2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

Maximally Permissive Distributed Supervisory Control of Nondeterministic Discrete-Event Systems

Rong Su, Jan H. van Schuppen and Jacobus E. Rooda

***Abstract*— In supervisor synthesis achieving nonblockingness is a major computational challenge for a large system. In the literature several automaton-based distributed synthesis approaches have been proposed, in which the plant is modeled by a collection of nondeterministic finite-state automata and the requirement and the final distributed supervisor are modeled by a collection of deterministic finite-state automata. In this paper we provide a sufficient condition, which guarantees maximal permissiveness of the synthesized distributed supervisor.**

***Index Terms*— discrete-event systems, nondeterministic finite-state automata, automaton abstraction, distributed supervisor synthesis, maximal permissiveness**

I. Introduction

In the Ramadge/Wonham supervisory control paradigm [1] [2] one of the main challenges of supervisor synthesis is to achieve nonblockingness when a target system has a large number of states, often resulting from synchronous product of many relatively small local components. To overcome this difficulty, many approaches have been proposed recently, e.g. state-feedback control based on state-tree structures [5], hierarchical interface-based control [4] and modular/distributed control [6] [8] [13] [14].

The modular/distributed approaches with decomposable requirements are particularly interesting for two reasons: potentially low synthesis complexity and high implementation flexibility. The low complexity is achieved through local synthesis, and implementation flexibility refers that a structural change of the target system may require only a small number of relevant local controllers to be updated. There are two major types of modular/distributed synthesis approaches: language-based approaches, e.g., [6] [7], and automaton-based approaches, e.g., [8] [10] [14]. The language-based approaches utilize the concept of natural observers [3] to guarantee that a nonblocking supervisor synthesized based on an abstracted plant model is also a nonblocking supervisor of the original plant model. To achieve the same goal, the automaton-based approaches usually use an appropriate variation of the concept of weak bisimulation equivalence relation in model abstraction. It has been shown that imposing the natural observer property on relevant projections may unnecessarily enlarge the size of the abstracted plant model, which may significantly increase the subsequent synthesis complexity. Therefore, using automaton-based abstraction techniques may be arguably more attractive for practical applications. There is one catch for using automaton-based approaches - the abstracted models usually are nondeterministic, which fortunately turns out to be just a minor side effect.

In general, a nonblocking modular/distributed supervisor is not maximally permissive. Nevertheless, under some circumstances the maximal permissiveness is attainable in modular/distributed control. For language-based approaches some sufficient conditions have been given in the literature, e.g., in [6] [7]. But it is still an open problem for automaton-based approaches, which this paper aims to solve. In addition, we will also address the issue of partial observation in maximally permissive distributed control, which has not been discussed in the literature. More explicitly, we extend the concept of *local control consistency* in [13] (which is more general than the concept of *output control consistency* used in [6]) and the concept of *mutual controllability* in [12] to handle nondeterministic plant models. We also introduce a new concept called *compatibility*. It generalizes the basic idea of natural observers so that its usage is less restrictive than natural observers, thus, usually resulting smaller abstracted models which still allow the existence of maximally permissive distributed supervisors. In addition, nondeterminism and partial observation are both taken into account in this new concept, making it even more general than natural observers.

This paper is organized as follows. In Section II we review relevant concepts, automaton operations and the general setup of distributed supervisory control described in [10]. Then in Section III we formulate a distributed supervisor synthesis problem and introduce the concept of a distributed supervisor. A sufficient condition to guarantee maximal permissiveness of a distributed supervisor is presented in Section IV. After providing a realistic example in Section V we draw conclusions in Section VI.

II. Basic concepts

In this section we first introduce basic concepts of languages and nondeterministic finite-state automata. Then we describe automaton abstraction introduced in [11].

Rong Su is affiliated with Division of Control and Instrumentation, School of Electrical and Electronic Engineering, Nanyang technological University, 50 Nanyang Avenue, Singapore 639798. Email: rsu@ntu.edu.sg
Jacobus E. Rooda is affiliated with the Systems Engineering Group in the Department of Mechanical Engineering, Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands. Email: j.e.rooda@tue.nl
Jan H. van Schuppen is affiliated with Centrum voor Wiskunde en Informatica (CWI), P.O. Box 94079, 1090 GB Amsterdam, The Netherlands. E-mail: j.h.van.schuppen@cwi.nl

A. Languages and nondeterministic finite-state automata

Let Σ be a finite alphabet and Σ^* be the free monoid on Σ, where the unit element is the empty string ϵ and the monoid operation is the concatenation. Given two strings $s, t \in \Sigma^*$, s is called a *prefix substring* of t, written as $s \leq t$, if there exists $s' \in \Sigma^*$ such that $ss' = t$, where ss' denotes the concatenation of s and s'. A subset $L \subseteq \Sigma^*$ is called a *language*. $\overline{L} = \{s \in \Sigma^* | (\exists t \in L)\, s \leq t\} \subseteq \Sigma^*$ is called the *prefix closure* of L. Given a set X we use 2^X and $|X|$ to denote respectively the power set and the size of X.

