Maximum permissive coordinated distributed supervisory control of nondeterministic discrete-event systems

Systems Engineering Group
Department of Mechanical Engineering
Eindhoven University of Technology
PO Box 513
5600 MB Eindhoven
The Netherlands
http://se.wtb.tue.nl/

SE Report: Nr. 2009-09

Maximum Permissive Coordinated Distributed Supervisory Control of Nondeterministic Discrete-Event Systems

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

ISSN: 1872-1567

SE Report: Nr. 2009-09
Eindhoven, November 2009
SE Reports are available via http://se.wtb.tue.nl/sereports

Abstract

In supervisor synthesis achieving nonblockingness is a major computational challenge when a target system consists of a large number of local components. To overcome this difficulty we propose an approach to synthesize a coordinated distributed supervisor, where the plant is modeled by a collection of nondeterministic finite-state automata and the requirement is modeled by a collection of deterministic finite-state automata. The synthesis is based on a previously developed automaton abstraction technique. We provide a sufficient condition, which guarantees the maximum permissiveness of the synthesized coordinated distributed supervisor. In addition, we show that the problem of finding a coordinator with the minimum number of states is NP-hard.

1 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 resulted 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 [26] [24] [14] [22] [25] [23] [6] [8] [18] [16] [15] [17].

The modular/distributed approaches with decomposable requirements are particular 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 result in only a small number of relevant local controllers to be updated. In this paper we propose an approach to synthesize a coordinated distributed supervisor. We adopt the basic setting of distributed supervisory control described in [11], where the plant is modeled by a collection of nondeterministic finite-state automata and the requirement is modeled by a collection of deterministic finite-state automata. The synthesis goal is to compute a collection of deterministic local nonblocking state-normal supervisors such that the global requirement satisfaction and nonblockingness can be achieved. We make three contributions in this paper. First, we present an approach to synthesize a coordinated distributed supervisor. Second, we show that the problem of computing a coordinator with the minimum number of states is NP-hard. Finally, we provide a sufficient condition, which guarantees the maximum permissiveness of the synthesized coordinated distributed supervisor. To illustrate the effectiveness of the proposed approach we apply it to a realistic problem.

The synthesis approach utilizes an automaton abstraction technique proposed in [10], which is different from the language-based abstraction technique presented in, e.g. [6] [18] [7] [16], and different from other automaton-based abstraction techniques provided in, e.g. [8] [19] [13] [15] [17]. In short, the proposed abstraction technique may potentially result in smaller abstracted models than those obtained by using the above mentioned language or automaton-based techniques. More detailed explanations about the comparison of abstraction techniques can be found in [10]. Since this paper is about distributed synthesis, its focus is completely different from that of [10] and [13], which are about centralized synthesis. Although the setting of distributed supervisory control of this paper is the same as that of [11], the latter aims to compute a distributed supervisor by using an aggregative approach. As a contrast, this paper is about computing a coordinated distributed supervisor. In a certain sense, the aggregative approach is a special instance of the approach proposed in this paper, where each coordinator is treated as a local supervisor. This paper also presents several results about distributed supervisor synthesis that have not been mentioned in [11] and all other aforementioned papers about modular/distributed synthesis. More explicitly, we first show the NP-hardness of computing a coordinator with the smallest state set, then provide a sufficient condition to guarantee the maximum permissiveness of the synthesized distributed supervisor (including the one obtained by using the aggregative approach) when partial observation and nondeterminism may be present in the plant model. Although in [26] [24] [14] [22] [25] [6] [16] some sufficient conditions for maximum permissiveness are also presented, they aim at a deterministic system with full observation. Furthermore, the concept of model abstraction is not used in them except for [6] and [16]. In [6] the authors assume that the plant is a shuffle system, in which the alphabets of subsystems are disjoint. They provide a sufficient condition to guarantee the maximum permissiveness of a modular supervisor computed

based on an abstracted plant model. In [16] the authors consider a general plant model, which need not be a shuffle system. To achieve the maximum permissiveness, the authors adopt the concept of mutual controllability defined in [14]. In this paper we also assume that the plant is not a shuffle system, and it is modeled by nondeterministic finite-state automata. In addition, the partial observation may be present. We extend the concepts of local control consistency defined in [16] and the mutual controllability defined in [14] so that they are applicable to a nondeterministic system. It turns out that the results of [16] and [14] about maximum permissiveness become special cases of the general results obtained in this paper.

This paper is organized as follows. In Section II we review relevant concepts, automaton operations and the general setting of distributed supervisory control described in [11]. Then in Section III we first present a synthesis approach which computes a coordinated distributed supervisor based on abstractions of nondeterministic automata, then show that the problem of finding a coordinator with the minimum number of states is NP-hard, and finally provide a sufficient condition to guarantee the maximum permissiveness of the synthesized coordinated distributed supervisor. A realistic example is provided in Section IV and conclusions are stated in Section V.

2 Necessary concepts and results of distributed supervisor synthesis

In this section we review basic concepts and results of distributed supervisor synthesis described in [10] [11], which will be used in the synthesis of coordinated distributed supervisors discussed in the next section. Because these concepts and results have been discussed in the literature, we will only provide simple explanations when we feel is necessary. More details can be found in [10] [11].

2.1 Concepts of languages, nondeterministic finite-state automata and automaton abstraction

Let Σ be a finite alphabet, and Σ^* denote the Kleene closure of Σ, i.e. the collection of all finite sequences of events taken from Σ. 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'. We use ϵ to denote the empty string of Σ^* such that for any string $s \in \Sigma^*$, $\epsilon s = s\epsilon = s$. 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. L is called *prefix closed* if $L = \overline{L}$. Given two languages $L, L' \subseteq \Sigma^*$, $LL' := \{ss' \in \Sigma^* | s \in L \wedge s' \in L'\}$.

Let $\Sigma' \subseteq \Sigma$. A mapping $P : \Sigma^* \to \Sigma'^*$ 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

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

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

$$L_1 || L_2 := P_1^{-1}(L_1) \cap P_2^{-1}(L_2) = \{s \in (\Sigma_1 \cup \Sigma_2)^* | P_1(s) \in L_1 \wedge P_2(s) \in L_2\}$$

where $P_1 : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_1^*$ and $P_2 : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_2^*$ are natural projections. Clearly, $||$ is commutative and associative. Next, we introduce automaton product and abstraction.

A *nondeterministic finite-state automaton* is a 5-tuple $G = (X, \Sigma, \xi, x_0, X_m)$, where X stands for the state set, Σ for the alphabet, $\xi : X \times \Sigma \to 2^X$ 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 any $x \in X$ and $\sigma \in \Sigma$, $\xi(x, \sigma)$ contains no more than one element, then G is called *deterministic*. Let

$$B(G) := \{s \in \Sigma^* | (\exists x \in \xi(x_0, s))(\forall s' \in \Sigma^*)\, \xi(x, s') \cap X_m = \varnothing\}$$

Any string $s \in B(G)$ can lead to a state x, from which no marker state is reachable, i.e. for any $s' \in \Sigma^*$, $\xi(x, s') \cap X_m = \varnothing$. Such a state x is called a *blocking state* of G, and we call $B(G)$ the *blocking set*. A state that is not a blocking state is called a *nonblocking state*. We say G is *nonblocking* if $B(G) = \varnothing$. For each $x \in X$, we define another set

$$N_G(x) := \{s \in \Sigma^* | \xi(x, s) \cap X_m \neq \varnothing\}$$

and call $N_G(x_0)$ the *nonblocking set* of G, which is simply the set of all strings recognized by G. For the notation simplicity, we use $N(G)$ to denote $N_G(x_0)$. It is possible that $B(G) \cap \overline{N(G)} \neq \varnothing$, due to nondeterminism. Let $\phi(\Sigma)$ be the collection of all finite-state automata over Σ. Given a language $K \subseteq \Sigma^*$, we say $G \in \phi(\Sigma)$ is a *recognizer* of L, if G is deterministic, nonblocking and $N(G) = K$.

Given two nondeterministic automata $G_i = (X_i, \Sigma_i, \xi_i, x_{0,i}, X_{m,i}) \in \phi(\Sigma_i)$ $(i = 1, 2)$, the *product* of G_1 and G_2, written as $G_1 \times G_2$, is an automaton in $\phi(\Sigma_1 \cup \Sigma_2)$ such that

$$G_1 \times G_2 = (X_1 \times X_2, \Sigma_1 \cup \Sigma_2, \xi_1 \times \xi_2, (x_{0,1}, x_{0,2}), X_{m,1} \times X_{m,2})$$

where $\xi_1 \times \xi_2 : X_1 \times X_2 \times (\Sigma_1 \cup \Sigma_2) \to 2^{X_1 \times X_2}$ is defined as follows,

$$(\xi_1 \times \xi_2)((x_1, x_2), \sigma) := \begin{cases} \xi_1(x_1, \sigma) \times \{x_2\} & \text{if } \sigma \in \Sigma_1 - \Sigma_2 \\ \{x_1\} \times \xi_2(x_2, \sigma) & \text{if } \sigma \in \Sigma_2 - \Sigma_1 \\ \xi_1(x_1, \sigma) \times \xi_2(x_2, \sigma) & \text{if } \sigma \in \Sigma_1 \cap \Sigma_2 \end{cases}$$

As usual, $\xi_1 \times \xi_2$ is extended to $X_1 \times X_2 \times (\Sigma_1 \cup \Sigma_2)^* \to 2^{X_1 \times X_2}$. By a slight abuse of notations, from now on we use $G_1 \times G_2$ to denote its *reachable* part. Next, we introduce automaton abstraction.

Definition 2.1. [10] Given $G = (X, \Sigma, \xi, x_0, X_m)$, let $\Sigma' \subseteq \Sigma$ and $P : \Sigma^* \to \Sigma'^*$ be the natural projection. A *marking weak bisimulation* relation on X with respect to Σ' is an equivalence relation $R \subseteq X \times X$ such that, $R \subseteq \{(x, x') \in X \times X | x \in X_m \iff x' \in X_m\}$ and

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

The largest marking weak bisimulation relation on X with respect to Σ' is called *marking weak bisimilarity* on X with respect to Σ', written as $\approx_{\Sigma',G}$. □

Definition 2.2. [10] Given $G = (X, \Sigma, \xi, x_0, X_m)$, let $\Sigma' \subseteq \Sigma$. The *automaton abstraction* of G with respect to the marking weak bisimulation $\approx_{\Sigma'}$ is an automaton $G/\approx_{\Sigma'} := (Z, \Sigma', \delta, z_0, Z_m)$ where

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

□

Definition 2.3. [10] Given $G_i = (X_i, \Sigma_i, \xi_i, x_{i,0}, X_{i,m})$ $(i = 1, 2)$, we say G_1 is *nonblocking preserving* with respect to G_2, denoted as $G_1 \sqsubseteq G_2$, if (1) $B(G_1) \subseteq B(G_2)$, (2) $N(G_1) = N(G_2)$, and (3) $(\forall s \in \overline{N(G_1)})(\forall x_1 \in \xi_1(x_{1,0}, s))(\exists x_2 \in \xi_2(x_{2,0}, s))\, N_{G_2}(x_2) \subseteq N_{G_1}(x_1) \wedge [x_1 \in X_{1,m} \iff x_2 \in X_{2,m}]$. We say G_1 is *nonblocking equivalent* to G_2, denoted as $G_1 \cong G_2$, if $G_1 \sqsubseteq G_2$ and $G_2 \sqsubseteq G_1$. □

To use the proposed automaton abstraction properly, we need to introduce the concept of *standardized automata*, which is defined as follows. We bring in a new event symbol τ, which does not belong to any alphabet, and is always treated as uncontrollable and unobservable. We call an automaton $G^\tau = (X, \Sigma \cup \{\tau\}, \xi, x_0, X_m)$ *standardized* if

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

A standardized automaton is nothing but an automaton, in which x_0 is not marked, τ is only defined at x_0, which only has outgoing τ transitions and no incoming transition. For notation simplicity, from now on we use Σ^τ to denote $\Sigma \cup \{\tau\}$, where $\tau \notin \Sigma$, and use $\phi(\Sigma^\tau)$ to denote the collection of all standardized automata whose alphabets are Σ^τ. We can check that, abstraction of a standardized automaton is still standardized and the product of two standardized automata is also standardized.

Proposition 2.4. [10] Given $G_1, G_2 \in \phi(\Sigma^\tau), G_3 \in \phi(\Sigma'^\tau)$, if $G_1 \sqsubseteq G_2$ then $G_1 \times G_3 \sqsubseteq G_2 \times G_3$. □

Proposition 2.5. [10] Given Σ_1^τ and Σ_2^τ, let $G_1 \in \phi(\Sigma_1^\tau)$, $G_2 \in \phi(\Sigma_2^\tau)$ and $\Sigma' \subseteq \Sigma_1 \cup \Sigma_2$. If $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma'$, then $(G_1 \times G_2)/\approx_{\Sigma'^\tau} \sqsubseteq (G_1/\approx_{(\Sigma_1 \cap \Sigma')^\tau}) \times (G_2/\approx_{(\Sigma_2 \cap \Sigma')^\tau})$. □

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 $G_i \in \phi(\Sigma_i^\tau)$ for each $i = 1, 2, \cdots, n$. How to compute $G/\approx_{\Sigma'}$ imposes a great computational difficulty. To overcome it, we propose the following algorithm. Let $I = \{1, \cdots, n\}$ for some $n \in \mathbb{N}$. For any $J \subseteq I$, let $\Sigma_J^\tau := \cup_{j \in J} \Sigma_j^\tau$.

Sequential Abstraction over Product: (SAP)
(1) Inputs: a collection $\{G_i \in \phi(\Sigma_i^\tau) | i \in I\}$ and an alphabet $\Sigma' \subseteq \cup_{i \in I} \Sigma_i^\tau$ with $\tau \in \Sigma'$.
(2) For $k = 1, 2, \cdots, n$, we perform the following computation.

