Joint 48th IEEE Conference on Decision and Control and
28th Chinese Control Conference
Shanghai, P.R. China, December 16-18, 2009

Supervisor Localization for Large-Scale Discrete-Event Systems

K. Cai and W.M. Wonham

***Abstract*— We propose a top-down approach, called *supervisor localization*, to distributed control of large-scale discrete-event systems (DES). The essence of this approach is to allocate, or *localize*, 'external' supervisory control action to individual plant components as their 'internal' control strategies. To cope with state explosion of large DES, we develop a decomposition-aggregation procedure that employs a flexible decentralized and hierarchical architecture which reduces computational effort in localization. This procedure is demonstrated in detail on a benchmark Production Cell that has state size of order 10^8.**

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

I. INTRODUCTION

In [1], [2], [3] we proposed a distributed control paradigm for DES consisting of coupled components. The plant to be controlled is assumed to comprise independent asynchronous components which are coupled implicitly through control specifications. The objective of distributed control is to allocate control action to each individual component such that the resulting 'private' or *localized* controllers collectively achieve optimal (i.e., minimally restrictive) and nonblocking controlled behavior for the whole system.

Distinct, though related, architectures are decentralized, hierarchical, and heterarchical (e.g., [4], [5]). Both distributed and these modular approaches aim to achieve efficient computation and transparent control logic, while realizing global optimality and nonblockingness. A structural distinction, however, is that with modular supervision, the 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 (i.e., plant) components.

To address this type of distributed control, we developed in [1], [2], [3] a procedure called *supervisor localization* (**SL**): First synthesize the (optimal and nonblocking) monolithic supervisor [6]; then decompose this supervisor into local controllers for individual active components. It is then proved that the family of local controllers (one for each component) provides the same global control action as the monolithic supervisor did – and is therefore optimal and nonblocking for the entire system. We note that a recent paper [7] proposed a scheme similar in general terms to our own, but provided a control synthesis which amounts merely to making copies of the (reduced) monolithic supervisor for each component, with some corresponding, extraneous selfloops removed. By contrast, our **SL** procedure exploits *supervisor reduction* [8] and achieves a 'truly local' result.

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

An underlying assumption of **SL** is that the monolithic supervisor can be feasibly computed; this, however, may not be the case for 'large-scale' DES owing to state explosion. Nevertheless, it is proposed in [1], [2] that we may in principle manage the complexity of large DES by combining **SL** with existing efficient modular control theories. In the present paper this combination is formalized as a *decomposition-aggregation procedure* (**DAP**): First design an organization of modular supervisors that achieves global optimality and nonblockingness; then apply **SL** to decompose each of these modular supervisors into local controllers for the relevant components. The novel contribution of this paper is the demonstration that our **SL** algorithm [1], [2], [3] and the modular theory in [5] can be combined to provide an effective attack on large systems. In Section II we formulate the distributed control problem for large-scale DES, and present **DAP** as a corresponding solution in Section III. Then in Sections IV and V we demonstrate **DAP** on a benchmark application – the Production Cell – having state size of order 10^8. Finally, we state our conclusion in Section VI.

II. DISTRIBUTED CONTROL PROBLEM

The plant to be controlled is modeled by a (nonempty) *generator* $\mathbf{G}$ defined over a (finite) *alphabet* Σ, with *closed* and *marked behaviors* $L(\mathbf{G})$ and $L_m(\mathbf{G})$, respectively [6]. As emphasized we deal only with $\mathbf{G}$ comprised of asynchronous components $\mathbf{G}^k$ ($k \in \mathcal{K}, \mathcal{K}$ an index set), called 'active'. Namely, the $\mathbf{G}^k$ are defined over pairwise disjoint alphabets Σ^k, and $\Sigma = \dot{\bigcup}\{\Sigma^k | k \in \mathcal{K}\}$. Let $L_k := L(\mathbf{G}^k)$ and $L_{m,k} := L_m(\mathbf{G}^k)$; then we have

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

where "||" denotes *synchronous product* [6]. For simplicity we assume that for every $k \in \mathcal{K}$, $\mathbf{G}^k$ is *nonblocking* (i.e., the *prefix-closure* $\bar{L}_{m,k} = L_k$). Then $\mathbf{G}$ is necessarily nonblocking (i.e., $\bar{L}_m(\mathbf{G}) = L(\mathbf{G})$).

With $\Sigma = \Sigma_c \dot{\cup} \Sigma_u$, the controllable and uncontrollable event subsets, we assign control structure to each component:

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

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

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

As to its observation, in nontrivial cases [1] $\mathbf{LOC}^k$ observes, and responds to, events generated by components other than

[1] See [1, Sect. 2.6], [2, Sect. 6.1] for the trivial case where no coupling among components is imposed by control specifications.

