Markov Clustering based Fully Automated Nonblocking Hierarchical Supervisory Control of Large-Scale Discrete-Event Systems ⋆

Yingying Liu [a], Zhaojian Cai [b],Kai Cai [b]

[a] *School of Mechanical and Electronic Engineering, Northwest A&F University, Xianyang, China*

[b] *Department of Core Informatics, Osaka Metropolitan University, Osaka, Japan*

Abstract

State explosion problem is a major challenge for supervisory control synthesis of large-scale discrete-event systems (DES). In this paper, we address this challenge by proposing a fully automated nonblocking heterarchical supervisory control scheme. We integrate a clustering method and an abstraction-based approach into a streamlined procedure to automatically perform horizontal decomposition and vertical aggregation of the overall system. Horizontal decomposition partitions the plant components and specifications into subsystems, while vertical aggregation renders the plant to be controlled by a hierarchy of decentralized supervisors and coordinators. We first synthesize a decentralized supervisor for each imposed control specification. Then we develop a method based on the Markov Clustering Algorithm to cluster the decentralized supervisors into modular subsystems and design coordinators to remove the blocking states (whenever they exist) in each subsystem. Moreover we employ an abstraction-based approach to create an abstraction for each subsystem, cluster these abstractions into higher level subsystems, and correspondingly synthesize higher level coordinators to remove any blocking states. This process proceeds until there remains a single higher level cluster, for which a top-level coordinator is synthesized to ensure nonblockingness of the global controlled behavior. Such a heterarchical synthesis approach based on multi-layer clustering and abstraction does not require or needs to rely on engineering insight of the system structure. It is shown that our obtained decentralized supervisors and hierarchical coordinators jointly achieve global optimality and nonblockingness. Our proposed synthesis scheme is demonstrated by two benchmark case studies.

Key words: Supervisory control; Decentralized and hierarchical supervisory control; Clustering; Abstraction-based approach; Large-scale discrete-event systems.

1 Introduction

The supervisory control theory (SCT) [25,26] plays a fundamental role in controlling large-scale DES. Examples include automated guided vehicles in logistic systems, multi-robot in manufacturing cells, and wireless sensor networks in communications system. Despite the great practical significance, one of the major drawbacks of supervisory control synthesis for large-scale DES is state-space explosion [5,10–12].

To reduce such computational difficulty, a variety of modular control architectures have been proposed. One is concerned with controlling the system by a hierarchy of decentralized supervisors and coordinators, called heterarchical architecture [2,25]. Heterarchical synthesis is given by combining horizontal decomposition and vertical clustering, where horizontal decomposition partitions the plant components and specifications into subsystems, and vertical clustering represents that the plant is controlled by a hierarchy of decentralized supervisors and coordinators to ensure overall nonblocking [8,14,16,18] . The central technique of heterarchical synthesis is *model abstraction*, with special conditions imposed on natural projections ensuring that the heterarchical controlled behavior is identical to the monolithic controlled behavior. There are several model abstraction techniques that are provided to synthesize supervisors and coordinators by abstracted models, where the coordinators achieve

⋆ This paper was not presented at any IFAC meeting. Corresponding author Kai Cai.

Email addresses: yingyingliu611@163.com (Yingying Liu), cai@omu.ac.jp (Kai Cai).

the overall nonblocking [17,20,27]. Based on the model abstraction techniques, the conditions that make the controlled behavior to be maximal permissive is introduced [7,18,24]. However, the decomposition method in abstraction-based approaches is non-automatic. That is to say, an automatic synthesis process for the decomposition is lacking.