- Set $J_k := \{1, 2, \cdots, k\}$, $T_k := \Sigma_{J_k}^\tau \cap (\Sigma_{I-J_k}^\tau \cup \Sigma')$.
- If $k = 1$ then $W_1 := G_1/\approx_{T_1}$
- If $k > 1$ then $W_k := (W_{k-1} \times G_k)/\approx_{T_k}$

(3) Output of SAP: W_n □

Proposition 2.6. [12] Suppose W_n is computed by SAP. Then $(\times_{i \in I} G_i)/\approx_{\Sigma'} \sqsubseteq W_n$. □

SAP allows us to obtain an abstraction of $G = \times_{i \in I} G_i$ in a sequential way. Thus, we can avoid computing G explicitly, which may be prohibitively large for systems of industrial size. Next, we discuss how to perform distributed supervisor synthesis.

2.2 Concepts and results of distributed supervisor synthesis

In this subsection we briefly review concepts and some results of distributed supervisor synthesis described in [10] and [11]. We first provide concepts of state controllability, state observability, state normality, and nonblocking supervisor, which are introduced in [10]. Then we present a distributed supervisor synthesis problem. Finally, we provide some results about distributed supervisor synthesis, whose proofs are given in [11].

Given $G = (X, \Sigma, \xi, x_0, X_m)$, 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. We now bring in the concept of *state controllability*. Let $\Sigma = \Sigma_c \cup \Sigma_{uc}$, where the disjoint subsets Σ_c and Σ_{uc} denote respectively the set of controllable events and the set of uncontrollable events. From now on, whenever τ appears in an alphabet, it is treated as an uncontrollable event. Let $L(G) := \{s \in \Sigma^* | \xi(x_0, s) \neq \varnothing\}$.

Definition 2.7. [10] Let $G = (X, \Sigma, \xi, x_0, X_m)$, $\Sigma' \subseteq \Sigma$, and $A = (Y, \Sigma', \eta, y_0, Y_m) \in \phi(\Sigma')$ and $P : \Sigma^* \to \Sigma'^*$ be the natural projection. A is *state-controllable* with respect to G and Σ_{uc} if

$$(\forall s \in L(G \times A))(\forall x \in \xi(x_0, s))(\forall y \in \eta(y_0, P(s)))\, E_G(x) \cap \Sigma_{uc} \cap \Sigma' \subseteq E_A(y)$$

□

We now introduce the concept of *state observability*. Let $\Sigma = \Sigma_o \cup \Sigma_{uo}$, where the disjoint subsets Σ_o and Σ_{uo} denote respectively the set of observable events and the set of unobservable events. Whenever τ appears in an alphabet, it is treated as an unobservable event. Let $P_o : \Sigma^* \to \Sigma_o^*$ be the natural projection.

Definition 2.8. [10] Let $G = (X, \Sigma, \xi, x_0, X_m) \in \phi(\Sigma)$, $\Sigma' \subseteq \Sigma$, and $A = (Y, \Sigma', \eta, y_0, Y_m) \in \phi(\Sigma')$. A is *state-observable* with respect to G and P_o if for any $s, s' \in L(G \times A)$ with $P_o(s) = P_o(s')$, we have

$(\forall (x, y) \in \xi \times \eta((x_0, y_0), s))(\forall (x', y') \in \xi \times \eta((x_0, y_0), s'))\, E_{G\times A}(x, y) \cap E_G(x') \cap \Sigma' \subseteq E_A(y')$

□

Notice that, if $\Sigma_o = \Sigma$, namely every event is observable, A may still not be state-observable, owing to nondeterminism. In many applications we are interested in an even stronger observability property called *state normality* which is defined as follows.

Definition 2.9. [10] Let $G = (X, \Sigma, \xi, x_0, X_m) \in \phi(\Sigma)$, $\Sigma' \subseteq \Sigma$, and $A = (Y, \Sigma', \eta, y_0, Y_m) \in \phi(\Sigma')$ and $P : \Sigma^* \to \Sigma'^*$ be the natural projection. A is *state-normal* with respect to G and P_o if for any $s \in L(G \times A)$ and $s' \in \overline{P_o^{-1}(P_o(s))} \cap L(G \times A)$, we have

$(\forall (x, y) \in \xi \times \eta((x_0, y_0), s'))(\forall s'' \in \Sigma^*)\, P_o(s's'') = P_o(s) \wedge \xi(x, s'') \neq \varnothing \Rightarrow \eta(y, P(s'')) \neq \varnothing$

□

We can check that state normality implies state observability. But the inverse statement is not true. We now introduce the concept of supervisor.

Definition 2.10. [10] Given $G \in \phi(\Sigma)$ and $H \in \phi(\Delta)$ with $\Delta \subseteq \Sigma' \subseteq \Sigma$, an automaton $S \in \phi(\Sigma')$ is a *nonblocking supervisor* of G under H, if S is deterministic and the following conditions hold:

1. $N(G \times S) \subseteq N(G \times H)$
2. $B(G \times S) = \varnothing$
3. S is state-controllable with respect to G and Σ_{uc}
4. S is state-observable with respect to G and P_o □

By the first condition of Def. 2.10, the closed-loop system $G \times S$ complies with the specification H in terms of language inclusion. Later we will use the term 'nonblocking state-normal supervisor' (NSN), when we want to emphasize that S is state-normal with respect to G and P_o. From Prop. 4 in [10] we get that

$$\mathcal{CN}(G, H) := \{S \in \phi(\Sigma) | S \text{ is a NSN supervisor of } G \text{ under } H \wedge L(S) \subseteq L(G)\}$$

contains an element $\hat{S}$ such that for all $S \in \mathcal{CN}(G, H)$, we have $N(S) \subseteq N(\hat{S})$. We call $\hat{S}$ the *supremal nonblocking state-normal supervisor* of G under H. In practice it is of our primary interest to compute such a supremal nonblocking state-normal supervisor. A computational procedure for such a supervisor is provided in [11]. We now present the concept of distributed systems.

Definition 2.11. [11] A *distributed system* with respect to given alphabets $\{\Sigma_i^\tau | i \in I\}$ is a finite collection of nondeterministic finite-state automata $\mathcal{G} := \{G_i = (X_i, \Sigma_i^\tau, \xi_i, x_{i,0}, X_{i,m}) \in \phi(\Sigma_i^\tau) | i \in I\}$. Each G_i $(i \in I)$ is called the i^{th} *component* of $\mathcal{G}$, and $\Sigma_i^\tau = \Sigma_{i,c} \cup \Sigma_{i,uc}^\tau = \Sigma_{i,o} \cup \Sigma_{i,uo}^\tau$, where disjoint subsets $\Sigma_{i,c}$ and $\Sigma_{i,uc}^\tau$ are the *controllable* and *uncontrollable* alphabets respectively, and disjoint subsets $\Sigma_{i,o}$ and $\Sigma_{i,uo}^\tau$ are the *observable* and *unobservable* alphabets respectively. For all $i, j \in I$ with $i \neq j$, we have $\Sigma_{i,uc}^\tau \cap \Sigma_{j,c} = \Sigma_{i,uo}^\tau \cap \Sigma_{j,o} = \varnothing$. 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 statement of a control problem.

Distributed Supervisory Control Problem: [11] 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 w.r.t. $\times_{i \in I} G_i$ and $\cup_{i \in I} \Sigma_{i,uc}$
4. $\times_{k \in K} S_k$ is state-normal w.r.t. $\times_{i \in I} G_i$ and $P_o : (\cup_{i \in I} \Sigma_i)^* \to (\cup_{i \in I} \Sigma_{i,uo})^*$ □

If such a collection $\mathcal{S}$ exists, then it is called a *nonblocking distributed supervisor* of $\mathcal{G}$ under $\mathcal{H}$, where each S_k is a *local supervisor* of $\mathcal{G}$ under $\mathcal{H}$. There are many ways to compute a nonblocking distributed supervisor. For example, in [11] an aggregative synthesis approach is proposed. In this paper we will present a synthesis approach that computes in parallel a set of local supervisors to take care of local specifications, then computes one or several coordinators to solve potential conflict among local supervisors. We call such a supervisor a *coordinated distributed supervisor*.

Before we discuss how to synthesize nonblocking coordinated distributed supervisors, we would like to present one more result in [11]. Our general strategy for distributed synthesis is to use automaton abstraction to simplify models. But the aforementioned automaton abstraction can only be applied to standardized automata. Therefore, we need to devise a procedure that allows abstraction-based distributed synthesis to be applicable to non-standardized automata. To this end we first introduce the concepts of *standardization* and *de-standardization*.

Definition 2.12. [11] Given $G = (X, \Sigma, \xi, x_0, X_m)$, we say $G^\tau = (X', \Sigma^\tau, \xi', x_0', X_m')$ is the *standardized version of* G if

1. $X' = X \cup \{x_0'\}$, where $x_0' \notin X$

2. $X'_m = X_m$

3. For all $x \in X'$ and $\sigma \in \Sigma^\tau$,

$$\xi'(x,\sigma) := \begin{cases} \xi(x,\sigma) & \text{if } x \in X \text{ and } \sigma \in \Sigma \\ \{x_0\} & \text{if } x = x'_0 \text{ and } \sigma = \tau \\ \varnothing & \text{otherwise} \end{cases}$$

□

The only difference between G^τ and G is that, the former contains a new state x'_0 and a new τ transition from x'_0 to x_0. From now on we use $\mu(G)$ to denote the standardized version of G. Next, we introduce the concept of *destandardization*, which is used to convert a standardized automaton into a nonstandardized one.

Definition 2.13. [11] Let $S^\tau = (Y, \Sigma^\tau, \eta, y_0, Y_m)$ be a deterministic standardized automaton. We say an automaton $S = (Y', \Sigma, \eta', y'_0, Y'_m)$ is the *destandardized version of* S^τ if

1. $Y' := Y - \{y_0\}$

2. $Y'_m := Y_m$

3. $y'_0 \in \eta(y_0, \tau)$

4. $\eta' : Y' \times \Sigma \to 2^{Y'} : (y,\sigma) \mapsto \eta'(y,\sigma) := \eta(y,\sigma)$ □

Since S^τ is deterministic, $\eta(y_0,\tau)$ contains only one element. Thus, the initial state y'_0 of S is unique, which means S is well defined. The only difference between S^τ and its destandardized version S is that, the latter contains no τ transition. From now on we use $\nu(S^\tau)$ to denote the destandardized version of S^τ. We have the following result.

Theorem 2.14. [11] Given a distributed system $\mathcal{G} = \{G_i \in \phi(\Sigma_i) | i \in I\}$ and a collection of deterministic specifications $\mathcal{H} = \{H_j \in \phi(\Delta_j) | \Delta_j \subseteq \cup_{i\in I}\Sigma_i \wedge j \in J\}$, let $\mathcal{G}^\tau := \{\mu(G_i) | i \in I\}$ be the standardized distributed system and $\mathcal{H}^\tau := \{\mu(H_j) | j \in J\}$ for the standardized deterministic specifications. If there exists a nonblocking distributed supervisor $\mathcal{S}^\tau := \{S^\tau_k \in \phi(\Gamma^\tau_k) | \Gamma^\tau_k \subseteq \cup_{i\in I}\Sigma^\tau_i \wedge k \in K\}$ of $\mathcal{G}^\tau$ under $\mathcal{H}^\tau$, then $\mathcal{S} := \{\nu(S^\tau_k) | k \in K\}$ is a nonblocking distributed supervisor of $\mathcal{G}$ under $\mathcal{H}$. □

Theorem 2.14 allows us to synthesize a nonblocking distributed supervisor of a nonstandardized distributed system under deterministic specifications. At this point we can see that, introducing the notion of τ and the concept of standardized automata, which are crucially important for automaton abstraction, does not impose any restriction on supervisor synthesis. For this reason, in the next section when we introduce synthesis of coordinated distributed supervisors, we directly start with standardized automata.

3 Synthesis of a coordinated distributed supervisor

In this section we first describe how to synthesize a coordinated distributed supervisor. Then we discuss under what conditions a coordinated distributed supervisor gains the maximum permissiveness.

3.1 Coordinated distributed control

Given a distributed system $\mathcal{G} = \{G_i \in \phi(\Sigma_i^\tau) | i \in I = \{1, 2, \cdots, n\} \wedge n \in \mathbb{N}\}$, suppose each local component G_i $(i \in I)$ has its deterministic local specification $H_i \in \phi(\Delta_i^\tau)$, where $\Delta_i \subseteq \Sigma_i$. Furthermore, there is one deterministic specification $H \in \phi(\Delta^\tau)$, where $\Delta \subseteq \cup_{i\in I}\Sigma_i$. We would like to synthesize a nonblocking coordinated distributed supervisor $\mathcal{S}$ of $\mathcal{G}$ under $\mathcal{H} := \{H, H_i | i \in I\}$. To solve this problem we need the following results.