Let $\Sigma' \subseteq \Sigma$. A mapping $P : \Sigma^* \rightarrow \Sigma'^*$ is called the *natural projection* with respect to (Σ, Σ'), if

1) $P(\epsilon) = \epsilon$
2) $(\forall \sigma \in \Sigma)\, P(\sigma) := \begin{cases} \sigma & \text{if } \sigma \in \Sigma' \\ \epsilon & \text{otherwise} \end{cases}$
3) $(\forall s\sigma \in \Sigma^*)\, P(s\sigma) = P(s)P(\sigma)$

Given a language $L \subseteq \Sigma^*$, $P(L) := \{P(s) \in \Sigma'^* | s \in L\}$. The inverse image mapping of P is

$$P^{-1} : 2^{\Sigma'^*} \rightarrow 2^{\Sigma^*} : L \mapsto P^{-1}(L) := \{s \in \Sigma^* | P(s) \in L\}.$$

Given $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$, the *synchronous product* of L_1 and L_2 is defined as:

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

where $P_1 : (\Sigma_1 \cup \Sigma_2)^* \rightarrow \Sigma_1^*$ and $P_2 : (\Sigma_1 \cup \Sigma_2)^* \rightarrow \Sigma_2^*$ are natural projections. $||$ is commutative and associative.

A *nondeterministic finite-state automaton* is a 5-tuple $G = (X, \Sigma, \xi, x_0, X_m)$, where X stands for the finite state set, Σ for the alphabet, $\xi : X \times \Sigma \rightarrow 2^X$ for the nondeterministic transition function, x_0 for the initial state and X_m for the marker state set. As usual [9], we extend the domain of ξ from $X \times \Sigma$ to $X \times \Sigma^*$. If for all $x \in X$ and $\sigma \in \Sigma$, $|\xi(x, \sigma)| \leq 1$, then G is called *deterministic*. Let $B(G) := \{s \in \Sigma^* | (\exists x \in \xi(x_0, s))(\forall s' \in \Sigma^*)\, \xi(x, s') \cap X_m = \varnothing\}$. We say G is *nonblocking* if $B(G) = \varnothing$. For each $x \in X$, we define another set $N(G) := \{s \in \Sigma^* | \xi(x_0, s) \cap X_m \neq \varnothing\}$ and call $N(G)$ the *nonblocking set* of G, which is simply the set of all strings recognized by G. It is possible that $B(G) \cap \overline{N(G)} \neq \varnothing$, due to nondeterminism. Let $\phi(\Sigma)$ be the collection of all nondeterministic finite-state automata over Σ. Given a language $K \subseteq \Sigma^*$, we say $G \in \phi(\Sigma)$ is a *recognizer* of K, if G is deterministic, nonblocking and $N(G) = K$. The synchronous product of two nondeterministic automata, say $G_1 \in \phi(\Sigma_1)$ and $G_2 \in \phi(\Sigma_2)$ is defined in the usual way [15], which is denoted as $G_1 \times G_2$.

B. Automaton abstraction

Definition 1: [11] Given $G = (X, \Sigma, \xi, x_0, X_m) \in \phi(\Sigma)$, let $\Sigma' \subseteq \Sigma$ and $P : \Sigma^* \rightarrow \Sigma'^*$ be the natural projection. A *weak observation equivalence* relation on X with respect to Σ' is an equivalence relation $R \subseteq \{(x, x') \in X \times X | x \in X_m \iff x' \in X_m\}$ such that for all $(x, x') \in R$, and all $s \in \Sigma^*$, if $P(s) \neq \epsilon$ then for all $y \in \xi(x, s)$,

$$(\exists s' \in \Sigma^*)\, P(s) = P(s') \wedge (\exists y' \in \xi(x', s'))\, (y, y') \in R.$$

The largest weak observation equivalence relation on X with respect to Σ' is denoted as $\approx_{G,\Sigma'}$. $\square$

If G is clear from the context we simply write $\approx_{\Sigma'}$ for $\approx_{G,\Sigma'}$. In [11] it has been shown that the largest weak observation equivalence relation is weaker than the weak bisimilarity used in many existing abstraction techniques, e.g., in [8] [14], thus can achieve a smaller abstracted model than those techniques, which are based on the weak bisimilarity, can achieve.

Definition 2: [11] Given $G = (X, \Sigma, \xi, x_0, X_m)$, let $\Sigma' \subseteq \Sigma$. The *automaton abstraction* of G with respect to $\approx_{\Sigma'}$ is an automaton $G/\approx_{\Sigma'} := (Z, \Sigma', \delta, z_0, Z_m)$ where

1) $Z := X/\approx_{\Sigma'} := \{\{x' \in X | (x, x') \in \approx_{\Sigma'}\} | x \in X\}$,
2) $z_0 := < x_0 >$,
3) $Z_m := \{z \in Z | z \cap X_m \neq \varnothing\}$,
4) $\delta : Z \times \Sigma' \rightarrow 2^Z$, where for any $(z, \sigma) \in Z \times \Sigma'$,

$$\delta(z, \sigma) := \{z' \in Z | (\exists x \in z)(\exists u, u' \in (\Sigma - \Sigma')^*)\, \xi(x, u\sigma u') \cap z' \neq \varnothing\}.$$
$\square$