To address the above issue, in this paper we extend the abstraction-based approach into a streamlined algorithm to automatically synthesize heterarchical supervisor control while guaranteeing global nonblocking as follows.We first synthesize decentralized supervisors to enforce each specification. Then by employing a markov clustering (MCL) algorithm [4] the synthesized decentralized supervisors are algorithmically clustered. To this end, we use dependency structure matrix (DSM) [9] to encode the relationship between each plant component and specification. We then check if there exist conflicts in each cluster. If so, we design a supervisor, called coordinator, to remove the conflicts. After ensuring each cluster is nonblocking, we employ the model abstraction technique to compute an abstract model for each nonblocking cluster. The clustering, conflicts removing, and model abstraction steps are repeated until the number of clusters is no more than two. Finally, it is checked if conflicts exist among the abstractions in the top-level cluster(s). If so, a coordinator is synthesized to ensure the top-level nonblockingness. A set of decentralized supervisors and hierarchical coordinators are generated by above procedure, which is shown to collectively guarantee the overall nonblocking of the entire system.

We note that in [10] a systematic approach is proposed to transform a set of plant components and a set of specifications modeled as extended finite automata into a tree-structured multilevel DES to which multilevel supervisory control synthesis [15] can be applied. The authors in [10] cluster plant components into a tree structure, so the DSM in [10] encodes the number of specifications that shares events with plant components. The output of MCL algorithm thus is the plant component clusters. In contrast, we record the number of plant components that share events with specifications because we synthesize a decentralized supervisor for each specification before clustering, and find clusters for these decentralized supervisors. Moreover, we conduct model abstraction in vertical aggregation to ensure the global nonblockingness, which is not guaranteed in [10].

To address nonblockingness the authors in [21] present an approach to synthesize a deterministic coordinated distributed supervisor under partial observation and provide a sufficient condition to ensure the maximal permissiveness of a coordinated distributed supervisor generated by the proposed synthesis approach. A collection of deterministic local nonblocking state-normal supervisors is computed such that the global requirement satisfaction and nonblockingness can be achieved. The authors in [21] give a procedure, called Sequential Abstraction over Product, to obtain an abstraction of the plant in a sequential way, which avoids computing the plant explicitly that may be prohibitively large for systems of industrial size. In contrast, we extend the abstraction-based approach into a streamlined algorithm to achieve the global nonblocking without losing the information of the plant and reduce the computational cost.

The authors in [27] integrate supervisory control theory and a model-based deep reinforcement learning method to synthesize a nonblocking coordinator for the modular supervisors. The deep reinforcement learning method significantly reduces the computational complexity by avoiding the computation of synchronization of multiple modular supervisors and the plant models. The supervisory control function in [27] is represented by the deep neural network instead of a large-scale automaton or a state-based lookup table. In contrast, we avoid to compute the synchronous product of multiple local modular supervisors and plant models by heterarchical architecture.

In summary the contribution of this paper is threefold. First, compare to the abstraction-based approach, system decomposition in our method based on MCL is automatic. Second, compare to the existing techniques of clustering, model abstraction in our method guarantees global nonblocking. Finally, we propose a streamlined algorithmic solution for the nonblocking heterarchical supervisory control problem, which does not require engineering insight of the system structure. The obtained results, in particular the state size of the coordinators for global nonblocking, are comparable with the existing approaches that need detailed analysis of the system structure.

The remainder of the paper is organized as follows. Section 2 introduces preliminaries. Algorithmic approach for heterarchical supervisor synthesis and the theoretical results are provided in Section 3. In Section 4, the proposed method is demonstrated with two examples. We then draw conclusions in the last section, Section 5.

2 PRELIMINARIES

The DES plant to be controlled is modeled by a generator $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is the finite state set, $\Sigma = \Sigma_c \dot{\cup} \Sigma_u$ is the finite event set which is partitioned into two subsets – the controllable event subset Σ_c and the uncontrollable event subset Σ_u. The event set Σ also can be partitioned into a set of observable events Σ_o and a set of unobservable events Σ_{uo} due to partial observation of a system, i.e., $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$. The natural projection is defined as $P : \Sigma \to \Sigma_o$ and can be extended in a usual way