Proposition 3.1. Let $G_1, G_2 \in \phi(\Sigma)$ be two nondeterministic plant models and $\hat{H} \in \phi(\Delta)$ a deterministic requirement with $\Delta \subseteq \Sigma$. Suppose $G_1 \sqsubseteq G_2$. Then a nonblocking state-observable (or state-normal) supervisor $S \in \phi(\Sigma)$ of G_2 under $\hat{H}$ is also a nonblocking state-observable (or state-normal) supervisor of G_1 under $\hat{H}$. □

Proof: Let $G_i = (X_i, \Sigma, \xi_i, x_{i,0}, X_{i,m})$ $(i = 1, 2)$ and $S = (Y, \Sigma, \eta, y_0, Y_m)$.
(1) First, we have

$$\begin{aligned} N(G_1 \times S) &= N(G_1)||N(S) \\ &= N(G_2)||N(S) \text{ because } G_1 \sqsubseteq G_2 \\ &= N(G_2 \times S) \\ &\subseteq N(G_2 \times \hat{H}) \text{ because } S \text{ is a nonblocking supervisor of } G_2 \text{ under } \hat{H} \\ &= N(G_2)||N(\hat{H}) \\ &= N(G_1)||N(\hat{H}) \\ &= N(G_1 \times \hat{H}) \end{aligned}$$

Therefore, we have $N(G_1 \times S) \subseteq N(G_1 \times \hat{H})$.
(2) Since $G_1 \sqsubseteq G_2$, by Prop. 2.4 we have $G_1 \times S \sqsubseteq G_2 \times S$, which means $B(G_1 \times S) \subseteq B(G_2 \times S)$. Since S is a nonblocking supervisor of G_2 under $\hat{H}$, we have $B(G_2 \times S) = \varnothing$. Thus $B(G_1 \times S) = \varnothing$.
(3) We now show S is state-controllable with respect to G_1 and Σ_{uc}. By Def. 2.7 we need to show that

$$(\forall s \in L(G_1 \times S))(\forall x_1 \in \xi_1(x_{1,0}, s))(\forall y \in \eta(y_0, P(s)))\, E_{G_1}(x_1) \cap \Sigma_{uc} \subseteq E_S(y)$$

To this end, let $s \in L(G_1 \times S)$. Since we have shown that $B(G_1 \times S) = \varnothing$, we have

$$L(G_1 \times S) = \overline{N(G_1 \times S)} = \overline{N(G_2 \times S)} = L(G_2 \times S)$$

Clearly, $E_{G_1}(x_1) \subseteq \cup_{x_2 \in \xi_2(x_{2,0}, s)} E_{G_2}(x_2)$ because $G_1 \sqsubseteq G_2$ implies that $L(G_1) \subseteq L(G_2)$. Since S is deterministic and state-controllable with respect to G_2 and Σ_{uc}, we have

$$\cup_{x_2 \in \xi_2(x_{2,0}, s)} E_{G_2}(x_2) \cap \Sigma_{uc} \subseteq E_S(y)$$

which means $E_{G_1}(x_1) \cap \Sigma_{uc} \subseteq E_S(y)$. Thus, S is state-controllable with respect to G_1 and Σ_{uc}.
(4) Suppose S is state-observable with respect to G_2 and P_o. We need to show that S is state-observable with respect to G_1 and P_o. By Def. 2.8 we need to show that, for any $s, s' \in L(G_1 \times S)$ with $P_o(s) = P_o(s')$, we have

$$(\forall (x_1, y) \in \xi_1 \times \eta((x_{1,0}, y_0), s))(\forall (x_1', y') \in \xi_1 \times \eta((x_{1,0}, y_0), s'))\, E_{G_1 \times S}(x_1, y) \cap E_{G_1}(x') \subseteq E_S(y')$$

To this end, let $s, s' \in L(G_1 \times S)$ with $P_o(s) = P_o(s')$. Since $L(G_1 \times S) = L(G_2 \times S)$, we have $s, s' \in L(G_2 \times S)$, and $E_{G_1 \times S}(x_1, y) \subseteq \cup_{(x_2,y) \in \xi_2 \times \eta((x_{2,0},y_0),s)} E_{G_2 \times S}(x_2, y)$. Since $L(G_1) \subseteq L(G_2)$, we have $E_{G_1}(x_1') \subseteq \cup_{x_2' \in \xi_2(x_{2,0},s')} E_{G_2}(x_2')$. Since S is deterministic and state-observable with respect to G_2 and P_o, we have

$$(\cup_{(x_2,y) \in \xi_2 \times \eta((x_{2,0},y_0),s)} E_{G_2 \times S}(x_2, y)) \cap (\cup_{x_2' \in \xi_2(x_{2,0},s')} E_{G_2}(x_2')) \subseteq E_S(y')$$

Thus, $E_{G_1 \times S}(x_1, y) \cap E_{G_1}(x') \subseteq E_S(y')$, which means S is state-observable with respect to G_1 and P_o.
(5) Finally, suppose S is state-normal with respect to G_2 and P_o. We need to show that S is state-normal with respect to G_1 and P_o. By Def. 2.9 we need to show that, for any $s \in L(G_1 \times S)$ and $s' \in \overline{P_o^{-1}(P_o(s))} \cap L(G_1 \times S)$, we have

$$(\forall (x_1, y) \in \xi_1 \times \eta((x_{1,0}, y_0), s'))(\forall s'' \in \Sigma^*)\, P_o(s's'') = P_o(s) \Rightarrow [\xi_1(x_1, s'') \neq \varnothing \Rightarrow \eta(y, s'') \neq \varnothing]$$

To this end, let $s \in L(G_1 \times S)$ and $s' \in \overline{P_o^{-1}(P_o(s))} \cap L(G_1 \times S)$. Since $L(G_1 \times S) = L(G_2 \times S)$, we have $s \in L(G_2 \times S)$ and $s' \in \overline{P_o^{-1}(P_o(s))} \cap L(G_2 \times S)$. For any $s'' \in \Sigma^*$, if $P_o(s's'') = P_o(s)$ and $\xi_1(x_1, s'') \neq \varnothing$, we get that $s's'' \in L(G_1) \subseteq L(G_2)$. Thus, there exists $(x_2, y) \in \xi_2 \times \eta((x_{2,0}, y_0), s')$ such that $P_o(s's'') = P_o(s)$ and $\xi_2(x_2, s'') \neq \varnothing$. Since S is deterministic and state-normal with respect to G_2 and P_o, we have $\eta(y, s'') \neq \varnothing$. Thus, S is state-normal with respect to G_1 and P_o.
From (1)-(5) we get that, S is a nonblocking state-observable (or state-normal) supervisor of G_2 under $\hat{H}$ implies that S is a nonblocking state-observable (or state-normal) supervisor of G_1 under $\hat{H}$. ■

Prop. 3.1 indicates that, if a plant G_1 is nonblocking preserving with respect to G_2, then a nonblocking supervisor for G_2 is also a nonblocking supervisor for G_1. In many cases it may be easier to obtain G_2 than G_1. For example, it is easier to use SAP to compute an abstraction, than simply compute the product first then perform the abstraction operation on the product. The latter abstracted model (denoted as G_1) is nonblocking preserving with respect to the former abstracted plant model (denoted by G_2).

Proposition 3.2. Suppose we have a collection of alphabets $\{\Sigma_i^\tau | i \in I\}$ for some finite index set I, and a collection of components $\{G_i \in \phi(\Sigma_i^\tau) | i \in I\}$. Let $\Sigma' \subseteq \cup_{i \in I} \Sigma_i^\tau$ such that $\cup_{i,j \in I : i \neq j} \Sigma_i^\tau \cap \Sigma_j^\tau \subseteq \Sigma'$. Then $(\times_{i \in I} G_i)/\approx_{\Sigma'} \sqsubseteq \times_{i \in I}(G_i/\approx_{\Sigma_i^\tau \cap \Sigma'})$. □

Proof: We use induction on the size of I. When $|I| = 2$, by Prop. 2.5 the result holds. Suppose it holds for $|I| = n$. We show that it also holds for $|I| = n + 1$ as follows:

$$\begin{aligned}
(\times_{i \in I} G_i)/\approx_{\Sigma'} &= (\times_{i \in I - \{j\}} G_i \times G_j)/\approx_{\Sigma'} \\
&\sqsubseteq ((\times_{i \in I - \{j\}} G_i)/\approx_{(\cup_{i \in I - \{j\}} \Sigma_i^\tau) \cap \Sigma'}) \times (G_j/\approx_{\Sigma_j^\tau \cap \Sigma'}) \\
&\quad \text{since } \Sigma_j \cap (\cup_{i \in I - \{j\}} \Sigma_i) \subseteq \Sigma' \text{ and by Prop. 2.5} \\
&\sqsubseteq \times_{i \in I - \{j\}}(G_i/\approx_{\Sigma_i^\tau \cap \Sigma'}) \times (G_j/\approx_{\Sigma_j^\tau \cap \Sigma'}) \\
&\quad \text{because } |I - \{j\}| = n \text{ and by the induction hypothesis and Prop. 2.4} \\
&= \times_{i \in I}(G_i/\approx_{\Sigma_i^\tau \cap \Sigma'})
\end{aligned}$$

Thus, the proposition is true. ■

Prop. 3.2 is an extension of Prop. 2.5 over the product of more than two standardized finite-state automata. We will use this result in the following theorem, which is the first main result of this paper.

Theorem 3.3. Given a distributed system $\mathcal{G} = \{G_i \in \phi(\Sigma_i^\tau) | i \in I\}$ and a collection of requirements $\mathcal{H} = \{H_i \in \phi(\Delta_i^\tau) | \Delta_i^\tau \subseteq \Sigma_i^\tau \wedge i \in I\} \cup \{H \in \phi(\Delta^\tau) | \Delta^\tau \subseteq \cup_{i\in I}\Sigma_i^\tau\}$, suppose for each G_i we have a nonblocking state-observable (or state-normal) supervisor $S_i \in \phi(\Sigma_i^\tau)$ under H_i. Let $\Sigma' \subseteq \cup_{i\in I}\Sigma_i^\tau$ such that $\cup_{i,j:i\neq j}\Sigma_i^\tau \cap \Sigma_j^\tau \subseteq \Sigma'$ and $\Delta^\tau \subseteq \Sigma'$. For each $i \in I$ suppose we have $W_i \in \phi(\Sigma_i^\tau \cap \Sigma')$ such that $(G_i \times S_i)/\approx_{\Sigma_i^\tau\cap\Sigma'} \sqsubseteq W_i$. Let $S = (Y, \Sigma', \eta, y_0, Y_m) \in \phi(\Sigma')$ be a nonblocking state-observable (or state-normal) supervisor of $\times_{i\in I}W_i$ under H. Then $S \times_{i\in I} S_i$ is a nonblocking state-observable (or state-normal) supervisor of $\times_{i\in I}G_i$ under $H \times_{i\in I} H_i$. □

Proof: Let $G_i = (X_i, \Sigma_i^\tau, \xi_i, x_{i,0}, X_{i,m})$ and $S_i = (Y_i, \Sigma_i^\tau, \eta_i, y_{i,0}, Y_{i,m})$ for each $i \in I$, and $S = (Y, \Sigma', \eta, y_0, Y_m)$. By Corollary 3.2 we get that $(\times_{i\in I}(G_i \times S_i))/\approx_{\Sigma'} \sqsubseteq \times_{i\in I}((G_i \times S_i)/\approx_{\Sigma_i^\tau\cap\Sigma'})$. Since $(G_i \times S_i)/\approx_{\Sigma_i^\tau\cap\Sigma'} \sqsubseteq W_i$, by Prop. 2.4 we get that

$$(\times_{i\in I}(G_i \times S_i))/\approx_{\Sigma'} \sqsubseteq \times_{i\in I}((G_i \times S_i)/\approx_{\Sigma_i^\tau\cap\Sigma'}) \sqsubseteq \times_{i\in I}W_i$$

Since S is a nonblocking state-observable (or state-normal) supervisor of $\times_{i\in I}W_i$ under H, by Prop. 3.1 we get that, S is a nonblocking state-observable (or state-normal) supervisor of $(\times_{i\in I}(G_i \times S_i))/\approx_{\Sigma'}$ under H. By Theorem 3 in [10] we get that, S is a nonblocking state-observable (or state-normal) supervisor of $\times_{i\in I}(G_i \times S_i)$ under H, which means

$$N(\times_{i\in I}G_i \times S \times_{j\in I} S_j) = N(\times_{i\in I}(G_i \times S_i) \times S) \subseteq N(\times_{i\in I}(G_i \times S_i) \times H)$$

Since S_i is a nonblocking supervisor of G_i under H_i, we have $N(G_i \times S_i) \subseteq N(G_i \times H_i)$. Thus,

$$N(\times_{i\in I}G_i \times S \times_{j\in I} S_j) \subseteq N(\times_{i\in I}(G_i \times H_i) \times H) = N(\times_{i\in I}G_i \times H \times_{j\in I} H_j)$$

Furthermore, we have $B(\times_{i\in I}G_i \times S \times_{j\in I} S_j) = B(\times_{i\in I}(G_i \times S_i) \times S) = \varnothing$.
Next, we show that $S \times_{i\in I} S_i$ is state-controllable with respect to $\times_{i\in I}G_i$ and $\cup_{i\in I}\Sigma_{i,uc}^\tau$. For notational brevity, let $\hat{S} = S \times_{i\in I} S_i$, $\hat{G} = \times_{i\in I}G_i$, $\hat{\xi} = \times_{i\in I}\xi_i$, $\hat{\eta} = \eta \times_{i\in I} \eta_i$ and $\Sigma_{uc} = \cup_{i\in I}\Sigma_{i,uc}$. By Def. 2.7 we need to show that

$$(\forall s \in L(\hat{G} \times \hat{S}))(\forall \hat{x} \in \hat{\xi}(\hat{x}_0, s))(\forall \hat{y} \in \hat{\eta}(\hat{y}_0, s))\, E_{\hat{G}}(\hat{x}) \cap \Sigma_{uc}^\tau \subseteq E_{\hat{S}}(\hat{y})$$