978-1-4244-3872-3/09/$25.00 ©2009 IEEE

$\mathbf{G}^k$, thereby ensuring correct local control decisions. Thus, while a local controller's control authority is strictly local, its observation scope need not, and generally will not, be [2].

The components are implicitly coupled through a control specification language E that imposes behavioral constraints on $\mathbf{G}$. As in the literature (e.g., [4]) assume that E is *decomposable* into specifications $E_p \subseteq \Sigma_{o,p}^*$ ($p \in \mathcal{P}$, $\mathcal{P}$ an index set), where the $\Sigma_{o,p} \subseteq \Sigma$ need not be pairwise disjoint; namely $E = ||\{E_p | p \in \mathcal{P}\}$. Thus E is defined over $\Sigma_o := \bigcup\{\Sigma_{o,p} | p \in \mathcal{P}\}$. Let $P_o : \Sigma^* \to \Sigma_o^*$ be the corresponding *natural projection*, and write $P_o^{-1} : Pwr(\Sigma_o^*) \to Pwr(\Sigma^*)$ for the inverse-image function of P_o, where $Pwr(\cdot)$ denotes powerset [6].

Let $F \subseteq \Sigma^*$, and recall that F is *controllable* (with respect to $\mathbf{G}$) if $\bar{F}\Sigma_u \cap L(\mathbf{G}) \subseteq \bar{F}$ (where $\bar{F}$ is the prefix-closure of F). Whether or not F is controllable, we denote by $\mathcal{C}(F)$ the family of all controllable sublanguages of F. Then $\mathcal{C}(F)$ is nonempty and contains a (unique) supremal element, denoted $sup\mathcal{C}(F)$ [6].

For the plant $\mathbf{G}$ and the specification E described above, let a generator **SUP** (over Σ) be the corresponding monolithic supervisor that is optimal and nonblocking. Since we are concerned with large-scale DES, assume that **SUP** is not feasibly computable. The marked language of **SUP** can, nevertheless, be expressed algebraically as

$$L_m(\mathbf{SUP}) = sup\mathcal{C}(P_o^{-1}E \cap L_m(\mathbf{G})).$$

Now we formulate the distributed control problem ($\divideontimes$): Construct for each component $\mathbf{G}^k$ ($k \in \mathcal{K}$) a family of local controllers $\mathbf{LOC}^k = \{\mathbf{LOC}_{i_k}^k | i_k \in \mathcal{I}_k\}$ ($\mathcal{I}_k$ an index set), with $L(\mathbf{LOC}^k) = \bigcap\{L(\mathbf{LOC}_{i_k}^k) | i_k \in \mathcal{I}_k\}$ and $L_m(\mathbf{LOC}^k) = \bigcap\{L_m(\mathbf{LOC}_{i_k}^k) | i_k \in \mathcal{I}_k\}$. Further let $\mathbf{LOC} = \{\mathbf{LOC}^k | k \in \mathcal{K}\}$ be the set of all local controllers, with $L(\mathbf{LOC}) = \bigcap\{L(\mathbf{LOC}^k) | k \in K\}$ and $L_m(\mathbf{LOC}) = \bigcap\{L_m(\mathbf{LOC}^k) | k \in K\}$. It is required that

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

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

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

To progress, we need to review two concepts that will be exploited in Section III. The natural projection $P_o : \Sigma^* \to \Sigma_o^*$ is *output control consistent* (OCC) [5] for $L(\mathbf{G})$ if for every string $s \in L(\mathbf{G})$ of the form $s = s'\sigma_1 \cdots \sigma_k$ ($k \geq 1$) where s' is either the empty string or terminates with an event in Σ_o, the following holds:

$$\big((\forall i \in [1, k-1])\, \sigma_i \in \Sigma - \Sigma_o\big) \;\; \& \;\; \sigma_k \in \Sigma_o \cap \Sigma_u \Rightarrow \big((\forall j \in [1, k])\, \sigma_j \in \Sigma_u\big).$$

By this definition, ensuring that P_o is OCC amounts to ensuring that for each uncontrollable event σ in Σ_o, σ's nearest 'upstream' controllable events are also in Σ_o.

The second concept is that of *natural observer*: P_o is called an $L_m(\mathbf{G})$-*observer* [5] if

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

In case P_o does not enjoy the observer property, we employ the *minimal extension* (MX) algorithm developed in [9] to 'reasonably' extend Σ_o so that the augmented observable subset does define an $L_m(\mathbf{G})$-observer.

III. Solution Procedure

We solve the distributed control problem ($\divideontimes$) for large-scale DES by combining the supervisor localization (**SL**) [1], [2], [3] with an efficient modular control theory [5]. Our solution is the decomposition-aggregation procedure (**DAP**) consisting of the following seven steps.

1) Plant Model Abstraction: Part of the plant dynamics that is unrelated to the imposed specification may be concealed. By hiding irrelevant transitions, we can simplify the models of components. The procedure is as follows [5].