$P : \Sigma^* \to \Sigma_o^*$. The inverse projection of P is defined as $P^{-1} : \Sigma_o^* \to 2^{\Sigma^*}$ with $P^{-1}(\omega) = \{\lambda \in \Sigma^* | P(\lambda) = \omega\}$, where $\omega \in \Sigma_o^*$. $\delta : Q \times \Sigma \to Q$ is the (partial) state transition function, $q_0 \in Q$ is the initial state, and $Q_m \subseteq Q$ is the set of marker states. In the usual way, we extend δ such that $\delta : Q \times \Sigma^* \to Q$, and write $\delta(q,s)!$ to mean that $\delta(q,s)$ is defined, where $q \in Q$ and $s \in \Sigma^*$. A string $s_1 \in \Sigma^*$ is a *prefix* of another string $s \in \Sigma^*$, written $s_1 \leq s$, if there exists $s_2 \in \Sigma^*$ such that $s_1 s_2 = s$. The *prefix closure* of L, written $\overline{L}$, is $\overline{L} := \{s_1 \in \Sigma^* \mid (\exists s \in L) s_1 \leq s\}$. A language L is *closed* if $L = \overline{L}$.

The closed behavior of $\mathbf{G}$ is the set of all strings that can be generated by $\mathbf{G}$: $L(\mathbf{G}) := \{s \in \Sigma^* | \delta(q_0, s)!\}$. As defined $L(\mathbf{G})$ is closed. On the other hand, the marked behavior of $\mathbf{G}$ is the subset of strings that can reach a marker state: $L_m(\mathbf{G}) := \{s \in L(\mathbf{G}) | \delta(q_0, s) \in Q_m\} \subseteq L(\mathbf{G})$. As in the case of an automaton, a state $q \in Q$ is *reachable* if $(\exists s \in \Sigma^*)\delta(q_0, s)! \,\&\, \delta(q_0, s) = q$; $\mathbf{G}$ itself is *reachable* if q is reachable for all $q \in Q$. A state $q \in Q$ is *coreachable* if $(\exists s \in \Sigma^*)\delta(q, s) \in Q_m$; $\mathbf{G}$ itself is *coreachable* if q is coreachable for all $q \in Q$. $\mathbf{G}$ is *nonblocking* if every reachable state is coreachable, or equivalently

$$L(\mathbf{G}) = \overline{L_m(\mathbf{G})},$$

namely every string in the closed behavior may be completed to a string in the marked behavior. $\mathbf{G}$ is *trim* if it is both reachable and coreachable. A language L is controllable with respect to $\mathbf{G}$ if $\overline{L}\Sigma_u \cap L(\mathbf{G}) \subseteq \overline{L}$. We employ $C(L)$ to denote the family of all controllable sublanguages of L, which contains a (unique) supremal element $\sup C(L) := \cup\{L' | L' \in C(L)\}$ [25,26].

Let $\mathbf{G}$ be the plant to be controlled, consisting of N (> 1) component agents $\mathbf{G}_k = (Q_k, \Sigma_k, \delta_k, q_{0,k}, Q_{m,k})$, $k = 1, 2, \ldots, N$. Write $\underline{N}$ for the set of integers $\{1, 2, \ldots, N\}$. Then the closed and marked behaviors of $\mathbf{G}$ are $L(\mathbf{G}) = \|\{L(\mathbf{G}_k) | k \in \underline{N}\}$ and $L_m(\mathbf{G}) = \|\{L_m(\mathbf{G}_k) | k \in \underline{N}\}$, respectively, where $\|$ denotes *synchronous product* ([26]). Each agent's event set Σ_k is partitioned into two subsets – a controllable subset $\Sigma_{c,k}$ and an uncontrollable subset $\Sigma_{u,k}$, i.e., $\Sigma_k = \Sigma_{c,k} \dot{\cup} \Sigma_{u,k}$. Hence the plant $\mathbf{G}$ is defined over $\Sigma := \cup\{\Sigma_k | k \in \underline{N}\}$, with controllable subset $\Sigma_c := \dot{\cup}\{\Sigma_{c,k} | k \in \underline{N}\}$ and uncontrollable subset $\Sigma_u := \dot{\cup}\{\Sigma_{u,k} | k \in \underline{N}\}$.