The time complexity of computing $G/\approx_{\Sigma'}$ is dominated by the time complexity of computing $X/\approx_{\Sigma'}$, which is $O(|\Sigma||X|^2 \log |X|)$ [11]. To use the proposed automaton abstraction technique properly, we need to introduce the concept of *standardized automata*, which is defined as follows. We bring in two new symbols τ and μ not contained in Σ. We call τ the *initial event* and μ the *marking event*. Let $\Sigma^{\tau,\mu} := \Sigma \cup \{\tau, \mu\}$. We say $G^{\tau,\mu} = (X, \Sigma^{\tau,\mu}, \xi, x_0, X_m)$ is *standardized* if the following hold

1) $x_0 \notin X_m \wedge (\forall x \in X)\, [\xi(x, \tau) \neq \varnothing \iff x = x_0] \wedge (\forall \sigma \in \Sigma - \{\tau\})\, \xi(x_0, \sigma) = \varnothing$
2) $(\forall x \in X)(\forall \sigma \in \Sigma)\, x_0 \notin \xi(x, \sigma)$
3) $(\forall x \in X)\, x \in X_m \iff x \in \xi(x, \mu)$

A standardized automaton is an automaton, in which x_0 is not marked (by condition 1), τ is only used at x_0, which only has outgoing transitions labeled by τ (by condition 1) and no incoming transition (by condition 2), and each marker state has a selflooping transition μ (by condition 3). From now on we only consider standardized automata. For notational simplicity we assume that each alphabet Σ contains τ and μ, and $\phi(\Sigma)$ is the set of all standardized finite-state automata, whose alphabets are Σ. In [11] it has been shown that the product of two standardized automata is a standardized automaton, and the abstraction of a standardized automaton, whose target alphabet contains τ and μ, is still standardized.

In control engineering examples G usually consists of a large number of small automata, namely $G = G_1 \times \cdots \times G_n$ for some very large number $n \in \mathbb{N}$, where $G_i \in \phi(\Sigma_i)$ for each $i = 1, 2, \cdots, n$. How to compute $G/\approx_{\Sigma'}$ imposes a great computational difficulty. To overcome it, a sequential abstraction algorithm called SAP is presented in [11], which generates an automaton $W \in \phi(\Sigma')$ such that $(\times_{i \in I} G_i)/\approx_{\Sigma'} \cong W$, where $\approx_{\Sigma'} \cong$ is called the *nonblocking* *equivalence relation*, which essentially says that $(\times_{i\in I}G_i)$ and W are equivalent in terms of their blocking and nonblocking behaviors.

III. A DISTRIBUTED SYNTHESIS PROBLEM

Given $G = (X, \Sigma, \xi, x_0, X_m) \in \phi(\Sigma)$, for each $x \in X$ let

$$E_G : X \to 2^{\Sigma} : x \mapsto E_G(x) := \{\sigma \in \Sigma | \xi(x, \sigma) \neq \varnothing\}.$$

Thus, $E_G(x)$ is simply the set of all events allowable at x in G. Let $\Sigma = \Sigma_c \cup \Sigma_{uc} = \Sigma_o \cup \Sigma_{uo}$, where the disjoint subsets Σ_c and Σ_{uc} denote respectively the set of controllable events and the set of uncontrollable events, and the disjoint subsets Σ_o and Σ_{uo} denote respectively the set of observable events and the set of unobservable events. In particular, $\tau \in \Sigma_{uc} \cap \Sigma_{uo}$ and $\mu \in \Sigma_c \cap \Sigma_{uo}$. Let $L(G) := \{s \in \Sigma^* | \xi(x_0, s) \neq \varnothing\} = B(G) \cup \overline{N(G)}$.

Definition 3: [10] Given $G = (X, \Sigma, \xi, x_0, X_m)$ and $\Sigma' \subseteq \Sigma$, let $A = (Y, \Sigma', \eta, y_0, Y_m) \in \phi(\Sigma')$. A is *state-controllable* with respect to G and Σ_{uc} if for all $s \in L(G \times A)$,
$(\forall (x, y) \in \xi \times \eta(x_0, y_0, s))\, E_G(x) \cap \Sigma_{uc} \cap \Sigma' \subseteq E_A(y)$. □

Intuitively speaking, A will not block occurrence of any uncontrollable event in G. When G and A are deterministic, state-controllability coincides with language controllability introduced in [1]. Next, we introduce the concept of *state normality*. Let $P_o : \Sigma^* \to \Sigma_o^*$ be the natural projection.