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

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

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

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

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

2) Decentralized Supervisor Synthesis: The system now consists of component model abstractions $(\mathbf{G}^k)'$ ($k \in \mathcal{K}$) and specifications $E_p \subseteq \Sigma_{o,p}^*$ ($p \in \mathcal{P}$). Since each E_p may impose coupling only on a subset of component abstractions, a decentralized supervisor with respect to E_p, denoted by $\mathbf{SUP}_p$, can be obtained with only those relevant abstractions. Specifically, we associate with each E_p its *event-coupled* abstractions: those sharing events with E_p (i.e., $(\Sigma^k)' \cap \Sigma_{o,p} \neq \emptyset$); and then, we synthesize a corresponding optimal nonblocking decentralized supervisor $\mathbf{SUP}_p$ [5, Theorem 2].

3) Subsystem Decomposition and Coordination: After synthesizing decentralized supervisors, we view the whole system as comprised of a set of modules $\{\mathcal{M}_p \mid p \in \mathcal{P}\}$, each $\mathcal{M}_p$ consisting of a decentralized supervisor $\mathbf{SUP}_p$ with associated component model abstractions. In this step, we decompose the overall system into small-scale subsystems, through grouping these modules based on their interconnection dependencies (e.g., event-coupling). If the modules admit certain special structures, *control-flow net* [10] is an effective approach for subsystem decomposition.

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

Having obtained a group of small subsystems, we verify the nonblocking property for each of them [3]. If a subsystem happens to be blocking, we design a *coordinator* [4] to resolve the conflict by employing a method proposed in [5, Proposition 7 and Theorem 4].

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

(i) Determine the shared event set, denoted by Σ_{sub}, of these subsystems. Let $P_{sub} : (\Sigma'_o)^* \rightarrow \Sigma^*_{sub}$ be the corresponding natural projection.
(ii) Check if P_{sub} is OCC for every subsystem; if not, for each $\sigma \in \Sigma_{sub} \cap \Sigma_u$ add its nearest upstream controllable events to Σ_{sub}. Denote the augmented alphabet by Σ'_{sub}, and let $P'_{sub} : (\Sigma'_o)^* \rightarrow (\Sigma'_{sub})^*$.
(iii) Check if P'_{sub} is an observer for every subsystem. If so, go to (iv); otherwise, employ the MX algorithm to compute an extension of Σ'_{sub} that does define an observer for every subsystem. Denote the extended alphabet again by Σ'_{sub}, and the corresponding natural projection again by P'_{sub}. Return to (ii).
(iv) Compute abstractions for each subsystem with P'_{sub}.

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

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

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

7) Decentralized Supervisors and Coordinators Localization: We now apply the **SL** algorithm to localize each of these decentralized supervisors $\mathbf{SUP}_p$ ($p \in \mathcal{P}$) and coordinators $\mathbf{CO}_q$ ($q \in \mathcal{Q}$) to local controllers for their relevant components. First, we bring in a criterion to determine if a component $\mathbf{G}^k$ ($k \in \mathcal{K}$) is related to $\mathbf{SUP}_p$ or $\mathbf{CO}_q$. For $\mathbf{SUP}_p$

[3]We use TCT [11] procedure **nonconflicting** for this verification.

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

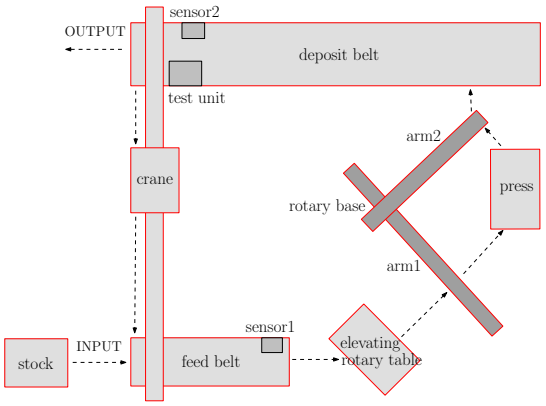

Fig. 1. Production cell

with *state set* X_p define $D^k : X_p \rightarrow Pwr(\Sigma^k_c)$ according to

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

Thus D^k associates each state $x \in X_p$ with the subset of controllable events of $\mathbf{G}^k$ that must be disabled at x. We then say $\mathbf{G}^k$ is *control-coupled* to $\mathbf{SUP}_p$ if $(\exists x \in X_p)\ D^k(x) \neq \emptyset$; in other words, $\mathbf{SUP}_p$ disables some controllable events of $\mathbf{G}^k$ [5]. Thus the set of components that are control-coupled to $\mathbf{SUP}_p$ is $\{\mathbf{G}^k \mid (\exists x \in X_p)\ D^k(x) \neq \emptyset,\ k \in \mathcal{K}\}$; and we localize $\mathbf{SUP}_p$ to local controllers only for this set. Similarly, we localize $\mathbf{CO}_q = (X_q, \_, \_, \_, \_)$ to local controllers only for the set $\{\mathbf{G}^k \mid (\exists x \in X_q)\ D^k(x) \neq \emptyset,\ k \in \mathcal{K}\}$.