In this paper, we consider the plant $\mathbf{G}$ consisting of N component agents $\mathbf{G}_k = (Q_k, \Sigma_k, \delta_k, q_{0,k}, Q_{m,k})$, $k = 1, ..., N$. The behavioral constraint is imposed through (local) specification languages $E_i \subseteq \Sigma_{E_i}^* (i \in I)$ where $\Sigma_{E_i} \subseteq \Sigma$ is the alphabet of E_i and the I is an index set. For each specification E_i, the decentralized plant $\mathbf{G}_{E_i}$ is synthesized from the agents that share events with Σ_{E_i}:

$$\mathbf{G}_{E_i} := ||\{\mathbf{G}_k | \Sigma_k \cap \Sigma_{E_i} \neq \emptyset\} \tag{1}$$

Then we synthesize an optimal and nonblocking *decentralized supervisor* $\mathbf{SUP}_i$:

$$L_m(\mathbf{SUP}_i) = \sup C(L_m(\mathbf{E}_i) || L_m(\mathbf{G}_{E_i})), \tag{2}$$

In general, the joint behavior of the decentralized supervisors fails to be nonblocking, i.e. $||_{i \in I} L(\mathbf{SUP}_i) \supsetneq \overline{||_{i \in I} L_m(\mathbf{SUP}_i)}$. An additional supervisor $\mathbf{CO}$, called the *coordinator*, needs to be designed to resolve conflicts among supervisors:

$$L_m(\mathbf{CO}) := \sup C(||_{i \in I} L_m(\mathbf{SUP}_i)) || \Sigma^*) \tag{3}$$

$$||_{i \in I} L(\mathbf{SUP}_i) || L(\mathbf{CO}) = \overline{||_{i \in I} L_m(\mathbf{SUP}_i) || L_m(\mathbf{CO})} \tag{4}$$

The joint behavior of the supervisors and the coordinator is identical to the monolithic controlled behavior [25,26].

However, checking conflicts and designing the coordinator as above may not be feasibly computable since the synchronous product of all $\mathbf{SUP}_i$ is required. We are motivated to improve the computation efficiency by developing a streamlined algorithmic solution for the nonblocking heterarchical supervisory control problem.

3 Algorithmic Approach for Heterarchical Supervisor Synthesis

In this section we propose a streamlined algorithm to conduct horizontal decomposition and vertical aggregation for large-scale DES, and achieve global nonblocking. We first synthesize decentralized supervisors to enforce each imposed control specification. Then we propose a method based on the Markov Cluster Algorithm [22] to automatically perform system decomposition without requiring engineering insight. Finally, we ensure the nonblockingness of global controlled behavior by computing a coordinator that resolves conflicts among decentralized supervisors at the abstraction level. Before presenting our architectural approach to heterarchical supervisor synthesis, we recall some essential previous results and definitions necessary for the synthesis procedure.

Firstly, we recall Markov Clustering Algorithm (MC Algorithm) [22], which is a graph clustering method that identifies clusters by generating Random Walk on the graph. The input of MC algorithm described in [6] is a stochastic matrix $\mathbf{M}$ wherein the column elements sum up to one. An entry $\mathbf{M}_{ij}$ in $\mathbf{M}$ represents the probability of a random walk from column element i to row element j. In this paper, we apply the MC Algorithm to automatically partition the decentralized supervisors given by (2) into clusters without requiring engineering insight. The input of MC Algorithm $\mathbf{M}$ is determined through the following three steps, which are originally from [8].