Definition 4: [10] Given $G = (X, \Sigma, \xi, x_0, X_m) \in \phi(\Sigma)$ and $\Sigma' \subseteq \Sigma$, let $A = (Y, \Sigma', \eta, y_0, Y_m) \in \phi(\Sigma')$. A is *state-normal* with respect to G and P_o if for all $s \in L(G \times A)$ and $s' \in P_o^{-1}(P_o(s)) \cap L(G \times A)$,
$(\forall (x, y) \in \xi \times \eta(x_0, y_0, s'))(\forall s'' \in \Sigma^*)\, P_o(s's'') = P_o(s) \wedge \xi(x, s'') \neq \varnothing \Rightarrow \xi \times \eta(x, y, s'') \neq \varnothing$. □

State-normality is a direct extension of language normality [18] for nondeterministic cases. They coincide when G and A are deterministic. We now present the concept of distributed systems.

Definition 5: [10] A *distributed system* with respect to given alphabets $\{\Sigma_i | i \in I\}$ is a finite collection of nondeterministic finite-state automata $\mathcal{G} := \{G_i = (X_i, \Sigma_i, \xi_i, x_{i,0}, X_{i,m}) \in \phi(\Sigma_i) | i \in I\}$. Each G_i ($i \in I$) is called the i^{th} *component* of $\mathcal{G}$, and $\Sigma_i = \Sigma_{i,c} \cup \Sigma_{i,uc} = \Sigma_{i,o} \cup \Sigma_{i,uo}$, where disjoint subsets $\Sigma_{i,c}$ and $\Sigma_{i,uc}$ are the *controllable* and *uncontrollable* alphabets respectively, and disjoint subsets $\Sigma_{i,o}$ and $\Sigma_{i,uo}$ are the *observable* and *unobservable* alphabets respectively. For all $i, j \in I$ with $i \neq j$, we assume that $\Sigma_{i,uc} \cap \Sigma_{j,c} = \Sigma_{i,uo} \cap \Sigma_{j,o} = \varnothing$. The *compositional behavior* of $\mathcal{G}$ is specified by $\times_{i\in I}G_i$. □

The product of local components is the system of interest. Interaction among local components is modeled by event sharing among local components. We now present a supervisor synthesis problem.

Distributed Supervisor Synthesis Problem: [10] Given a distributed system $\mathcal{G} = \{G_i \in \phi(\Sigma_i) | i \in I\}$ and a set of specifications $\mathcal{H} = \{H_j \in \phi(\Delta_j) | \Delta_j \subseteq \cup_{i\in I}\Sigma_i \wedge j \in J\}$, where J is a finite index set and each H_j is a deterministic automaton, synthesize a collection of deterministic finite-state automata

$$\mathcal{S} = \{S_k \in \phi(\Gamma_k) | \Gamma_k \subseteq \cup_{i\in I}\Sigma_i \wedge k \in K\},$$

where K is a finite index set, such that the following conditions hold,

1) $N((\times_{i\in I}G_i) \times (\times_{k\in K}S_k)) \subseteq N((\times_{i\in I}G_i) \times (\times_{j\in J}H_j))$,
2) $B((\times_{i\in I}G_i) \times (\times_{k\in K}S_k)) = \varnothing$,
3) $\times_{k\in K}S_k$ is state-controllable with respect to $\times_{i\in I}G_i$ and $\cup_{i\in I}\Sigma_{i,uc}$,
4) $\times_{k\in K}S_k$ is state-normal with respect to $\times_{i\in I}G_i$ and $P_o : (\cup_{i\in I}\Sigma_i)^* \to (\cup_{i\in I}\Sigma_{i,o})^*$. □

We can check that the decentralized supervisor synthesis mentioned in [16] is a special case of the aforementioned distributed synthesis problem. Therefore, the existence of $\mathcal{S}$ with respect to $\mathcal{G}$, $\mathcal{H}$ and given alphabets $\{\Gamma_k | k \in K\}$ is in general undecidable. Nevertheless, in many practical applications such an $\mathcal{S}$ can be computed, see in [10]. When this happens, $\mathcal{S}$ is called a *nonblocking distributed supervisor* of $\mathcal{G}$ under $\mathcal{H}$, and each S_k is a *local supervisor*.

IV. MAXIMALLY PERMISSIVE DISTRIBUTED CONTROL

In general, given a distributed system $\mathcal{G}$ and a set of requirements $\mathcal{H}$, a distributed supervisor will not achieve the same permissiveness as that of a monolithic supervisor, which is obtained by first computing the product of all components and the product of all requirements, then performing centralized supervisor synthesis. In this section we will discuss under what conditions an aforementioned distributed supervisor attains maximal permissiveness.