Now we present our main result; the proof is given in [3].

Theorem 1: **DAP** solves distributed control problem $(\divideontimes)$.

IV. Production Cell: System Description

This section and the next apply **DAP** to solve the distributed control problem for a benchmark application, Production Cell originating with [12], in a version we adapt from [13] [6]. In [13], optimal and nonblocking decentralized supervision has been established by the modular approach in [5]. This result will now be carried further to achieve our objective of distributed control.

Production Cell consists of nine asynchronous components: stock, feed belt, elevating rotary table, rotary base, arm1, arm2, press, deposit belt, and crane. The cell processes workpieces, called 'blanks', as displayed in Fig. 1.

A. Stock

Stock adds blanks into the cell by placing them on feed belt (ST_add); see **ST** in Fig. 2 [7].

B. Feed belt

According to **FB** in Fig. 2 the feed belt, once loaded, forwards blanks towards table (FB_F). Sensor1 at the end

[5]The control coupling relation can be determined by inspecting the control data table generated by the TCT procedure **condat** [11].

[6]We have slightly modified the modeling in [13] for easier interpretation of the cell's physical operations.

[7]In generator models, by convention we mark controllable events with a tick on the corresponding arrows.

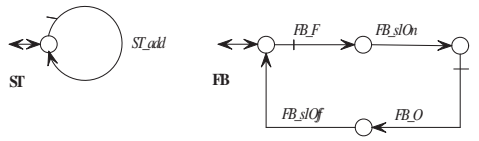

Fig. 2. Plant models of stock and feed belt

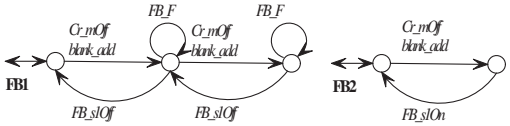

Fig. 3. Specification models of feed belt

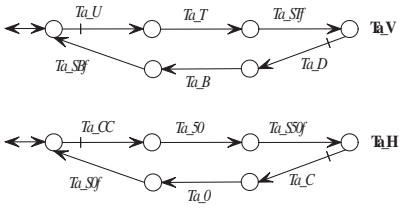

Fig. 4. Plant model of elevating rotary table

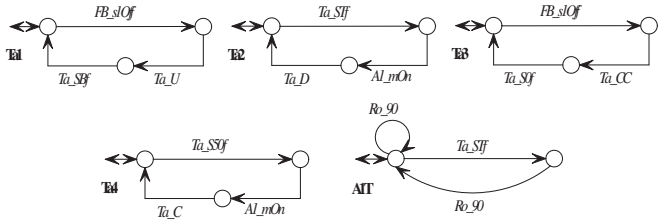

Fig. 5. Specification models of elevating rotary table

of the belt switches to 'on' when it detects the arrival of a blank (FB_s1On). Feed belt outputs a blank onto table if the latter is available (FB_O). Sensor1 switches to 'off' when a blank leaves (FB_s1Off).

Feed belt interacts with stock, crane, and table, subject to the following two specifications (Fig. 3):

- **FB1**: Feed belt forwards (FB_F) only when there are blanks loaded; and it can hold at most two blanks.
- **FB2**: If there is already one blank on feed belt, then for safety reasons a new blank is prohibited from being loaded before the first reaches the end of the belt and activates Sensor1 (FB_s1On).

C. Elevating rotary table

Feed belt is located lower than arm1. As displayed in Fig. 4, after being loaded by feed belt table moves up (Ta_U, Ta_T, Ta_STf) and turns counterclockwise (CCW) to $-50°$ (Ta_CC, Ta_50, Ta_S50f) for arm1 to pick up a blank. Thereafter table moves down (Ta_D, Ta_B, Ta_SBf) and turns clockwise (CW) back to $0°$ (Ta_C, Ta_0,Ta_S0f).

