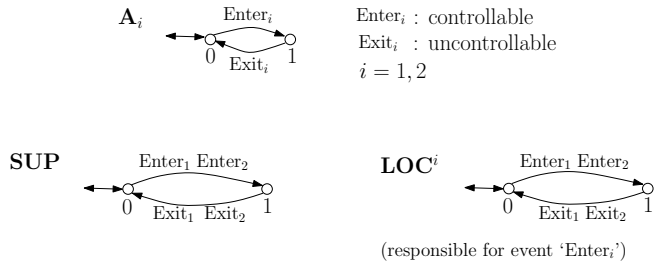

Fig. 8. Example: non-localizable

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

Definition 11. Let $\mathbf{MLOC}^k$ be a state-minimal local controller for agent $\mathbf{G}^k$ (defined over $\Sigma^k \subseteq \Sigma$). **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 further merged, which in turn implies that the transition structure will remain the same.

By Theorem 6, $\mathbf{MLOC}^k$ 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

(3) $(x, x') \notin \mathcal{R}^k$

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

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

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

The above discussion has proved the following fact.

Proposition 12. $|\mathbf{MLOC}^k| = \Omega_{\mathcal{C}^k}$. ■

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

Proposition 13. **SUP** is non-localizable (with respect to $\Sigma_c^k \subseteq \Sigma$) if and only if $|\mathbf{SUP}| = \Omega_{\mathcal{C}^k}$

Proof. Follows from Definition 11 and Proposition 12. ■

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 [23]. In particular, the implemented polynomial-time algorithm that computes *lower bound estimate* (lbe) in [19, 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 14. 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 12. ■

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 10, and applying the adopted algorithm from [19], 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 the copies of **SUP** as their local controllers.

7. CONCLUSION

We have formulated a distributed control problem and presented a top-down approach, supervisor localization, that solves the problem. A polynomial-time algorithm has been proposed to carry out the computation and an AGV example has been discussed for illustration. In addition, we have elucidated two boundary cases of the localization problem.

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," Systems Control Group, ECE Dept, University of Toronto, http://www.control.toronto.edu/DES, updated July 1, 2008.

[2] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal of Control and Optimization*, 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 Journal of Control and Optimization*, vol. 25, no. 3, pp. 637–659, 1987.

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

[5] K. Rudie and W. M. Wonham, "Think globally, act locally: Decentralized supervisory control," *IEEE Transactions on Automatic Control*, vol. 37, no. 11, pp. 1692–1708, 1992.

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

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

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

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

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

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

[12] K. Schmidt, T. Moor, and S. Park, "A hierarchical architecture for nonblocking control of decentralized discrete event systems," in *Proc. 13th Mediterranean Conference on Control and Automation*, Limassol, Cyprus, Jun 2005, pp. 902–907.

[13] R. J. Leduc, M. Lawford, and W. M. Wonham, "Hierarchical interface-based supervisory control - parallel case," *IEEE Transactions on Automatic Control*, vol. 50, no. 9, pp. 1336–1348, 2005.

[14] L. Feng and W. M. Wonham, "Computationally efficient supervisory design: Abstraction and modularity," in *Proc. Int. Workshop Discrete Event Systems (WODES06)*, Ann Arbor, Michigan, U.S.A., Jul 2006, pp. 3–8.

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

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

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

[18] S. Lafortune, "On decentralized and distributed control of partially-observed discrete event systems," in *Advances in Control Theory and Applications*, L. M. C. R. C. Bonivento, A. Isidori, Ed. Springer Berlin / Heidelberg, 2007, vol. 353, pp. 171–184.

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

[20] K. Cai, "Supervisor Localization: A Top-Down Approach to Distributed Control of Discrete-Event Systems," Master's thesis, ECE Dept, University of Toronto, 2008, available at http://www.control.toronto.edu/DES/CaiKai_MASc_Thesis.pdf.

[21] W. M. Wonham, "Design software: XPTCT," Systems Control Group, ECE Dept, University of Toronto, http://www.control.toronto.edu/DES, Version 121, Windows XP, updated July 1, 2008.

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

[23] P. M. Pardalos and J. Xue, "The maximum clique problem," *Global Optimization*, vol. 4, no. 3, pp. 301–328, Apr 1994.