To this end, let $s \in L(\hat{G} \times \hat{S})$, $\hat{x} = (x_1, x_2, \cdots, x_n)$ and $\hat{y} = (y, y_1, y_2, \cdots, y_n)$. For each $i \in I$, let $P_i : (\cup_{j\in I}\Sigma_j^\tau)^* \to (\Sigma_i^\tau)^*$ be the natural projection. For each $\sigma \in E_{\hat{G}}(\hat{x}) \cap \Sigma_{uc}^\tau$, if $\sigma \in \Sigma_i^\tau$, then by the assumption (A1) we have $\sigma \in \Sigma_{i,uc}^\tau$. Furthermore, we get that $\sigma \in E_{G_i}(x_i) \cap \Sigma_{i,uc}^\tau$. Since S_i is deterministic and state-controllable with respect to G_i and $\Sigma_{i,uc}^\tau$, we get that $\eta_i(y_i, \sigma) \neq \varnothing$. Thus, $\sigma \in E_{\times_{i\in I}(G_i\times S_i)}(x_1, y_1, \cdots, x_n, y_n)$. Since S is state-controllable with respect to $\times_{i\in I}(G_i \times S_i)$ and Σ_{uc}^τ, if $\sigma \in \Sigma'$, we get that $\eta(y, \sigma) \neq \varnothing$. Thus, $\hat{\eta}(\hat{y}, \sigma) \neq \varnothing$, which means $\sigma \in E_{\hat{S}}(\hat{y})$. Therefore, $E_{\hat{G}}(\hat{x}) \cap \Sigma_{uc}^\tau \subseteq E_{\hat{S}}(\hat{y})$.
Next, assume that S_i is state-observable with respect to G_i and $P_{i,o} : (\Sigma_i^\tau)^* \to \Sigma_{i,o}^*$, and S is state-observable with respect to $\times_{i\in I}(G_i \times S_i)$ and $P_o : (\cup_{i\in I}\Sigma_i^\tau)^* \to (\cup_{i\in I}\Sigma_{i,o})^*$. We need to show that $\hat{S}$ is state-observable with respect to $\hat{G}$ and P_o. By Def. 2.8 we need to show that, for any $s, s' \in L(\hat{G} \times \hat{S})$ with $P_o(s) = P_o(s')$, we have

$$(\forall(\hat{x}, \hat{y}) \in \hat{\xi} \times \hat{\eta}((\hat{x}_0, \hat{y}_0), s))(\forall(\hat{x}', \hat{y}') \in \hat{\xi} \times \hat{\eta}((\hat{x}_0, \hat{y}_0), s'))\, E_{\hat{G}\times\hat{S}}(\hat{x}, \hat{y}) \cap E_{\hat{G}}(\hat{x}') \subseteq E_{\hat{S}}(\hat{y}')$$

To this end, let $s, s' \in L(\hat{G} \times \hat{S})$ with $P_o(s) = P_o(s')$, $\hat{x} = (x_1, \cdots, x_n)$, $\hat{x}' = (x_1', \cdots, x_n')$, $\hat{y} = (y, y_1, \cdots, y_n)$ and $\hat{y}' = (y', y_1', \cdots, y_n')$. For each $\sigma \in E_{\hat{G}\times\hat{S}}(\hat{x}, \hat{y}) \cap E_{\hat{G}}(\hat{x}')$, if $\sigma \in \Sigma_i^\tau$, then we get that $\sigma \in E_{G_i\times S_i}(x_i) \cap E_{G_i}(x_i')$. Since S_i is deterministic and state-observable with respect to G_i and $P_{i,o}$, by the assumption (A1) we can derive that $\eta_i(y_i', \sigma) \neq \varnothing$. Thus, $\sigma \in E_{\times_{i\in I}(G_i\times S_i)}(x_1', y_1', \cdots, x_n', y_n')$. Since S is state-observable with respect to $\times_{i\in I}(G_i \times S_i)$ and P_o, if $\sigma \in \Sigma'$, we get that $\eta(y', \sigma) \neq \varnothing$. Thus, $\hat{\eta}(\hat{y}', \sigma) \neq \varnothing$, which means $\sigma \in E_{\hat{S}}(\hat{y}')$. Therefore, $E_{\hat{G}\times\hat{S}}(\hat{x}, \hat{y}) \cap E_{\hat{G}}(\hat{x}') \subseteq E_{\hat{S}}(\hat{y}')$.
Finally, assume that S_i is state-normal with respect to G_i and $P_{i,o}$, and S is state-normal with respect to $\times_{i\in I}(G_i \times S_i)$ and P_o. We need to show that $\hat{S}$ is state-normal with

respect to $\hat{G}$ and P_o. By Def. 2.9 we need to show that, for any $s \in L(\hat{G} \times \hat{S})$ and $s' \in \overline{P_o^{-1}(P_o(s))} \cap L(\hat{G} \times \hat{S})$, we have

$$(\forall(\hat{x}, \hat{y}) \in \hat{\xi} \times \hat{\eta}((\hat{x}_0, \hat{y}_0), s'))(\forall s'' \in \Sigma^*)\, P_o(s's'') = P_o(s) \Rightarrow [\hat{\xi}(\hat{x}, s'') \neq \varnothing \Rightarrow \hat{\eta}(\hat{y}, s'') \neq \varnothing]$$

To this end, let $s \in L(\hat{G} \times \hat{S})$ and $s' \in \overline{P_o^{-1}(P_o(s))} \cap L(\hat{G} \times \hat{S})$. Suppose $P_o(s's'') = P_o(s)$ and $\hat{\xi}(\hat{x}, s'') \neq \varnothing$. We need to show that $\hat{\eta}(\hat{y}, s'') \neq \varnothing$. Let $\hat{x} = (x_1, \cdots, x_n)$, $\hat{y} = (y, y_1, \cdots, y_n)$, and $P_i : (\cup_{j \in I} \Sigma_j^\tau)^* \to (\Sigma_i^\tau)^*$, $P' : (\cup_{j \in I} \Sigma_j^\tau)^* \to \Sigma'^*$ be the natural projection. Then we have $P_i(s) \in L(G_i \times S_i)$, $P_i(s') \in \overline{P_o^{-1}(P_{i,o}(P_i(s)))} \cap L(G_i \times S_i)$. Furthermore, by the assumption (A1) we have $P_{i,o}(P_i(s's'')) = P_{i,o}(P_i(s))$ and $\xi_i(x_i, P_i(s'')) \neq \varnothing$. Since S_i is deterministic and state-normal with respect to G_i and $P_{i,o}$, we get that $\eta_i(y_i, P_i(s'')) \neq \varnothing$. Thus, $\times_{i \in I} \xi_i \times \eta_i((x_1, y_1, \cdots, x_n, y_n), s'') \neq \varnothing$. Since S is state-normal with respect to $\times_{i \in I}(G_i \times S_i)$ and P_o, we get that $\eta(y, P'(s'')) \neq \varnothing$. Thus, $\hat{\eta}(\hat{y}, s'') \neq \varnothing$.■

By Theorem 3.3 we can perform the following distributed synthesis, as illustrated in Figure 1. We first synthesize a local supervisor S_i for each component G_i so that the

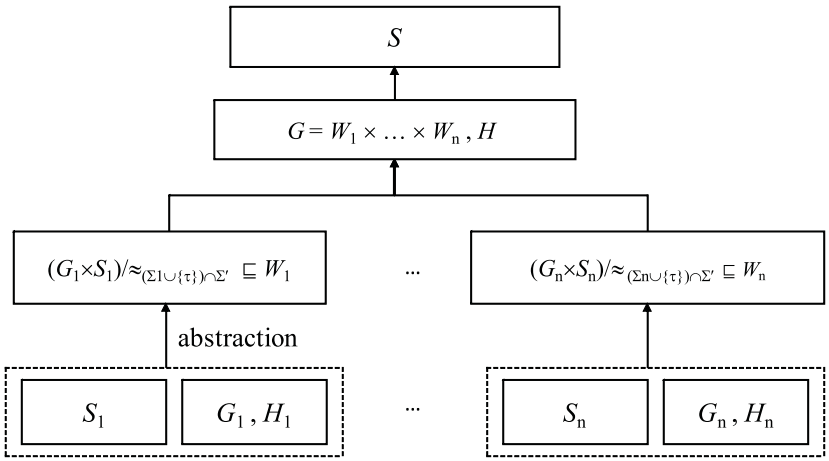

Figure 1: Synthesis of Coordinated Distributed Supervisor

local specification H_i can be enforced. Then we compute an abstraction so that we can synthesize a local supervisor to take care of H. In practical applications sometimes a specification, say H_i, may cover several local components, say $\{G_{il} \in \phi(\Sigma_{il}^\tau) | l = 1, \cdots, r\}$, in the sense that, $\Delta_i \subseteq \cup_{l=1}^r \Sigma_{ij}$. In this case, we can compute $G_i := \times_{l=1}^r G_{il}$ and treat it as a local component so that H_i is defined for G_i. Thus, the setting in Theorem 3.3 is general enough. The reason that we bring in W_i in Theorem 3.3 is because, when G_i consists of many small components, e.g. $\{G_{il} \in \phi(\Sigma_{il}^\tau) | l = 1, \cdots, r\}$, computing $(G_i \times S_i)/\approx_{\Sigma_i^\tau \cap \Sigma'}$ may be feasible only through a sequential procedure, e.g. using the SAP. In that case, the outcome of that procedure may not be exactly equal to $(G_i \times S_i)/\approx_{\Sigma_i^\tau \cap \Sigma'}$. The theorem says that, as long as $(G_i \times S_i)/\approx_{\Sigma_i^\tau \cap \Sigma'}$ is nonblocking preserving with respect to W_i, which is computed by an appropriate procedure, e.g. the SAP, then synthesizing a local supervisor based on $\{W_i | i \in I\}$ will result in a nonblocking supervisor for the original local components. In Theorem 3.3 we call each S_i a *local supervisor* of $\mathcal{G}$ and S a *coordinator* of $\mathcal{G}$, which is mainly used to coordinate local supervisors $\{S_i | i \in I\}$ to avoid conflict. The existence of S gives rise to the term *coordinated distributed supervisor*. Of course, S itself is a supervisor, which enforces the specification H. The structure in Figure 1 can be treated as one module of a large system. Thus, a multiple-level multiple-coordinator

distributed supervisor can be computed in the same spirit. For example, after obtaining $\{S_i | i \in I\} \cup \{S\}$, we can compute an appropriate abstraction of $\times_{i\in I}(G_i \times S_i) \times S$ (by using the proposed SAP) so that high level local supervisors and/or coordinators can be synthesized.

In Theorem 3.3 we only require that $\cup_{i,j:i\neq j}\Sigma_i^\tau \cap \Sigma_j^\tau \subseteq \Sigma'$, namely Σ' should contain every event that is shared by at least two components. Usually there is more than one choice of Σ', and each choice leads to a coordinator. It is interesting to know whether we can find a nonempty coordinator, whose state set is the smallest one among those of all possible nonempty coordinators. This can be formulated into the following problem. Given an automaton $G \in \phi(\Sigma)$, we use $|G|$ to denote the size of the state set of G.

Minimum Supervisor Synthesis Problem (MSS): Let $\mathcal{G} = \{G_i \in \phi(\Sigma_i^\tau) | i \in I\}$ be a distributed system and $H \in \phi(\Delta^\tau)$ be a requirement with $\Delta^\tau \subseteq \cup_{i\in I}\Sigma_i^\tau$, define a set

$$\mathcal{S}(\mathcal{G}, H) := \{S | (\exists \Sigma' \subseteq \cup_{i\in I}\Sigma_i^\tau)\ \cup_{i,j:i\neq j} \Sigma_i^\tau \cap \Sigma_j^\tau \subseteq \Sigma' \wedge \Delta^\tau \subseteq \Sigma' \wedge S \in \phi(\Sigma') \wedge S \text{ is a nonblocking supervisor of } \mathcal{G} \text{ under } H \wedge |S| > 0\}$$

Find $S \in \mathcal{S}(\mathcal{G}, H)$ such that, for all $S' \in \mathcal{S}(\mathcal{G}, H)$, we have $|S| \leq |S'|$. Such a S is called a *minimum supervisor of $\mathcal{G}$ under H.* □

In [20] the authors present the **minimum supervisor reduction problem (MSR)**, which says that given a deterministic plant $G = (X, \Sigma, \xi, x_0, X_m)$ and a supervisor $S = (Y, \Sigma, \eta, y_0, Y_m)$ of G, find a supervisor $S' = (Y', \Sigma, \eta', y_0', Y_m')$ of G with the minimum number of states, which is control equivalent to S with respect to G, i.e. $N(G) \cap N(S) = N(G) \cap N(S')$ and $L(G) \cap L(S) = L(G) \cap L(S')$. It has been proved in [20] that solving the MSR is NP-hard. We will show that solving the MSS is as hard as solving the MSR. To this end, we present a procedure that reduces the MSR to the MSS.