Definition 6: Given $G = (X, \Sigma, \xi, x_0, X_m) \in \phi(\Sigma)$ and deterministic $H = (W, \Delta, \psi, w_0, W_m) \in \phi(\Delta)$ with $\Delta \subseteq \Sigma$, let $P'_o : \Sigma^* \to (\Delta \cap \Sigma_o)^*$ be natural projections. G is Δ-*compatible with respect to* H if for all $s, s' \in L(G \times H)$ and $(x, w) \in \xi \times \psi(x_0, w_0, s)$,

1) $P'_o(s) = P'_o(s') \Rightarrow (\forall (x', w') \in \xi \times \psi(x_0, w_0, s'))\, (x, w) \approx_{G\times H,\Delta} (x', w')$. □

The concept of Δ-compatibility is a generalization of natural observers in a nondeterministic setup under partial observation. The condition in the definition is an analogue to the condition for the projection $P : \Sigma^* \to \Delta^*$ to be an $L(G \times H)$-observer and an $L_m(G \times H)$-observer simultaneously, except that we use the weak observation equivalence relation to deal with nondeterminism and we consider partial observation. It is interesting to notice that if G is deterministic, fully observed and nonblocking, then

P is an $L_m(G)$-observer implies that Condition (1) holds, regardless of what the requirement H is. This suggests that the common practice of using $L_m(G)$-observers in existing maximal permissive synthesis may be over restrictive. The condition of Δ-compatibility can be checked by the following algorithm.

Procedure SC:
(1) Input: $G = (X, \Sigma, \xi, x_0, X_m) \in \phi(\Sigma)$ and deterministic $H = (W, \Delta, \psi, w_0, W_m) \in \phi(\Delta)$.
(2) Compute $G \times H = (Z := X \times W, \Sigma, \delta := \xi \times \psi, z_0 := (x_0, w_0), Z_m := X_m \times W_m)$.
(3) Compute $Z/\approx_{G\times H,\Delta}$.
(4) Output **True** if for all $<z>, <\hat{z}> \in Z/\approx_{G\times H,\Delta}$ with $<z> \neq <\hat{z}>$ and $z' \in <z>, \hat{z}' \in <\hat{z}>$,

$$(\forall \sigma \in \Sigma)\, \hat{z}' \in \delta(z', \sigma) \Rightarrow \delta(z', \sigma) \subseteq <\hat{z}> \wedge \sigma \in \Delta \cap \Sigma_o;$$

Otherwise, output **false**. □

Proposition 1: Condition (1) in Def. 6 holds iff Procedure SC outputs **True**. □

Since the complexity of determining all equivalent states with respect to the weak observation equivalence relation $\approx_{G\times H,\Delta}$, i.e., $Z/\approx_{G\times H,\Delta}$, is $\mathcal{O}(|X|^2|W|^2|\Sigma|\log(|X||W|))$ [17], the complexity of checking the condition in Def. 6 is $\mathcal{O}(|X|^2|W|^2|\Sigma|\log(|X||W|))$.

Definition 7: Let $G = (X, \Sigma, \xi, x_0, X_m)$ and $\Sigma' \subseteq \Sigma$. We say G is *control consistent* with respect to Σ' if for all $x, x' \in X$ and all $s \in P^{-1}((\Sigma_{uc} \cap \Sigma')^+)$,

$$x' \in \xi(x, s) \Rightarrow (\exists s' \in \Sigma_{uc}^+)\, P(s) = P(s') \wedge x' \in \xi(x, s'),$$

where $P : \Sigma^* \to (\Sigma_{uc} \cap \Sigma')^*$ is the natural projection. □

Informally speaking, if G is control consistent with respect to Σ', then no uncontrollable event in Σ' can be stopped by disabling a controllable event in $\Sigma - \Sigma'$. The concept of control consistency is a direct extension of the concept of *local control consistency* presented in [13] to the nondeterministic cases. Thus, the complexity of checking control consistency is the same as that for local control consistency, which is $O(|X|^4|\xi|^3)$, where $|\xi|$ denotes the number of transitions.

Theorem 1: Given a plant $G \in \phi(\Sigma)$ and a requirement $H \in \phi(\Delta)$ with $\Delta \subseteq \Sigma$. Assume that we have $W \in \phi(\Delta)$ with $W \cong G/\approx_\Delta$. Let $S \in \phi(\Delta)$ be the supremal NSN supervisor of W under H. If G is control consistent with respect to Δ and Δ-compatible with respect to H, then a recognizer of $N(G)||N(S)$ is the supremal NSN supervisor of G under H. □

Theorem 1 is about maximal permissiveness of a supervisor computed based on an abstracted model. The reason why we introduce the condition $W \cong G/\approx_\Delta$ is that in cases when G consists of many small automata, computing $G/\approx_\Delta$ is done via some sequential procedure such as SAP, which produces W instead of $G/\approx_\Delta$. Theorem 1 assures that we can synthesize the supremal state-normal supervisor based on an abstracted model W obtained from SAP. Compared with the results in [6] and [13], Theorem 1 drops the requirement of observers used in [6] [13] and replace it with the concept of Δ-compatibility with respect to H, which is in general weaker than the concept of observers, and extends the concept of local control consistency introduced in [13] (or the concept of output control consistency used in [6]). Partial observation is dealt with in the concept of Δ-compatibility, which is not considered in [6] and [13].

So far we have provided a sufficient condition to guarantee maximal permissiveness of a supervisor computed based on an abstracted model. It is interesting to know under what conditions a nonblocking distributed supervisor obtained by coordinated synthesis achieves maximal permissiveness. To this end we first extend the concept of mutual controllability so that it is applicable to nondeterministic models.