Table must synchronize with feed belt and arm1 when transferring blanks; the corresponding specifications are shown in Fig. 5:

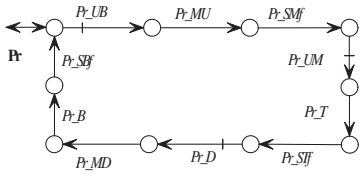

Fig. 6. Plant model of press

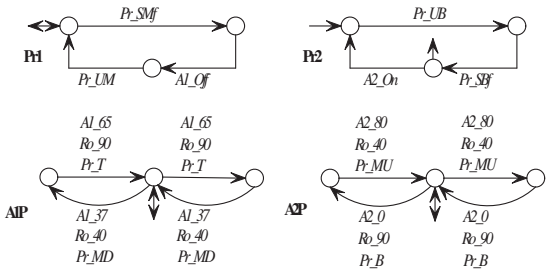

Fig. 7. Specification models of press

- **Ta1** and **Ta3**: Table can accept a blank from feed belt (FB_s1Off) only when it is at bottom (Ta_SBf) and at angle $0°$ (Ta_S0f); only after accepting a blank can table ascend (Ta_U) and turn CCW (Ta_CC).
- **Ta2** and **Ta4**: Table can transfer a blank to arm1 ($A1\_On$) only when it is at top (Ta_STf) and at angle $-50°$ (Ta_S50f); only after transferring a blank can table descend (Ta_D) and turn CW (Ta_C).
- Collision between two blanks could occur if and when arm1 has picked up one blank from table, rotary base has not yet turned CCW to $90°$ (Ro_90), and table returns to top with a new blank (Ta_STf). The specification that prevents this collision is enforced by **A1T**.

D. Press

Press operates at three different positions: bottom, middle, and top. According to Fig. 6 it is initially at bottom, and ascends to middle where arm1 may load a blank (Pr_UB, Pr_MU, Pr_SMf). After being loaded, press continues to top where it forges the blank (Pr_UM, Pr_T, Pr_STf). Then it descends back to bottom and prepares to unload the forged blank to arm2 (Pr_D, Pr_MD, Pr_B, Pr_SBf).

Press coordinates with arm1, arm2 as specified in Fig. 7:

- **Pr1**: Press can accept a blank from arm1 ($A1\_Off$) only at its middle position (Pr_SMf); only after accepting a blank can press move to top (Pr_UM).
- **Pr2**: Press can transfer a blank to arm2 ($A2\_On$) only at its bottom position (Pr_SBf); only after the transfer can press move to middle (Pr_UB).
- There are, additionally, two collision scenarios. First, press collides with arm1 if and when it is at top, arm1 is longer than 37, and rotary base is at $90°$. Second, press collides with arm2 if and when it is not at bottom, arm2 is longer than 0, and rotary base is at $40°$. The specifications for avoiding these collisions are enforced by **A1P** and **A2P**, respectively.

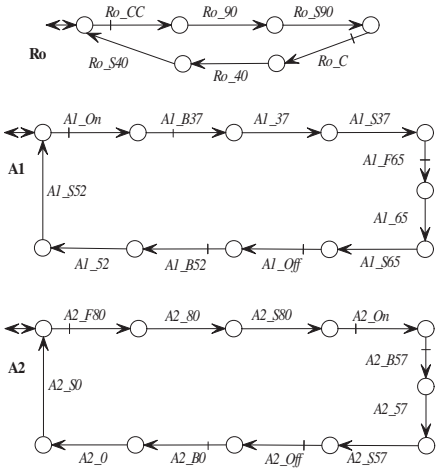

Fig. 8. Plant models of rotary base, arm1, and arm2

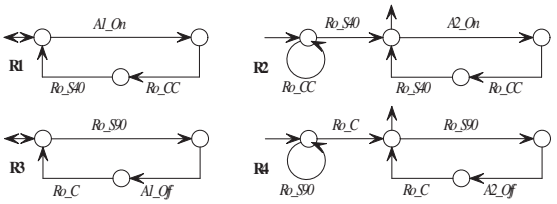

Fig. 9. Specification models of rotary base, arm1, and arm2

E. Rotary base, arm1, and arm2

Rotary base, arm1, and arm2 cooperatively transfer blanks from table through press to deposit belt. As displayed in Fig. 8, rotary base initially at 40° rotates CCW to 90° (Ro_CC, Ro_90, Ro_S90), and then CW back to 40° (Ro_C, Ro_40, Ro_S40). Arm1, once loaded ($A1\_On$), first retracts to length 37 ($A1\_B37$, $A1\_37$, $A1\_S37$) so as to avoid collision, and then extends to length 65 ($A1\_F65$, $A1\_65$, $A1\_S65$) at which point it can unload a blank onto press ($A1\_Off$); after unloading arm1 retracts to its initial length 52 ($A1\_B52$, $A1\_52$, $A1\_S52$). Lastly, arm2 first extends its length to 80 ($A2\_F80$, $A2\_80$, $A2\_S80$) at which point it can pick up a blank from press ($A2\_On$); it then retracts to 57 ($A2\_B57$, $A2\_57$, $A2\_S57$) and places a blank onto deposit belt ($A2\_Off$); thereafter it retracts to 0, its initial length ($A2\_B0$, $A2\_0$, $A2\_S0$).