1. Inputs:
   - a deterministic plant $G = (X, \Sigma, \xi, x_0, X_m)$
   - a nonempty supervisor $S = (Y, \Sigma, \eta, y_0, Y_m)$ of G
2. Let $G \times S = (Z, \Sigma, \xi \times \eta, z_0 := (x_0, y_0), Z_m)$, where
   - $Z := \{(x, y) \in X \times Y | (\exists s \in \Sigma^*)\ (x, y) \in \xi \times \eta(z_0, s)\}$
   - $Z_m := Z \cap (X_m \times Y_m)$
3. Enumerate elements of $Z \times \Sigma$ as $\{(z_1, \sigma_1), (z_1, \sigma_2), \cdots, (z_{|Z|}, \sigma_{|\Sigma|})\}$, where $|Z|$ and $|\Sigma|$ denote the sizes of Z and Σ respectively.
4. Construct a new automaton

   $G' = (Z \cup X \cup Z \times \Sigma \cup \{d\}, \Sigma' := \Sigma \cup \{\gamma_{(z_1,\sigma_1)}, \cdots, \gamma_{(z_{|Z|},\sigma_{|\Sigma|})}\}, \xi', z_0, Z_m \cup X_m \cup \{d\})$

   where $\{\gamma_{(z_1,\sigma_1)}, \cdots, \gamma_{(z_{|Z|},\sigma_{|\Sigma|})}\} \cap \Sigma = \varnothing$, $\Sigma_{uc}' := \Sigma_{uc} \cup \{\gamma_{(z_1,\sigma_1)}, \cdots, \gamma_{(z_{|Z|},\sigma_{|\Sigma|})}\}$ and $\Sigma_o' = \Sigma$, namely events $\{\gamma_{(z_1,\sigma_1)}, \cdots, \gamma_{(z_{|Z|},\sigma_{|\Sigma|})}\}$ are uncontrollable in G' and only events of G are observable in G'. For all $w \in Z \cup X \cup Z \times \Sigma \cup \{d\}$ and all

$\sigma \in \Sigma \cup \{\gamma_{(z_1,\sigma_1)}, \cdots, \gamma_{(z_{|Z|},\sigma_{|\Sigma|})}\}$, define

$$\xi'(w,\sigma) := \begin{cases} \xi \times \eta(w,\sigma) & \text{if } w \in Z \wedge \sigma \in \Sigma \wedge \xi \times \eta(w,\sigma) \neq \varnothing \\ \xi(x,\sigma) & \text{if } w = (x,y) \in Z \wedge \sigma \in \Sigma \wedge \eta(y,\sigma) = \varnothing \vee w = x \in X \\ \{(z_i,\sigma_j)\} & \text{if } w = z_i \wedge \sigma = \gamma_{(z_i,\sigma_j)} \\ \{d\} & \text{if } w = (z_i,\sigma_j) \wedge \sigma = \sigma_j \\ \varnothing & \text{otherwise} \end{cases}$$

5. Solve the MSS with $\mathcal{G} = \{G'\}$ and H being a recognizer of $N(G \times S)$. Suppose the solution is $S' = (Y', \Gamma, \eta', y_0', Y_m')$ whose alphabet is $\Gamma \subseteq \Sigma \cup \{(z_1,\sigma_1),(z_1,\sigma_2),\cdots,(z_{|Z|},\sigma_{|\Sigma|})\}$.
6. Construct a new automaton S^* by simply remove all transitions from S', whose labels are in the set of $\{(z_1,\sigma_1),(z_1,\sigma_2),\cdots,(z_{|Z|},\sigma_{|\Sigma|})\}$ and selflooping events of $\Sigma - \Sigma'$ at each state of S'.
7. Output: S^* □

The key part of this procedure is how to construct G' and H such that, the resulting controllable sublanguage of $N(G')$ under H is unique. If this is true, then no matter what supervisor S' we have, since the language $N(G' \times S') = N(G' \times S)$ and $L(G' \times S') = L(G' \times S)$, S is control equivalent to S, which means, if we apply a solver of MSS on this problem, the outcome is simply a solution to the MSR with G' and S. Notice that in the above procedure, $N(G \times S)$ is the controllable sublanguage of $N(G)$ under some unspecified requirement. By adding those uncontrollable events $\{(z_1,\sigma_1),(z_1,\sigma_2),\cdots,(z_{|Z|},\sigma_{|\Sigma|})\}$ we can see that, there is no smaller controllable sublanguage of $N(G')$ under H than $N(G' \times S)$, because any removal of a string from $N(G' \times S)$ will result in a blocking state reachable by an uncontrollable event from the set $\{(z_1,\sigma_1),(z_1,\sigma_2),\cdots,(z_{|Z|},\sigma_{|\Sigma|})\}$ when we try to achieve the state-normality property. So applying a solver for MSS on this is equal to solving a MSS on G' and S. The only trick left is to convert a reduced supervisor for G' and S back to a reduced supervisor for G and S. But this can be achieved by simply removing all transitions labeled with events of $\{(z_1,\sigma_1),(z_1,\sigma_2),\cdots,(z_{|Z|},\sigma_{|\Sigma|})\}$, and selflooping all events of $\Sigma - \Sigma'$. Mode details are provided in the proof of the following result.

Proposition 3.4. The procedure always has a nonempty output S^*, which is a solution to the MSR. □

Proof: By the definition of G', we can check that, S is a nonblocking supervisor of G' under H. Furthermore, we have that, for any nonblocking state-normal supervisor S' of G' under H, we have $N(G' \times S') = N(G' \times S)$ and $L(G' \times S') = L(G' \times S)$. This can be shown as follows. Suppose it is not true. Then there exists a supervisor S'' such that $N(G' \times S'') \subset N(G' \times S)$, which means there exists $s \in N(G' \times S)$ but $s \notin N(G' \times S'')$. Clearly, there exists $s'\sigma \leq s$ such that $s' \in N(G' \times S')$ but $s'\sigma \notin N(G' \times S'')$. Let $w \in \xi'(z_0, s')$. Clearly, $w \notin Z \times \Sigma$ because, otherwise, w will be a blocking state. Thus, $w \in Z$. But since $\gamma_{(w,\sigma)} \in \Sigma_{uc}' \cap \Sigma_{uo}'$, the event σ at state w cannot be disabled - otherwise, $G' \times S''$ is not state-controllable with respect to G' and Σ_{uc}', and not state-normal with respect to G' and $P_o : \Sigma'^* \to \Sigma_{uo}'^*$. This contradicts the assumption that $s'\sigma \notin N(G' \times S'')$ but $s' \in N(G' \times S'')$. Thus, S'' is control equivalent to S with respect to G'. Notice that, $N(G') = N_1(G') \cup N_2(G')$, where $N_1(G')$ contains all strings of $N(G')$, which contains only events of Σ, and $N_2(G')$ contains the remaining strings of $N(G')$. We can check that $N(G' \times S) = N_1(G')||N(S) \cup N_2(G')||N(S) = N(G \times S) \cup N_2(G')||N(S)$. Similarly, $N(S') = N_1(S') \cup N_2(S')$. Thus, $N(G' \times S') = N_1(G')||N_1(S') \cup (N_2(G')||N_1(S') \cup N_1(G')||N_2(S') \cup N_2(G')||N_2(S'))$. Clearly, $N_1(G')||N_1(S') = N(G \times S)$. We now show

that $N_1(G')||N_1(S') = N_1(G')||N(S^*) = N(G \times S^*)$, which is clear based on the definition of S^*. Similarly, we have $L(G \times S) = L(G \times S^*)$. Thus, S^* is control equivalent to S with respect to G. Since removing and selflooping transitions will not create new states, we have $|S^*| \leq |S'|$. To show that S^* is the minimum supervisor for the problem of the MSR, suppose it is not true. Then there exists another supervisor $\tilde{S} \in \phi(\Sigma)$ such that $\tilde{S}$ is control equivalent to S and $|\tilde{S}| < |S^*|$. We can easily check that $\tilde{S}$ is a nonblocking supervisor of $\mathcal{G}$ under H, namely $\tilde{S} \in \mathcal{S}(\mathcal{G}, H)$. But this means $|\tilde{S}| \geq |S'| \geq |S^*|$ - contradiction. Thus, S^* is a solution to the MSR. ■

Corollary 3.5. Solving the MSS is NP-hard. □

Proof: We can check that, every step in the above procedure is polynomial-time. Thus, the MSR can be polynomial-time reduced to the MSS. By Prop. 3.4 we know that the above procedure can solve the MSR. If solving the MSS is not NP-hard, then so is solving the MSR, which unfortunately has been shown in [20] to be NP-hard. Thus, solving the MSS must be NP-hard. ■

Corollary 3.5 simply confirms our intuition - it is computationally intensive to find a coordinator with the minimum number of states in a general setting. It is an interesting question whether there exists a heuristic rule that can lead to a small coordinator with only polynomial-time computational effort.

3.2 Maximum permissiveness of coordinated distributed control

In general, given a distributed system $\mathcal{G}$ and a set of requirements $\mathcal{H}$, a coordinated 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. The reason is that, some local supervisor may be "over conservative" in the sense that, it tries to prevent some "bad" string which exists only in some local component(s) but does not exist in the compositional behavior of $\mathcal{G}$. Such a bad string is called a *phantom* string, which, if seen locally, exists, but, if seen globally, does not exist. For example, suppose we have two components $G_1 \in \phi(\Sigma_1)$ and $G_2 \in \phi(\Sigma_2)$, where $\Sigma_1 = \Sigma_{1,uc} = \Sigma_{1,o} = \{a\}$, $\Sigma_2 = \Sigma_{2,uc} = \Sigma_{2,o} = \{a, b\}$, $L(G_1) = \overline{L_m(G_1)} = \{\epsilon, a\}$ and $L(G_2) = \overline{L_m(G_2)} = \{\epsilon, b\}$. Suppose the requirement is $\{H_1 \in \phi(\Delta_1), H_2 \in \phi(\Delta_2)\}$ with $\Delta_1 = \{a\}$, $\Delta_2 = \{b\}$, $L(H_1) = \overline{N(H_1)} = \{\epsilon\}$ and $L(H_2) = \overline{L_m(H_2)} = L(G_2)$. Since a is uncontrollable, we can easily see that, there is no local supervisor to control G_1 such that H_1 can be enforced. Thus, if we apply the aforementioned synthesis approach, there is no coordinated distributed supervisor. But if we compute the composition of G_1 and G_2, we can see that, string a will never appear. Thus, a monolithic supervisor exists, which recognizes $N(G_2)$. Here, the reason that there is no coordinated distributed supervisor is because of the existence of a phantom string a in G_1. To achieve the same permissiveness between a distributed supervisor and a monolithic supervisor, part of a sufficient condition is that, there exist no phantom strings in any local component, which is captured by, e.g. the concept of *mutual controllability* in the literature [14]. Since we use abstraction to derive a local supervisor based on an abstract model, to guarantee that such a local supervisor has the same permissiveness as the one based on the original model, we need to make sure that abstraction will not reduce our means for control, namely if in the original model we can remove an undesirable string by disabling a certain event, then in the abstract model we can also disable the same event to remove (the projected image of) that string. One simple condition is that, all controllable and observable events are included in the

abstraction alphabet. Then it is guaranteed that, the supremal nonblocking supervisor of the abstract model is also the supremal nonblocking supervisor of the original model under the same requirement. An improvement on such an intuitive condition can be found in the concept of *output control consistency* [6] or the concept of *local control consistency* [16]. Those aforementioned concepts (i.e. mutual controllability, output control consistency or local control consistency) are applicable to systems modeled by languages (or equivalently, deterministic automata). In this section we will extend them to the framework of nondeterministic finite-state automata.

Definition 3.6. [10] An automaton $G = (X, \Sigma, \xi, x_0, X_m)$ is *marking aware* with respect to $\Sigma' \subseteq \Sigma$, if

$$(\forall x \in X - X_m)(\forall s \in \Sigma^*)\, \xi(x, s) \cap X_m \neq \varnothing \Rightarrow P(s) \neq \epsilon$$

where $P : \Sigma^* \to \Sigma'^*$ is the natural projection. □

The concept of marking awareness is used to guarantee that the proposed automaton abstraction will not create extra blocking behaviors. Thus, the maximum permissiveness of a coordinated distributed supervisor can be achieved by using the proposed automaton abstraction. This concept is not needed if we directly use the standard quotient construction based on the weak bisimilarity, as done in, e.g. [8].

Definition 3.7. 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 $x \in X$ and all $s \in ((\Sigma - \Sigma')^*(\Sigma_{uc} \cap \Sigma'))^*$,

$$\xi(x, s) \neq \varnothing \Rightarrow (\exists s' \in ((\Sigma_{uc} - \Sigma')^*(\Sigma_{uc} \cap \Sigma'))^*)\, P(s) = P(s') \wedge \xi(x, s') \neq \varnothing$$

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

The concept of control consistency is a direct extension of the concept of *local control consistency* presented in [16] to fit in the nondeterministic setting. If G is control consistent, then at all state x and all string s, if s contains some uncontrollable event(s) in Σ', then there exists another string s', which contains only uncontrollable events such that its projected image over $\Sigma_{uc} \cap \Sigma'$ is the same as the project image of s over $\Sigma_{uc} \cap \Sigma'$. Informally speaking, in s' there is no controllable event not belonging to Σ' that can block the occurrence of an uncontrollable event in Σ' by disabling itself. We can check that, when G is deterministic, the concept of control consistency consumes the concept of local control consistency.

Definition 3.8. Given a distributed system $\mathcal{G} = \{G_i \in \phi(\Sigma_i^\tau) | i \in I\}$ and a deterministic requirement $H \in \phi(\Delta^\tau)$ with $\Delta^\tau \subseteq \cup_{i \in I} \Sigma_i^\tau$, let $\Sigma' \subseteq \cup_{i \in I} \Sigma_i^\tau$ with $\Delta^\tau \subseteq \Sigma'$, and

$$P : (\cup_{i \in I} \Sigma_i^\tau)^* \to \Sigma'^*,\ P_o : (\cup_{i \in I} \Sigma_i^\tau)^* \to (\cup_{i \in I} \Sigma_{i,o})^* \text{ and } P_o' : \Sigma'^* \to (\Sigma' \cap (\cup_{i \in I} \Sigma_{i,o}))^*$$

be the natural projections. We say $\mathcal{G}$ is *indistinguishable* with respect to H and Σ' if the following holds: for all $t \in \overline{N(\times_{i \in I}(G_i / \approx_{\Sigma_i^\tau \cap \Sigma'}) \times H)}$ and $t' \in \overline{N(\times_{i \in I}(G_i / \approx_{\Sigma_i^\tau \cap \Sigma'}))} - \overline{N(\times_{i \in I}(G_i / \approx_{\Sigma_i^\tau \cap \Sigma'}) \times H)}$ or $t' \in B(\times_{i \in I}(G_i / \approx_{\Sigma_i^\tau \cap \Sigma'}) \times H)$, if $P_o'(t) = P_o'(t')$ then for all $s, s' \in L(\times_{i \in I} G_i)$ with $P(s) = t$, $P(s') = t'$, we have $P_o(s) = P_o(s')$. □

The concept of indistinguishableness specifies that, if there are two strings t and t' not distinguishable based on observations in the abstracted model, where t is "good", i.e. $t \in \overline{N(\times_{i\in I}(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'})\times H)}$, and t' is "bad", i.e. it does not satisfy the requirement, namely either $t' \in \overline{N(\times_{i\in I}(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'}))} - \overline{N(\times_{i\in I}(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'})\times H)}$ or $t' \in B(\times_{i\in I}(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'})\times H))$, then for all strings s and s' with $P(s) = t$ and $P(s') = t'$, they are not distinguishable based on the observations in the original plant model G. To guarantee that $\mathcal{G}$ is indistinguishable with respect to H and Σ', one simple condition is $\cup_{i\in I}\Sigma_{i,o} \subseteq \Sigma'$, namely every observable event is contained in Σ'. When every event in $\mathcal{G}$ is observable, $\mathcal{G}$ may not be necessarily indistinguishable with respect to H and Σ', owing to nondeterminism. In the case of full observation, we can impose the following concept, which is derived from the concept of natural observer [3].