Definition 8: Given a distributed system $\mathcal{G} = \{G_i = (X_i, \Sigma_i, \xi_i, x_{i,0}, X_{i,m}) \in \phi(\Sigma_i) | i \in I\}$, for each $i, j \in I$ let $P_{ij} : \Sigma_i^* \to (\Sigma_i \cap \Sigma_j)^*$ be the natural projection. We say $\mathcal{G}$ is *mutually controllable* if for each $i, j \in I$ with $i \neq j$, for all $s_i \in \Sigma_i^*$ and $s_j \in \Sigma_j^*$ with $P_{ij}(s_i) = P_{ji}(s_j)$, and for all $x_i \in \xi_i(x_{i,0}, s_i)$, $x_j \in \xi_j(x_{j,0}, s_j)$ and $\sigma \in \Sigma_{i,uc} \cap \Sigma_{j,uc}$, $\xi_i(x_i, \sigma) \neq \varnothing$ if and only if $\xi_j(x_j, \sigma) \neq \varnothing$. □

A distributed system $\mathcal{G}$ is mutually controllable if for every two different subsystems G_i and G_j running together, G_i allows an uncontrollable event shared by both G_i and G_j to be fired if and only if G_j also allows the same uncontrollable event to be fired. In other words, there is no uncontrollable event, whose occurrence can be blocked simply by the parallel composition of subsystems. Thus, to prevent the occurrence of an uncontrollable event, an appropriate controllable event disabling must be taken.

Theorem 2: Given a distributed system $\mathcal{G} = \{G_i \in \phi(\Sigma_i) | i \in I\}$ and a collection of requirements $\mathcal{H}$ consisting of local requirements $\{H_i \in \phi(\Delta_i) | \Delta_i \subseteq \Sigma_i \wedge i \in I\}$ and a global requirement $H \in \phi(\Delta)$ with $\cup_{i,j:i\neq j}\Sigma_i \cap \Sigma_j \subseteq \Delta \subseteq \cup_{i\in I}\Sigma_i$, suppose for each G_i we have the supremal NSN supervisor $S_i \in \phi(\Sigma_i)$ under H_i. Let $P_i : \Delta^* \to (\Delta \cap \Sigma_i)^*$ be the natural projection. Suppose $S \in \phi(\Delta)$ is the supremal NSN supervisor of $\times_{i\in I}((G_i \times S_i)/\approx_{\Delta\cap\Sigma_i})$ under H. If for each $i \in I$, $G_i \times S_i$ is control consistent with respect to $\Delta \cap \Sigma_i$ and $\Delta \cap \Sigma_i$-compatible with respect to $\hat{H}_i$ which recognizes $P_i(L_m(H))$, $\{G_i \times S_i | i \in I\}$ is mutually controllable and for each $i \in I$ we have $\cup_{j\in I, j\neq i}(\Sigma_i \cap \Sigma_j) \subseteq \Sigma_{i,o}$, then $S \times_{i\in I} S_i$ is the supremal NSN supervisor of $\times_{i\in I} G_i$ under $H \times_{i\in I} H_i$. □

In addition to the componentwise conditions of control consistency and compatibility, to guarantee maximal permissiveness of a distributed supervisor, Theorem 2 also requires that $\{G_i \times S_i | i \in I\}$ is mutually controllable and in addition, for each $i \in I$, $\Sigma_{i,o} \supseteq \cup_{j\in I, j\neq i}(\Sigma_i \cap \Sigma_j)$, which means all shared events are observable. The last condition is used to deal with partial observation. In the case of full observation the condition $\cup_{j\in I, j\neq i}(\Sigma_i \cap \Sigma_j) \subseteq \Sigma_{i,o}$ is automatically satisfied. It is still an open question whether the last condition can be relaxed further. To illustrate the proposed sufficient condition, we apply it to the following cluster tool example.

V. Example - a cluster tool

We consider the following cluster tool depicted in Figure 1, which consists of one entering load lock (L_{in}) and one exit load lock (L_{out}), four chambers (C_{11}, C_{12}, C_{21}, C_{22}), two one-slot buffers (B_1, B_2), and three transportation robots (R_1, R_2, R_3). Wafers are transported into the system from the entering load lock by the robot R_1, then moved through designated chambers for processing based on pre-specified routing sequences by relevant robots located in different clusters. Finally, processed wafers are transported out of the system through exit load lock by R_1. As an

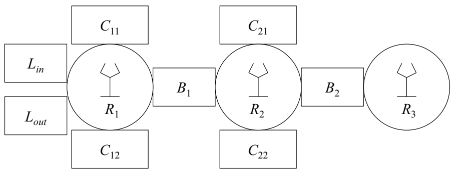

Fig. 1. Structure of cluster tool

illustration, we choose the following routing sequence: $L_{in} \to C_{11} \to B_1 \to C_{21} \to B_2 \to C_{22} \to B_1 \to C_{12} \to L_{out}$. Different routine sequences result in different requirements, which lead to different distributed supervisors and the corresponding synthesis complexities. Without supervision the system may be blocked owing to wafers competing for buffer slots. Our goal is to synthesize a maximally permissive distributed supervisor that guarantees continuous wafer processing, namely blocking should not happen. To this end, we first model the system as follows.