The collaboration among base and two arms must satisfy the specifications in Fig. 9:

- **R1** and **R2**: Arm1 and arm2 may be loaded only when base is at 40° (Ro_S40); only after both are loaded may base turn CCW (Ro_CC).
- **R3** and **R4**: Arm1 and arm2 may unload only when base is at 90° (Ro_S90); only after both unloading actions are completed may base turn CW (Ro_C).

Notice that the marked states in **R2** and **R4** are so chosen because arm2 has no blank to be loaded or to load during the first work cycle of base. An analogous reason accounts for the choice of marked state of **Pr2** in Fig. 7: press has no blank to load arm2 for the first iteration of its actions.

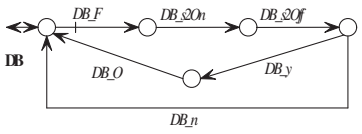

Fig. 10. Plant model of deposit belt

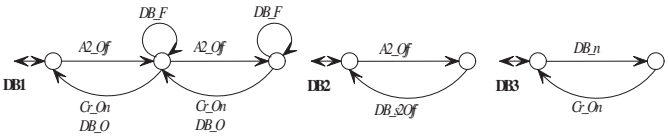

Fig. 11. Specification model of deposit belt

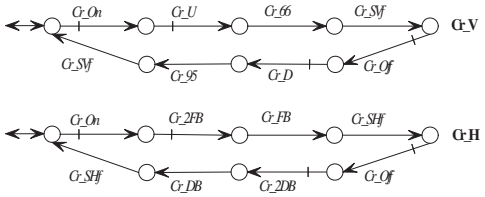

Fig. 12. Plant model of crane

F. Deposit belt

As shown in Fig. 10, once loaded deposit belt forwards blanks (DB_F) towards the other end; there Sensor2 switches to 'on' (DB_s2On) when it detects the arrival of a blank, and 'off' (DB_s2Off) to show that the blank has been checked by the test unit. If the blank passes the check (DB_y), then it will be output from the system (FB_O); otherwise (DB_n) it waits to be picked up by crane.

Deposit belt interacts with arm2 and crane, subject to the following three specifications (Fig. 11):

- **DB1**: Deposit belt forwards (DB_F) only when there are blanks loaded; and it can hold at most two blanks.
- **DB2**: If there is already one blank on deposit belt, then for safety reasons a new blank can be loaded only after the first is checked by the test unit (DB_s2Off).
- **DB3**: If a blank fails the test (DB_n), then it has to be taken by crane back for another cycle (Cr_On).

G. Crane

Deposit belt is located lower than feed belt. According to Fig. 12, after picking up a faulty blank from deposit belt (Cr_On) crane moves up (Cr_U, Cr_66, Cr_SVf) and horizontally towards feed belt (Cr_2FB, Cr_FB, Cr_SHf), to which it delivers the blank (Cr_Off). Thereafter crane moves down (Cr_D, Cr_95, Cr_SVf) and horizontally back towards deposit belt (Cr_2DB, Cr_DB, Cr_SHf).

V. Production Cell: Distributed Control

We are ready to apply **DAP** to the distributed control design for Production Cell.

1) Plant Model Abstraction: By the abstraction procedure in Section IV we effectively simplify the models of table, press, arm1, arm2, and crane, as displayed in Fig. 13.

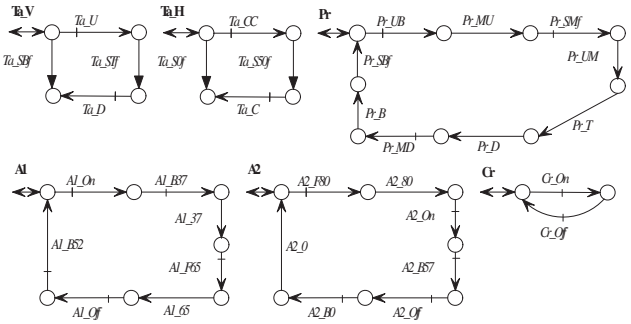

Fig. 13. Model abstractions of plant components

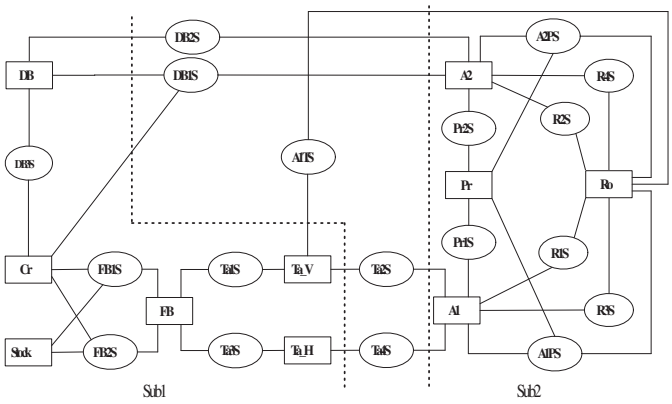

Fig. 14. Interconnection structure of production cell

2) Decentralized Supervisor Synthesis: For each specification we group together its event-coupled components, and synthesize for each group an optimal nonblocking decentralized supervisor using TCT procedure **supcon**. The result is presented in Fig. 14, with components in blocks, supervisors in ovals, and solid lines denoting event-coupling.