Definition 3.9. Given a nondeterministic automaton $G = (X, \Sigma^\tau, \xi, x_0, X_m) \in \phi(\Sigma^\tau)$ and an alphabet $\Sigma' \subseteq \Sigma^\tau$ with $\tau \in \Sigma'$, we say $G/\approx_{\Sigma'}$ is an *observer* of G with respect to Σ' if

$$(\forall t \in N(G/\approx_{\Sigma'}))(\forall s \in L(G))(\forall x \in \xi(x_0, s))\, P(s) \le t \Rightarrow (\exists s' \in \Sigma^*)\, \xi(x, s') \cap X_m \neq \varnothing \wedge P(ss') = t$$

where $P : \Sigma^* \to \Sigma'^*$ is the natural projection. □

We can check that, if for every $i \in I$, $\Sigma_{i,o} = \Sigma_i$ and $G_i/\approx_{\Sigma_i^\tau\cap\Sigma'}$ is an observer of G_i with respect to $\Sigma_i^\tau \cap \Sigma'$, and $\cup_{i,j:i\neq j}(\Sigma_i^\tau \cap \Sigma_j^\tau) \subseteq \Sigma'$, then $\mathcal{G}$ is indistinguishable with respect to H and Σ'. This can be easily shown that, for all $t \in \overline{N(\times_{i\in I}(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'})\times H)}$ and $t' \in \overline{N(\times_{i\in I}(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'}))} - \overline{N(\times_{i\in I}(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'})\times H)}$ or $t' \in B(\times_{i\in I}(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'})\times H)$, we have $P'_o(t) \neq P'_o(t')$. We are still investigating whether there exists a condition to guarantee that $\mathcal{G}$ is indistinguishable with respect to H and Σ' no matter whether full or partial observation presents. We now present our second major result.

Theorem 3.10. Given a distributed system $\mathcal{G} = \{G_i \in \phi(\Sigma_i^\tau) | i \in I\}$ and a requirement $H \in \phi(\Delta^\tau)$ with $\Delta^\tau \subseteq \cup_{i\in I}\Sigma_i^\tau$, let $\Sigma' \subseteq \Sigma^\tau$ with $\cup_{i,j\in I:i\neq j}\Sigma_i^\tau \cap \Sigma_j^\tau \subseteq \Sigma'$ and $\Delta^\tau \subseteq \Sigma'$. Let $S \in \phi(\Sigma')$ be the supremal nonblocking state-normal supervisor of $\times_{i\in I}(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'})$ under H. If $\mathcal{G}$ is indistinguishable with respect to H and Σ', and for each $i \in I$, G_i is marking aware with respect to $\Sigma' \cap \Sigma_i^\tau$ and control consistent with respect to $\Sigma' \cap \Sigma_i^\tau$, then a recognizer of $||_{i\in I}N(G_i)||N(S)$ is the supremal nonblocking state-normal supervisor of $\times_{i\in I}G_i$ under H. □

Proof: Let $G_i = (X_i, \Sigma_i^\tau, \xi_i, x_{i,0}, X_{i,m})$, $G_i/\approx_{\Sigma_i^\tau\cap\Sigma'} = (Z_i, \Sigma_i^\tau \cap \Sigma', \delta_i, z_{i,0}, Z_{i,m})$, $S = (Y, \Sigma', \eta, y_0, Y_m)$, $S' = (Y', \cup_{i\in I}\Sigma_i^\tau, \eta', y'_0, Y'_m)$ and $H = (W, \Delta^\tau, \psi, w_0, W_m)$. Let $P_o : (\cup_{i\in I}\Sigma_i^\tau)^* \to (\cup_{i\in I}\Sigma_{i,o})^*$, $P'_o : \Sigma'^* \to (\Sigma' \cap (\cup_{i\in I}\Sigma_{i,o}^\tau))^*$ and $P : (\cup_{i\in I}\Sigma_i^\tau)^* \to \Sigma'^*$ be the natural projection. By Theorem 3 in [10] we know that, S is a nonblocking state-normal supervisor of G under H. So we only need to show that S is supremal. If the supremal nonblocking state-normal supervisor of $\times_{i\in I}G_i$ under H is empty, then so is S. We assume that the supremal nonblocking state-normal supervisor of $\times_{i\in I}G_i$ under H is not empty, and S is not supremal. Then there exists a nonblocking state-normal supervisor $S' \in \phi(\cup_{i\in I}\Sigma_i)$ of $\times_{i\in I}G_i$ under H such that

$$(||_{i\in I}N(G_i)||N(S')) - (||_{i\in I}N(G_i)||N(S)) \neq \varnothing$$

Let $s \in (||_{i\in I}N(G_i)||N(S')) - (||_{i\in I}N(G_i)||N(S)) \neq \varnothing$. Since all automata are standardized, we know that, there exists $s' \in (\cup_{i\in I}\Sigma_i^\tau)^*$ and $\sigma \in \cup_{i\in I}\Sigma_i$ such that $s'\sigma \le s$, $s' \in \overline{||_{i\in I}N(G_i)||N(S)}$ but $s'\sigma \notin \overline{||_{i\in I}N(G_i)||N(S)}$. Since $s'\sigma \in \overline{||_{i\in I}N(G_i)}$, we get that, $\sigma \in \Delta^\tau \subseteq \Sigma'$. Since S is a supervisor, we know that $\sigma \in \cup_{i\in I}\Sigma_{i,c}$. Since

$s \in ||_{i\in I}N(G_i)||N(H)$, we get that, $P(s) \in ||_{i\in I}N(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'})||N(H)$. Thus, $P(s'\sigma) = P(s')\sigma \in \overline{||_{i\in I}N(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'})||N(H)}$, which means there exist $t \in ((\cup_{i\in I}\Sigma_{i,uc}^\tau)\cap\Sigma')^*$, $t' \in \Sigma'^*$, and $z' = (< x_1' >, \cdots, < x_n' >) \in \prod_{i\in I} Z_i$ such that $z' \in \delta_1 \times \cdots \times \delta_n((z_{1,0}, \cdots, z_{n,0}), t')$, $P(s')\sigma t \in L(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'})$, $P_o'(t') = P_o'(P(s')\sigma t)$ and one of the following cases hold. Without the loss of generality, let $I = \{1, 2, \cdots, n\}$.
Case 1: there exists $w \in W$ such that $(z', w) \in \delta_1 \times \cdots \times \delta_n \times \psi((z_{1,0}, \cdots, z_{n,0}, w_0), t')$ and for all $t'' \in \Sigma'^*$,

$$\delta_1 \times \cdots \times \delta_n \times \psi((z', w), t'') \cap (Z_{1,m} \times \cdots \times Z_{n,m} \times W_m) = \varnothing$$

In other words, $t' \in B((G_i/\approx_{\Sigma_i^\tau\cap\Sigma'}) \times H)$. Since all G_i's are marking aware with respect to $\Sigma_i^\tau \cap \Sigma'$, we get that, there exist $x_i'' \in < x_i' > (i \in I)$ and $s'' \in (\cup_{i\in I}\Sigma_i^\tau)^*$ with $P(s'') = t'$ such that $(x_1'', \cdots, x_n'', w) \in \xi_1 \times \cdots \times \xi_n \times \psi((x_{1,0}, \cdots, x_{n,0}, w_0), s'')$ and for all $s''' \in (\cup_{i\in I}\Sigma_i^\tau)^*$,

$$\xi_1 \times \cdots \times \xi_n \times \psi((x_1', \cdots, x_n', w), s''') \cap (X_{1,m} \times \cdots \times X_{n,m} \times W_m) = \varnothing$$

Since each G_i is control consistent with respect to $\Sigma_i^\tau \cap \Sigma'$, we get that, there exists $s'''' \in (((\cup_{i\in I}\Sigma_{i,uc}^\tau) - \Sigma')^*(\Sigma' \cap (\cup_{i\in I}\Sigma_{i,uc})))^*$ such that $P(s'''') = t$ and $s'\sigma s'''' \in L(\times_{i\in I}G_i)$. There are two possibilities, either $P(s')\sigma t \in \overline{N(\times_{i\in I}(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'}) \times H)}$, or $P(s')\sigma t \notin \overline{N(\times_{i\in I}(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'}) \times H)}$. If the latter case is true, then since all G_i's are marking aware with respect to $\Sigma_i^\tau \cap \Sigma'$, we can derive that, $s'\sigma s'''' \notin \overline{N(\times_{i\in I}G_i \times H)}$, which means S' is not state-controllable. If the former is true, since $\mathcal{G}$ is indistinguishable with respect to H and Σ', we have $P_o(s'') = P_o(s'\sigma s'''')$. Since S' is a nonblocking state-normal supervisor of $\times_{i\in I}G_i$ under H, we get that, $s'\sigma \notin \overline{||_{i\in I}N(G_i)||N(S')}$. But this contradicts the assumption that $s'\sigma \in \overline{||_{i\in I}N(G_i)||N(S')}$.
Case 2: $t' \in \overline{||_{i\in I}N(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'})} - \overline{||_{i\in I}N(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'})||N(H)}$. Since $\cup_{i,j\in I:i\neq j}\Sigma_i^\tau \cap \Sigma_j^\tau \subseteq \Sigma'$, and by a result in [10] we have

$$P(||_{i\in I}N(G_i)) = ||_{i\in I}P(N(G_i)) = ||_{i\in I}N(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'})$$

and

$$P(||_{i\in I}N(G_i)||N(H)) = ||_{i\in I}P(N(G_i))||N(H) = ||_{i\in I}N(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'})||N(H)$$

we have $t' \in P(\overline{||_{i\in I}N(G_i)}) - P(\overline{||_{i\in I}N(G_i)||N(H)})$. Thus, there must exist $s'' \in \overline{||_{i\in I}N(G_i)} - \overline{||_{i\in I}N(G_i)||N(H)}$ such that $P(s'') = t'$. By using a similar argument as for Case 1, we get that $s'\sigma \notin \overline{||_{i\in I}N(G_i)||N(S')}$ - contradiction.
Thus, we have that $(||_{i\in I}N(G_i)||N(S')) - (||_{i\in I}N(G_i)||N(S)) = \varnothing$, meaning a recognizer of the language $||_{i\in I}N(G_i)||N(S)$ is the supremal nonblocking state-normal supervisor of $\times_{i\in I}G_i$ under H. □

Theorem 3.10 is about the maximum permissiveness of a supervisor computed based on an abstracted model. In Theorem 3.10 the sufficient condition consists of three parts: (1) $\mathcal{G}$ is indistinguishable with respect to H and Σ'; (2) each component G_i is marking aware with respect to $\Sigma_i^\tau \cap \Sigma'$; (3) each component G_i is control consistent with respect to $\Sigma_i^\tau \cap \Sigma'$. Among those parts, (1) is used to deal with partial observation (or the state-normality property). Without this part, we can find a counter example, in which the supremal nonblocking state-normal supervisor of an abstract plant is not the supremal nonblocking state-normal supervisor of the original plant. (2) is to guarantee that the projections of marked behaviors in the original plant (i.e. $\times_{i\in I}G_i$) are also marked behaviors in the abstracted model (i.e. $\times_{i\in I}(G_i/\approx_{\Sigma_i^\tau\cap\Sigma'})$). This condition is used only for the special automaton abstraction proposed in this paper. If we use a standard quotient approach to construct an abstraction, e.g. automaton abstractions defined in [8] [19], then (2) can be dropped from Theorem 3.10. (3) is an extension of the local control consistency proposed in [16] in order to deal with nondeterminism. Compared with the

results in [6] and [16], Theorem 3.10 drops the requirement of L-observer because the automaton abstraction will ensure some necessary properties. Furthermore, it deals with partial observation and nondeterminism. Thus, it is a significant extension of the results in [6] and [16].

It is interesting to know under what conditions a nonblocking distributed supervisor achieves the maximum permissiveness. To this end we first extend the concept of mutual controllability so that it is applicable to nondeterministic models.

Definition 3.11. Given a distributed system $\mathcal{G} = \{G_i = (X_i, \Sigma_i^\tau, \xi_i, x_{i,0}, X_{i,m}) \in \phi(\Sigma_i^\tau) | i \in I\}$, for each $i, j \in I$ let $P_{ij} : (\Sigma_i^\tau)^* \to (\Sigma_i^\tau \cap \Sigma_j^\tau)^*$ be the natural projection. We say $\mathcal{G}$ is *mutually controllable* if for each $i, j \in I$ with $i \neq j$, for all $s_i \in (\Sigma_i^\tau)^*$ and $s_j \in (\Sigma_i^\tau)^*$ 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. We now present our last major result.