For simplicity we assume that the entering load lock L_{in} behaves like an infinite wafer source and the exit load lock L_{out} like an infinite wafer sink. Figure 2 depicts the models of load locks. We assume that in each chamber a wafer is

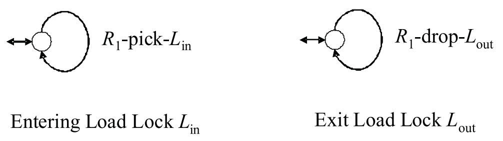

Fig. 2. Load locks

first dropped in by a relevant robot, then processed and finally picked up by the relevant robot. Since each chamber has the same automaton model, except for different alphabets, we only provide the model for one chamber, which is depicted in Figure 3, where $i = 1, 2$ and $j = 1, 2$. Notice that each

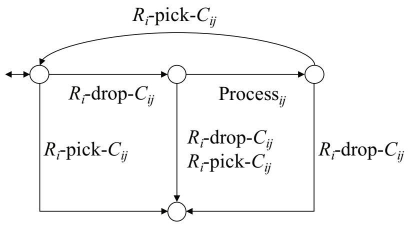

Fig. 3. Model of chamber C_{ij}

chamber behaves like a one-slot buffer, except that it contains an internal transition Process$_{ij}$. If robot R_i tries to pick when the chamber is empty, or drop when the chamber is full, the component will become deadlock. By modeling in such a way we will force a nonblocking supervisor to prevent inappropriate pick or drop actions to happen. The models of robots are depicted in Figure 4. Finally we model each buffer B_i ($i = 1, 2$) as a component, whose model is provided in Figure 5, indicating that buffer overflow or underflow will result in deadlock. In these models we assume that all events are observable and only events of the robots are controllable. The local requirements are depicted in Figure 6.

To synthesize a distributed supervisor, we first standardize all component models then partition the system into two modules. Module 1 consists of L_{in}, L_{out}, C_{11}, C_{12}, B_1 and R_1. Module 2 consists of C_{21}, C_{22}, B_1, B_2 and R_2 and R_3. With Module i ($i = 1, 2$) we associate H_{i1}, H_{i2}, H_{i3} and H_{i4} as local specifications. For each module we synthesize a local supremal nonblocking state-normal supervisor. For Module 1 we first compute the standardized plant model

$$G_1 := L_{in} \times L_{out} \times C_{11} \times C_{12} \times B_1 \times R_1$$

then we compute the local requirement

$$H_1 := H_{11} \times H_{12} \times H_{13} \times H_{14}$$

Based on G_1 and H_1 we synthesize the local supervisor S_1. Similarly, we synthesize the local supervisor S_2 for Module 2. The computational results are listed as follows:

$$G_1\ (129, 578)\ ;\ H_1\ (17, 66)\ ;\ S_1\ (58, 120)$$
$$G_2\ (509, 2534)\ ;\ H_2\ (17, 66)\ ;\ S_2\ (138, 328)$$

where in each tuple (x, y), x denotes the number of states and y for the number of transitions.

It is not difficult to see that G_1 and G_2 are mutually controllable because none of uncontrollable events are shared. It is also true that $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma_{i,o}$ for $i = 1, 2$ because all events are observable. We set $\Delta := \{R_1$-pick-C_{11}, R_2-pick-C_{22} and R_3-drop-$B_2\}$, which makes $G_i \times S_i$ ($i = 1, 2$) control consistent with respect to $\Delta \cap \Sigma_i$ and strongly Δ-compatible with respect to $\hat{H}_i$ - a recognizer of $(\Delta \cap \Sigma_i)^*$. Thus, we can perform abstraction-based synthesis. After computing $(G_1 \times S_1 \times G_2 \times S_2)/\approx_\Delta$, whose size is (13, 84), we use it as an abstracted model

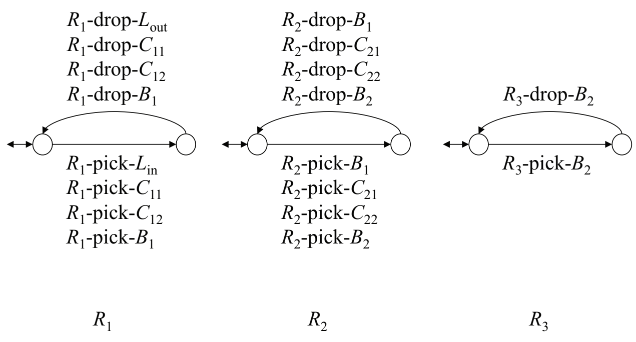

Fig. 4. Models of robots

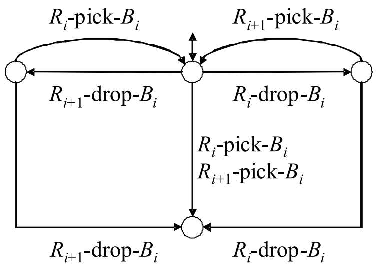

Fig. 5. Model of buffer B_i

and synthesize a new local supervisor CA, which is called a coordinator whose size is (4, 9). A language-equivalence test shows that $S_1 \times S_2 \times CA$ is maximal permissive.