3) Subsystem Decomposition and Coordination: We have eighteen decentralized supervisors, thus eighteen modules. Following [13] we decompose the overall system into two subsystems, leaving five supervisors in between, as shown with dotted lines in Fig. 14. It is further checked that while **Sub1** is nonblocking, **Sub2** turns out to be blocking. We thus design a coordinator, **CO1**, to resolve the conflict, and denote the resulting nonblocking subsystem by **NSub2**.

4) Subsystem Model Abstraction: We must now verify the nonconflicting property among **Sub1**, **NSub2**, and the five intermediate supervisors [8]. First, we determine their shared event set, Σ_{sub}; by inspection we have

$$\Sigma_{sub} = \{DB\_F, DB\_s2Off, DB\_n, DB\_O, Cr\_On, Ta\_U, Ta\_STf, Ta\_D, Ta\_S50f, Ta\_C, Ro\_90, A1\_On, A2\_Off\}.$$

To ensure OCC we need to add events Ta_CC and Ro_CC to Σ_{sub}, for these are the nearest upstream controllable events from the uncontrollable ones Ta_S50f and Ro_90,

[8]In this and the next steps, we take for the supervisors their (control-equivalent) reduced generator models (i.e., the results of the **supreduce** procedure in TCT [11]).

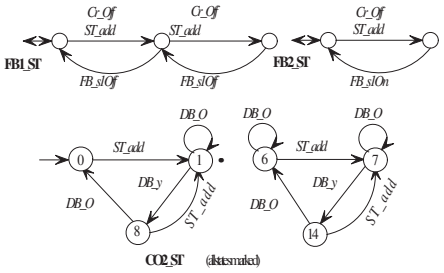

Fig. 15. Local controllers for stock

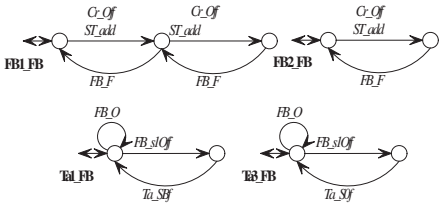

Fig. 16. Local controllers for feed belt

respectively. Further, for the observer property, we find by the MX algorithm that the additional events ST_add, DB_y, Cr_Off, and $A2\_F80$ are needed in Σ_{sub}. Denote by Σ'_{sub} the augmented alphabet; it defines an OCC observer P'_{sub} for both subsystems. So with P'_{sub} we compute the model abstractions, denoted by $\mathbf{Sub1}'$ and $\mathbf{Sub2}'$ respectively.

5) Abstracted Subsystem Decomposition and Coordination: We treat $\mathbf{Sub1}'$, $\mathbf{Sub2}'$, and the five intermediate supervisors as a single group, and directly check the nonblocking property. This group turns out to be blocking; a coordinator, **CO2**, is then designed to resolve the conflict.

6) Higher-Level Abstraction: The modular supervisory control design terminates with the previous Step 5. We have obtained a hierarchy of eighteen decentralized supervisors and two coordinators; their synchronized behavior is globally optimal and nonblocking [13].

7) Decentralized Supervisors and Coordinators Localization: We first determine the control-coupling relation between supervisors/coordinators and components; we do this by inspecting the corresponding **condat** tables in TCT [11]. Then we apply the **SL** algorithm to localize each supervisor/coordinator for its control-coupled component; the resulting local controllers are displayed in Figs. 15 – 23 (for clarity extraneous selfloops are omitted), grouped with respect to individual components. We observe that all controllers decomposed from the supervisors have only two to four states; thus their control logic can be easily understood by referring to the corresponding specification descriptions in Section IV. There are, additionally, two controllers, one from each of the two coordinators: **CO1_A2** in Fig. 21 forces arm2 to stay put at its initial state during the first work cycle of press; **CO2_ST** in Fig. 15 disables adding blanks into the cell if and when there are seven there already. Finally, by Theorem 1 these local controllers will collectively achieve global optimality and nonblockingness.