Theorem 3.12. Given a distributed system $\mathcal{G} = \{G_i = (X_i, \Sigma_i^\tau, \xi_i, x_{i,0}, X_{i,m}) \in \phi(\Sigma_i^\tau) | i \in I\}$ and a collection of requirements $\mathcal{H} = \{H_i = (W_i, \Delta_i^\tau, \psi_i, w_{i,0}, W_{i,m}) \in \phi(\Delta_i^\tau) | \Delta_i \subseteq \Sigma_i \wedge i \in I\} \cup \{H \in \phi(\Delta^\tau) | \Delta \subseteq \cup_{i \in I} \Sigma_i\}$, suppose for each G_i we have the supremal nonblocking state-normal supervisor $S_i \in \phi(\Sigma_i^\tau)$ under H_i. Let $\Sigma' \subseteq \cup_{i \in I} \Sigma_i^\tau$ such that $\cup_{i,j:i \neq j} \Sigma_i^\tau \cap \Sigma_j^\tau \subseteq \Sigma'$ and $\Delta^\tau \subseteq \Sigma'$. Let $S = (Y, \Sigma', \eta, y_0, Y_m) \in \phi(\Sigma')$ be the supremal nonblocking state-normal supervisor of $\times_{i \in I}((G_i \times S_i)/\approx_{\Sigma_i^\tau \cap \Sigma'})$ under H. If $\mathcal{G}$ is indistinguishable with respect to H and Σ', and for each $i \in I$, $\Sigma_{i,o} \supseteq \cup_{j \in I, j \neq i}(\Sigma_i^\tau \cap \Sigma_j^\tau)$, G_i is marking aware with respect to $\Sigma_i^\tau \cap \Sigma'$, $G_i \times S_i$ is control consistent with respect to $\Sigma_i^\tau \cap \Sigma'$, and $\mathcal{G}$ is mutually controllable with respect to $\{H_i | i \in I\}$, then $S \times_{i \in I} S_i$ is the supremal nonblocking state-normal supervisor of $\times_{i \in I} G_i$ under $H \times_{i \in I} H_i$. □

Proof: By Theorem 3.3 we know that, $S \times_{i \in I} S_i$ is a nonblocking state-normal supervisor of $\times_{i \in I} G_i$ under $H \times_{i \in I} H_i$. So we only need to show that it is supremal. To this end, let $S' \in \phi(\cup_{i \in I} \Sigma_i^\tau)$ be the supremal nonblocking state-normal supervisor of $\times_{i \in I} G_i$ under $H \times_{i \in I} H_i$. It suffices to show that,

$$N(S') \subseteq ||_{i \in I} N(S_i) \tag{1}$$

because then by Theorem 3.10 we can derive that, $S \times_{i \in I} S_i$ is the supremal nonblocking state-normal supervisor of $\times_{i \in I} G_i$ under $H \times_{i \in I} H_i$. For each $i \in I$ let $P_i : (\cup_{j \in I} \Sigma_i^\tau)^* \to (\Sigma_i^\tau)^*$ be the natural projection. We will show that $P_i(N(S')) \subseteq N(S_i)$. Suppose it is not true. Then there exists $s \in P_i(N(S')) - N(S_i)$. Since $s \in P_i(N(S')) \subseteq N(G_i)$, and $s \notin N(S_i)$, we can derive that, there exist $s' \leq P_i(s)$, $t \in (\Sigma_{i,uc}^\tau))^*$, $t' \in (\Sigma_i^\tau)^*$, and $x_i, x_i' \in X_i$ such that $x_i' \in \xi_i(x_{i,0}, t')$, $P_{i,o}(t') = P_{i,o}(s't)$ and one of the following cases hold. Without the loss of generality, suppose $I = \{1, 2, \cdots, n\}$.
Case 1: there exists $w_i \in W_i$ such that $(x_i', w_i) \in \xi_i \times \psi_i((x_{i,0}, w_{i,0}), t')$ and for all $t'' \in (\Sigma_i^\tau)^*$,

$$\xi_i \times \psi_i((x_i', w_i), t'') \cap (X_{i,m} \times W_{i,m}) = \varnothing$$

Since $s' \leq P_i(s)$, there must exists $\hat{s} \leq s$ such that $s' = P_i(\hat{s})$. Since $t \in (\Sigma^\tau_{i,uc})^*$ and $\mathcal{G}$ is mutually controllable, we get that $\hat{s}t \in ||_{j\in I}L(G_j)$. Since $P_{i,o}(t') = P_{i,o}(s't)$ and $\Sigma_{i,o} \supseteq \cup_{j\in I, j\neq i}(\Sigma^\tau_i \cap \Sigma^\tau_j)$ and $\mathcal{G}$ is mutually controllable, we get that, there exists $\hat{s}' \in ||_{j\in I, j\neq i}\{P_j(\hat{s})\}||\{t'\}$ such that $P_o(\hat{s}') = P_o(\hat{s}t)$, where $P_o : (\cup_{j\in I}\Sigma^\tau_j)^* \to (\cup_{j\in I}\Sigma_j)^*$ is the natural projection. Clearly, there exist $(x_1, \cdots, x'_i, \cdots, x_n) \in \xi_1 \times \cdots \xi_n((x_{1,0}, \cdots, x_{n,0}), \hat{s}')$ such that for all $u' \in (\cup_{j\in I}\Sigma^\tau_j)^*$,

$$\xi_1 \times \cdots \times \xi_n \times \psi_1 \times \cdots \times \psi_n((x_1, \cdots, x'_i, \cdots, x_n, w_1, \cdots, w_i, \cdots, w_n), u') \cap (X_m \times W_m) = \varnothing$$

where $X_m := X_{1,m} \times \cdots X_{n,m}$ and $W_m := W_{1,m} \times \cdots \times W_{n,m}$. Thus, we can derive that $\hat{s} \notin \overline{N(S')}$ because S' is a state-normal nonblocking supervisor, which means $s \notin N(S')$ - contradiction.
Case 2: $t' \in \overline{N(G_i)} - \overline{N(G_i)||N(H_i)}$. By using a similar argument as for Case 1, we can derive that, $\hat{s}' \in \overline{||_{j\in I}N(G_j)} - \overline{||_{j\in I}(N(G_j)||N(H_j))}$ with $P_i(\hat{s}') = t'$. Then we can derive that $\hat{s} \notin \overline{N(S')}$ because S' is a state-normal nonblocking supervisor, which means $s \notin N(S')$ - contradiction.
Therefore, we have $P_i(N(S')) \subseteq N(S_i)$, which means $N(S') \subseteq ||_{i\in I}N(S_i)$. □

In addition to the conditions prescribed in Theorem 3.10, to guarantee the maximum permissiveness of a distributed supervisor, Theorem 3.12 also requires that $\mathcal{G}$ is mutually controllable and furthermore, for each $i \in I$, $\Sigma_{i,o} \supseteq \cup_{j\in I, j\neq i}(\Sigma^\tau_i \cap \Sigma^\tau_j)$, which means all shared events are observable. The latter does not appear in the corresponding results in [6] and [16] because they do not deal with partial observation (recall that nondeterminism can be captured by partial observation). In the case of full observation the condition $\Sigma_{i,o} \supseteq \cup_{j\in I, j\neq i}(\Sigma^\tau_i \cap \Sigma^\tau_j)$ is automatically satisfied. Thus, Theorem 3.12 is an extension of results in [6] and [16], and is valid for distributed supervisory control of a nondeterministic distributed plant where partial observation may be present.

To illustrate the effectiveness of the proposed synthesis approach for computing coordinated distributed supervisors, we apply it to the following cluster tool example.

4 Example - a cluster tool

A cluster tool is an integrated manufacturing system used for wafer processing. It consists of *load locks* for wafer entering and leaving the system, *chambers*, where wafers are processed, *buffers* between different clusters in the system, and *transportation robots* for moving wafers in the system [21]. We consider the following cluster tool depicted in Figure 2, which consists of one entering load lock (L_{in}) and one exit load lock (L_{out}), nine chambers (C_{11}, C_{12}, C_{21}, C_{22}, C_{31}, C_{32}, C_{41}, C_{42}, C_{43}), three one-slot buffers (B_1, B_2, B_3), and four transportation robots (R_1, R_2, R_3 and R_4). 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 illustration, we choose the following routing sequence: $L_{in} \to C_{11} \to B_1 \to C_{21} \to B_2 \to C_{31} \to B_3 \to C_{41} \to C_{42} \to C_{43} \to B_3 \to C_{32} \to B_2 \to C_{22} \to B_1 \to C_{12} \to L_{out}$. Without supervision the system may be blocked owing to wafers competing for buffer slots. Our goal is to synthesize a coordinated distributed supervisor that can guarantee continuous wafer processing, namely blocking should never happen. To this end, we first model the system as follows.

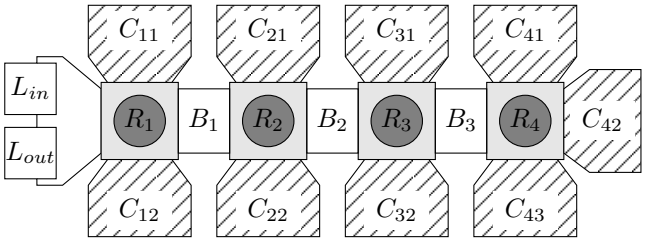

Figure 2: Example 1: Structure of cluster tool

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 3 depicts the models of load locks. We assume that in each chamber a wafer is first dropped in by a relevant

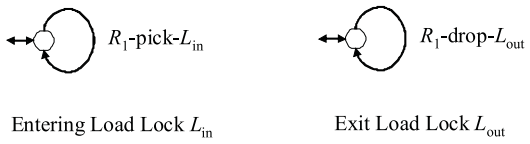

Figure 3: Example 1: Load locks

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 4, where, when $i = 1, 2, 3$, we have $j = 1, 2$,

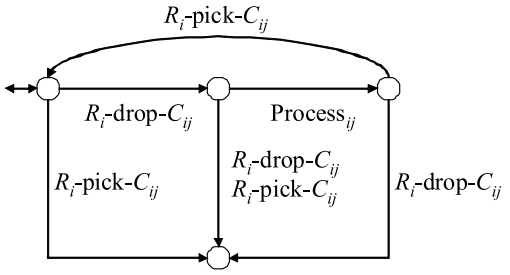

Figure 4: Example 1: Model of chamber C_{ij}

and when $i = 4$, we have $j = 1, 2, 3$. Notice that each 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 5. Finally we model each buffer B_i $(i = 1, 2, 3)$ as a component, whose model is provided in Figure 6. It says that, buffer overflow or underflow will result in deadlock. In these models we assume that all events of the robots are controllable and observable, and events $\{\text{Process}_{ij} | i = 1, 2, 3, 4 \wedge j = 1, 2\} \cup \{\text{Process}_{43}\}$ are uncontrollable and unobservable. The local requirements are depicted in Figure 7.

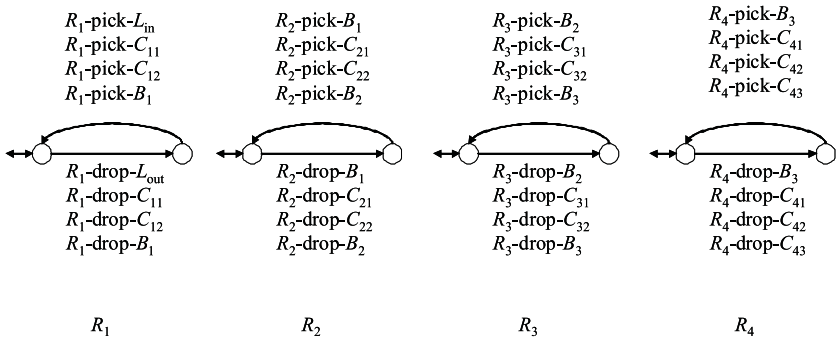

Figure 5: Example 1: Models of robots

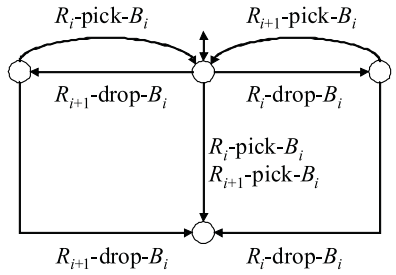

Figure 6: Example 1: Model of buffer B_i

To synthesize a coordinated distributed supervisor, we first partition the system into four 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. Module 3 consists of C_{31}, C_{32}, B_2, B_3 and R_3. Module 4 consists of C_{41}, C_{42}, C_{43}, B_3 and R_4. The system partition is depicted in Figure 8. With Module i $(i = 1, 2, 3, 4)$ 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 example, for Module 1 we first compute the standardized plant model

$$G_1^\tau := \mu(L_{in}) \times \mu(L_{out}) \times \mu(C_{11}) \times \mu(C_{12}) \times \mu(B_1) \times \mu(R_1)$$

then we compute the local requirement

$$H_1^\tau := \mu(H_{11}) \times \mu(H_{12}) \times \mu(H_{13}) \times \mu(H_{14})$$

Based on G_1^τ and H_1^τ we synthesize the local supervisor S_1^τ. Then we apply the same procedure to synthesize local supervisors for other modules. The computational results are listed as follows:

$$\begin{array}{c} G_1^\tau \ (73,\ 277)\ ;\ H_1^\tau \ (17,\ 65)\ ;\ S_1^\tau \ (58,\ 119) \\ G_2^\tau \ (285,\ 1229)\ ;\ H_2^\tau \ (17,\ 65)\ ;\ S_2^\tau \ (138,\ 327) \\ G_3^\tau \ (285,\ 1229)\ ;\ H_3^\tau \ (17,\ 65)\ ;\ S_3^\tau \ (138,\ 327) \\ G_4^\tau \ (209,\ 729)\ ;\ H_4^\tau \ (17,\ 65)\ ;\ S_4^\tau \ (112,\ 222) \end{array}$$