As a comparison we also apply the language-based approach [6] to derive a distributed supervisor. The results for two modules are listed below.

G_1 (128, 544);S_1 (57, 118); G_2 (508, 2404);S_2 (137, 326)

At the module level the local supervisors are almost the same except that in automaton-based results each local supervisor contains one extra state and two extra transitions (one for τ and one for μ). When we perform model abstraction with respect to the target alphabet Δ, it turns out we need to add R_1-pick-L_{in} to Δ to guarantee that both natural projections are observers. The size of the final coordinator CO is (7, 16), which is bigger than the size of CA.

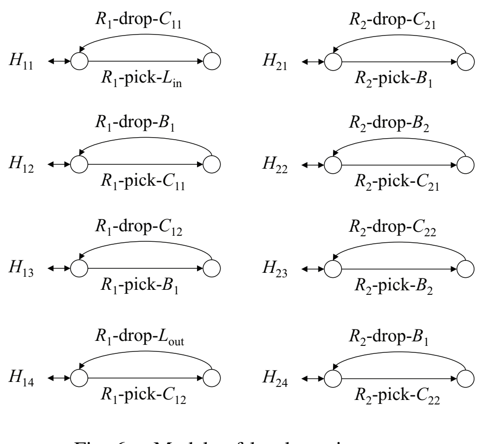

Fig. 6. Models of local requirements

VI. Conclusions

In this paper we have first introduced a distributed supervisor synthesis problem, then provided a sufficient condition to guarantee maximal permissiveness of the synthesized distributed supervisor when partial observation and nondeterminism are considered in the plant model. The proposed sufficient condition is more general and relaxed than those existing conditions in the literature, thus, potentially applicable to a larger class of systems.

References

[1] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event systems. *SIAM J. Control and Optimization*, 25(1):206–230, 1987.
[2] W.M. Wonham and P.J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, 25(3):637–659, 1987.
[3] K.C. Wong and W.M. Wonham. Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 6(3):241–273, 1996.
[4] R.J. Leduc, M. Lawford and W.M. Wonham. Hierarchical interface-based supervisory control-part II: parallel case. *IEEE Trans. Automatic Control*, 50(9):1336–1348, 2005.
[5] C. Ma and W.M. Wonham. Nonblocking supervisory control of state tree structures. *IEEE Trans. Automatic Control*, 51(5):782–793, 2006.
[6] L. Feng and W.M. Wonham. Supervisory control architecture for discrete-event systems. *IEEE Trans. Automatic Control*, 53(6):1449-1461, 2008.
[7] K. Schmidt, T. Moor and S. Perk. Nonblocking hierarchical control of decentralized discrete event systems. *IEEE Trans. Automatic Control*, 53(10):2252–2265, 2008.
[8] R. Su and J.G. Thistle. A distributed supervisor synthesis approach based on weak bisimulation. In *Proc. 8th International Workshop on Discrete Event Systems (WODES06)*, pages 64–69, 2006.
[9] W. M. Wonham. *Supervisory Control of Discrete-Event Systems*. Systems Control Group, Dept. of ECE, University of Toronto. URL: www.control.utoronto.ca/DES, July 1, 2007.
[10] R. Su, J.H. van Schuppen and J.E. Rooda. Aggregative synthesis of distributed supervisors based on automaton abstraction. *IEEE Trans. Automatic Control*, 55(7):1627-1640, 2010.
[11] R. Su, J.H. van Schuppen, J.E. Rooda and A.T. Hofkamp. Nonconflict check by using sequential automaton abstractions. *Automatica*, 46(6):968-978, 2010.
[12] S.H. Lee and K.C. Wong. Structural decentralized control of concurrent discrete-event systems. *European Journal of Control*, 8(5):477–491, 2002.
[13] K. Schmidt and C. Breindl. On maximal permissiveness of hierarchical and modular supervisory control approaches for discrete event systems. In *Proc. 9th International Workshop on Discrete Event Systems (WODES08)*, pages 462–467, 2008.
[14] R.C. Hill, D.M. Tilbury and S. Lafortune. Modular supervisory control with equivalence-based conflict resolution. In *Proc. 2008 American Control Conference (ACC08)*, pages 491–498, 2008.
[15] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems (2nd ed.)*, Springer, 2007.
[16] S. Tripakis. Undecidable problems of decentralized observation and control on regular languages. *Information Processing Letters*, 90(1):21-28, 2004.
[17] J.C. Fernandez. An implementation of an efficient algorithm for bisimulation equivalence. *Science of Computer Programming*, 13(2-3): 219-236, 1990
[18] F. Lin and W.M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44(2):173–198, 1988.
[19] R. Su, J.H. van Schuppen and J.E. Rooda. *Model abstraction of nondeterministic finite state automata in supervisor synthesis*, 55(11):2527–2541, 2011. .