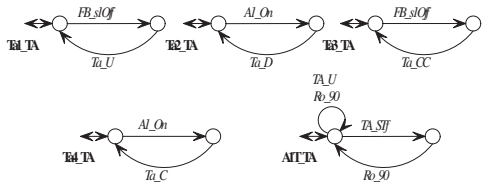

Fig. 17. Local controllers for elevating rotary table

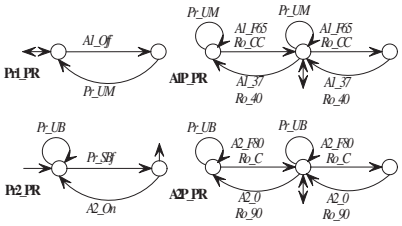

Fig. 18. Local controllers for press

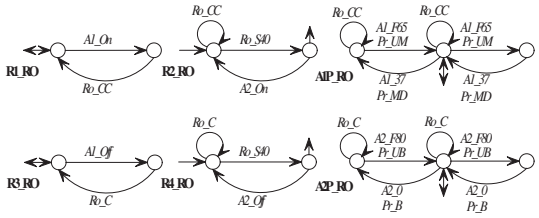

Fig. 19. Local controllers for robot

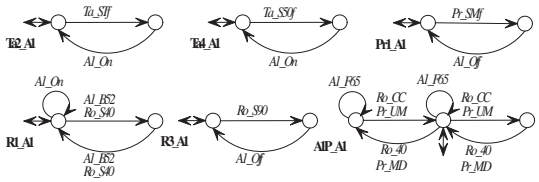

Fig. 20. Local controllers for arm1

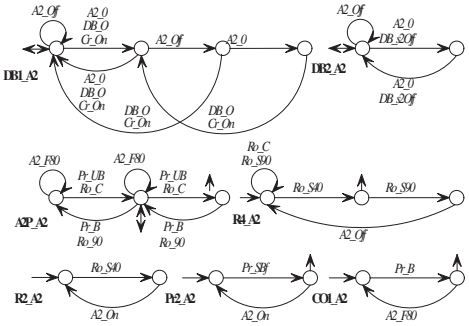

Fig. 21. Local controllers for arm2

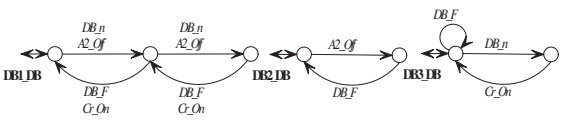

Fig. 22. Local controllers for deposit belt

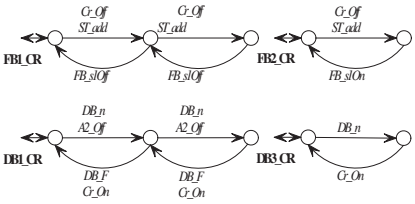

Fig. 23. Local controllers for crane

VI. Conclusion

We have studied distributed control design for large DES, and presented the decomposition-aggregation procedure as a corresponding solution. The procedure has been successfully demonstrated with the benchmark Production Cell.

One direction of future work aims to adapt supervisor localization to the *state tree structure*, a framework known to be highly efficient in monolithic supervisor synthesis (even for systems of state size 10^{24} or more) [14]. Equipped with that computational power, we may tackle distributed control design for much larger DES.

References

[1] 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 online at http://www.control.toronto.edu/DES/CaiKai_MASc_Thesis.pdf.

[2] K. Cai and W. M. Wonham, "Supervisor localization: a top-down approach to distributed control of discrete-event systems," in *Proc. 2nd Mediterranean Conference on Intelligent Systems and Automation (CISA09)*, Zarzis, Tunisia, Mar 2009, pp. 302–308.

[3] ——, "Supervisor localization: a top-down approach to distributed control of discrete-event systems," *IEEE Trans. Autom. Control*, to appear.

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

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

[6] W. M. Wonham, "Supervisory control of discrete-event systems," Systems Control Group, ECE Dept, University of Toronto, updated July 1, 2009. Available online at http://www.control.toronto.edu/DES.

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

[8] 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.

[9] L. Feng and W. M. Wonham, "On the computation of natural observers in discrete-event systems," *Discrete Event Dynamic Systems*, to appear. Available online at www.SpringerLink.com.

[10] ——, "Computationally efficient supervisory design: control flow decomposition," in *Proc. Int. Workshop Discrete Event Systems (WODES06)*, Ann Arbor, Michigan, U.S.A., Jul 2006, pp. 9–14.

[11] W. M. Wonham, "Design software: XPTCT," Systems Control Group, ECE Dept, University of Toronto, Version 133, Windows XP, updated July 1, 2009. Available online at http://www.control.toronto.edu/DES.

[12] C. Lewerentz and T. Lindner, Eds., *Formal Development of Reactive Systems – Case Study Production Cell*. London, UK: Springer-Verlag, 1995.

[13] L. Feng, K. Cai, and W. M. Wonham, "A structural approach to the nonblocking supervisory control of discrete-event systems," *International J. of Advanced Manufacturing Technology*, vol. 41, no. 11, pp. 1152–1167, 2009.

[14] C. Ma and W. M. Wonham, *Nonblocking Supervisory Control of State Tree Structures*, ser. LNCIS 317, M. Thoma and M. Morari, Eds. Berlin Heidelberg: Springer-Verlag, 2005.