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

Next, we compute a local coordinator for Module 1 and Module 2, and a local coordinator

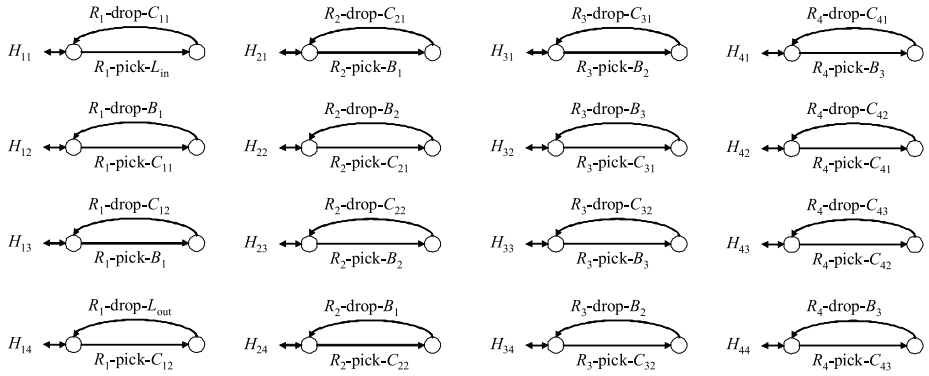

Figure 7: Example 1: Models of local requirements

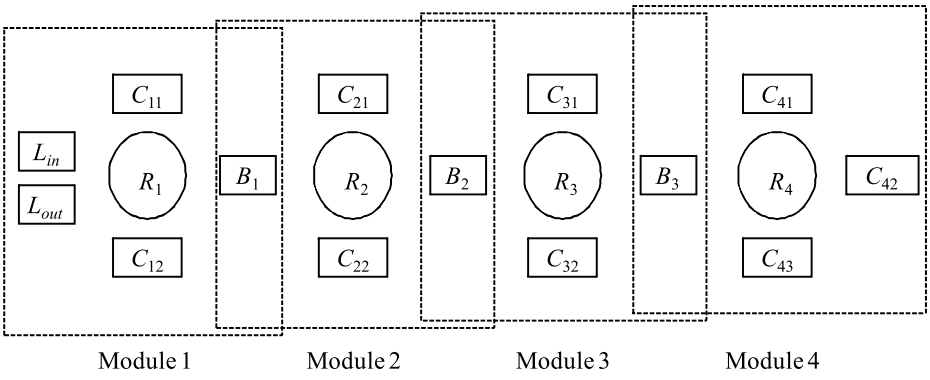

Figure 8: Example 1: Partition of the system

for Module 3 and Module 4. To this end, we first compute an abstraction of each module. We use a heuristic rule to choose an abstraction alphabet for each module, which is described as follows: the alphabet should contain all "boundary" events, i.e. events shared between the current module and other modules, and between the current module and the environment. For example, the abstraction alphabet for Module 1 consists of events shared between Module 1 and Module 2 (which are events of B_1) and events $R_1 - \text{pick} - L_{in}$ and $R_1 - \text{drop} - L_{out}$ because these two events describe how Module 1 gets input from the external world and generates output to the external world. The abstraction alphabet for Module 2 is the set of all events shared between Module 2 and its two neighbors Module 1 and Module 3. Similarly, we have the abstraction alphabets for Module 3 and Module 4 respectively. As an illustration we describe how to obtain the coordinator S_{12} for Module 1 and Module 2. We first compute two abstractions:

$$W_1 := (G_1^\tau \times S_1^\tau)/\approx_{\hat{\Sigma_1}^\tau} (22, 58); W_2 := (G_2^\tau \times S_2^\tau)/\approx_{\hat{\Sigma_2}^\tau} (53, 139)$$

where $\hat{\Sigma_i}^\tau$ $(i = 1, 2)$ is the local abstraction alphabet for Module i. Then we compute the product of two abstractions $G_{12}^\tau := W_1 \times W_2$. Since there is no extra requirement for G_{12}^τ, we create a nominal one, which simply allows all events of two abstractions. After that, we synthesize the supremal nonblocking state-normal supervisor S_{12}^τ of G_{12}^τ under the nominal requirement, which is treated as a coordinator of Module 1 and Module 2. The computational results are listed as follows:

$$G_{12}^{\tau}\ (363,\ 1418);\ S_{12}^{\tau}\ (102,\ 240)$$

Similarly, the computational results for synthesizing coordinator S_{34}^{τ} are listed as follows:

$$W_3\ (53,\ 139);\ W_4\ (15,\ 24);\ G_{12}^{\tau}\ (241,\ 705);\ S_{12}^{\tau}\ (36,\ 58)$$

Finally, we compute a coordinator to prevent potential conflict among all four modules. To this end, we first compute two abstractions

$$(G_1^{\tau} \times G_2^{\tau} \times S_1^{\tau} \times S_2^{\tau} \times S_{12}^{\tau})/\approx_{\hat{\Sigma}_{12}^{\tau}} \sqsubseteq W_{12}\ (127,\ 843)$$
$$(G_3^{\tau} \times G_4^{\tau} \times S_3^{\tau} \times S_4^{\tau} \times S_{34}^{\tau})/\approx_{\hat{\Sigma}_{34}^{\tau}} \sqsubseteq W_{34}\ (15,\ 24)$$

where $\hat{\Sigma}_{12}^{\tau}$ and $\hat{\Sigma}_{34}^{\tau}$ are abstraction alphabets for the composition of Module 1 and Module 2 and the composition of Module 3 and Module 4 respectively. Here, W_{12} and W_{34} are computed by the SAP. Again, since there is no extra requirement, we create a nominal one. Then we synthesize the supremal nonblocking state-normal supervisor for $W_{12} \times W_{34}$ under the nominal requirement, which is treated as the top-level coordinator. The computational results are listed as follows:

$$W_{12} \times W_{34}\ (575,\ 3935);\ S_{1234}^{\tau}\ (28,\ 48)$$

Figure 9 depicts the local supervisors and coordinators with their corresponding modules.

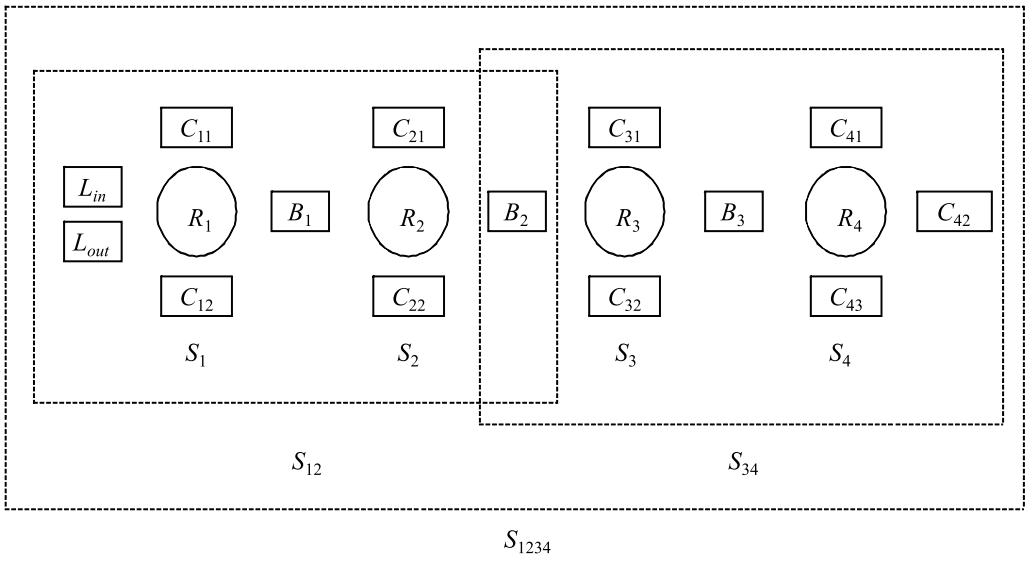

Figure 9: Example 1: Computation of multiple-level coordinators

The final distributed supervisor is $\mathcal{S} = \{\nu(S_1^{\tau}), \nu(S_2^{\tau}), \nu(S_3^{\tau}), \nu(S_4^{\tau}), \nu(S_{12}^{\tau}), \nu(S_{34}^{\tau}), \nu(S_{1234}^{\tau})\}$. We can check that, the maximum size of all computational results during the synthesis is (575, 3935). As a comparison, the monolithic plant model has about 2.68×10^8 states. Thus, the proposed coordinated distributed synthesis approach is computationally more effective than the monolithic synthesis approach.

5 Conclusions

In this paper we have introduced a coordinated distributed supervisor synthesis approach based on abstractions of nondeterministic finite-state automata. The main advantage of this approach is its simplicity and potentially low computational complexity in contrast to existent distributed synthesis approaches based on observers. When a module contains a large number of components, we can apply the proposed SAP procedure to obtain an abstraction, which may significantly reduce the computational complexity. Because supervisor synthesis is done in a local fashion, high complexity incurred by synchronous product of a large number of components may be avoided. In addition, a certain degree of implementation flexibility can be achieved in terms of reusing some local supervisors when the structure of a target system changes. Although it is practically attractive to compute a coordinator with the minimum number of states, we have shown that finding such a coordinator is NP-hard. In general, the coordinated distributed supervisor is less permissive than a monolithic supervisor. In this paper we have provided a sufficient condition that guarantees the maximum permissiveness of the synthesized coordinated distributed supervisor when partial observation and nondeterminism may be present in the plant model.

Acknowledgement: We would like to thank Dr. Albert T. Hofkamp of the Systems Engineering Group at Eindhoven University of Technology for coding all algorithms mentioned in this paper. We have used his code to generate the solution of the example of Section IV.

Bibliography

[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. Computationally efficient supervisor design: modularity and abstraction. In *Proc. 8th International Workshop on Discrete Event Systems (WODES06)*, pages 3–8, 2006.

[7] K. Schmidt, H. Marchand and B. Gaudin. Modular and decentralized supervisory control of concurrent discrete event systems using reduced system models. In *Proc. 8th International Workshop on Discrete Event Systems (WODES06)*, pages 149–154, 2006.

[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. *Model abstraction of nondeterministic finite state automata in supervisor synthesis*. IEEE Trans. Automatic Control (conditionally acccepted), September, 2009. It also appears in SE Technical Report No. 2008-3, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven University of Techonology, Eindhoven, The Netherlands, 2008. ISSN 1567-1872. URL: http://se.wtb.tue.nl/sereports.

[11] R. Su, J.H. van Schuppen and J.E. Rooda. *Aggregative synthesis of distributed supervisors based on automaton abstraction*. IEEE Trans. Automatic Control (accepted), October, 2009. It also appears in SE Technical Report No. 2009-1, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven University of Techonology, Eindhoven, The Netherlands, 2009. ISSN 1567-1872. URL: http://se.wtb.tue.nl/sereports.

[12] R. Su, J.H. van Schuppen, J.E. Rooda and A.T. Hofkamp. *Efficient nonconflict check by using automaton abstractions*. In *Proc. 10th European Control Conference*, pages 1997-2002, 2009. Its extended version also appears in SE Technical Report No. 2008-10, Systems Engineering Group, Department of Mechanical Engineering, Eindhoven University of Techonology, Eindhoven, The Netherlands, 2008. ISSN 1567-1872. URL: http://se.wtb.tue.nl/sereports.

[13] H. Flordal, R. Malik, M. Fabian and K. Akesson. Compositional synthesis of maximally permissive supervisors using supervisor equivalence. In *Discrete Event Dynamic Systems*, 17(4):475-504, 2007.

[14] S.H. Lee and K.C. Wong. Structural decentralized control of concurrent discrete-event systems. *European Journal of Control*, 8(5):477491, 2002.

[15] R. Malik and H. Flordal. Yet another approach to compositional synthesis of discrete event systems. In *Proc. 9th International Workshop on Discrete Event Systems (WODES08)*, pages 16–21, 2008.

[16] 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, 2006.

[17] 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.

[18] R. Hill and D. Tilbury. Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction. In *Proc. 8th International Workshop on Discrete Event Systems (WODES06)*, pages 399–406, 2006.

[19] H. Flordal and R. Malik. Modular nonblocking verification using conflict equivalence. In *Proc. 8th International Workshop on Discrete Event Systems (WODES06)*, pages 100–106, 2006.

[20] R. Su and W.M. Wonham. Supervisor reduction for discrete-event systems. *Discrete Event Dynamic Systems*, 14(1):31-53, 2004

[21] J. Yi, S. Ding, M.T. Zhang, M.T. and P. van der Meulen. Throughput analysis of linear cluster tools. In *proc. 3rd IEEE International Conference on Automation Science and Engineering (CASE2007)*, pages 1063-1068, 2007.

[22] K. Akesson, H. Flordal and M. Fabian. Exploiting modularity for synthesis and verification of supervisors. In *proc. 15th IFAC world congress*, 2007.

[23] J. Komenda, J. van Schuppen, B. Gaudin and H. Marchand. Modular supervisory control with general indecomposable specification languages. In *proc. 44th IEEE Conference on Decision and Control (CDC05)*, pages 3474-3479, 2005.

[24] M.H. de Queirozm and J.E.R. Cury. Modular supervisory control of large scale discrete event systems. In *Discrete event systems - analysis and control (edited by R. Boel and G. Stremersch)*, pp.103-110, Kluwer. Boston, 2000.

[25] K. Rohloff and S. lafortune. The control and verification of similar agents operating in a broadcast network environment. In *proc. 42nd IEEE Conference on Decision and Control (CDC03)*, pages 2673-2679, 2003.

[26] Y. Willner and M. Heymann. Supervisory control of concurrent discrete-event systems. *International Journal of Control*, 54(5):1143-1169, 1991.