(S1) Consider a set of plant models and a set of specification models. First generate a binary matrix $\mathbf{P}$ called Domain Mapping Matrix (DMM) [4] to record whether a plant and a specification have shared events. $\mathbf{P}(i,j) = 1$ if specification i and plant j have shared events; otherwise $\mathbf{P}(i,j) = 0$.

(S2) Multiply $\mathbf{P}$ with its tranpose matrix $\mathbf{P}^\top$ to get a square matrix, called *Dependency Structure Matrix* (DSM)[8,19],

$$\mathbf{P_D} = \mathbf{P} \times \mathbf{P}^\top. \tag{5}$$

$\mathbf{P_D}(i,j) = k$ means that there exist k plant models that have shared events with both specification models $\mathbf{E}_i$ and $\mathbf{E}_j$.

(S3) Convert $\mathbf{P_D}$ into stochastic matrix $\mathbf{M}$ by normalizing the columns of $\mathbf{P_D}$:

$$\mathbf{M}_{ij} = \frac{\mathbf{P_D}_{ij}}{\sum\limits_i \mathbf{P_D}_{ij}}. \tag{6}$$

The MC Algorithm is an iterative algorithm that consists of two steps, namely *expansion* and *inflation*. The *expansion* is the powers of matrix $\mathbf{M}$ meaning the random walker does α jumps in each iteration k, i.e.,

$$\mathbf{M}_{k+1} = \mathbf{M}_k^\alpha. \tag{7}$$

Typically, a value of $\alpha = 2$ is used for the stability of the algorithm [23]. However, other values of α can also be tried. When $\alpha = 1$, the clustering may diverge, while when $\alpha = 3$, the clustering may have less influence on the results. The $\mathbf{M}_0$ is the input of MC Algorithm matrix $\mathbf{M}$. The *inflation* aims to strengthen the probability $\mathbf{M}_{ij}$ in $\mathbf{M}$ after each expansion step. It is updated by

$$(\mathbf{M}_{k+1})_{ij} := \frac{(\mathbf{M}_{k+1})_{ij}^\beta}{\sum\limits_{i=1}^{N} (\mathbf{M}_{k+1})_{ij}^\beta} \tag{8}$$

After the inflation step, the transition probabilities for high values are increased while those for low values are decreased. By repeating the *expansion* and *inflation* steps given above, the matrix $\mathbf{M}$ will eventually converge to a stable matrix.

Next, we recall two definitions that will be employed to construct models for verifying the nonblockingness of the system.

Definition 1 *A natural projections P_i is said to be a natural observer for a language $L \subseteq \Sigma^*$ if*

$$(\forall s \in \overline{L},\ \forall t_o \in \Sigma_o^*)\ P(s)t_o \in P(L) \tag{9}$$

$$\Rightarrow (\exists t \in \Sigma^*)\ P(t) = t_o\ \&\ st \in L$$

In plain words, P is a *natural observer* for L if that *P(s)* can be extended to *P(L)* by an observable string t_o implies that s can be extended to L by a string t with $P(t) = t_o$.

Definition 2 *A language L is said to be Output Control Consistency (OCC) [28] if for every string $s = s'\sigma_1 \cdots \sigma_k, k \geq 1 \in L$, where s' is either the empty string ε or terminates with an event in Σ_o, the following holds*

$$((\forall i \in \{1, k-1\})\sigma_i \in \Sigma \setminus \Sigma_o)\ \&\ \sigma_k \in \Sigma_o \cap \Sigma_u$$

$$\Rightarrow ((\forall j \in \{1,k\})\sigma_j \in \Sigma_u) \tag{10}$$

This property shows that if the prefix of a string in L terminates with an observable event or is an empty string, and this string terminates with an observable and uncontrollable event, then all immediately preceding unobservable events of the prefix must be uncontrollable.

3.1 Algorithmic Approach for Heterarchical Supervisor Synthesis

Now we are ready to consider the global nonblocking problem of a large-scale DES based on the heterarchical architecture. We first synthesize decentralized supervisors for each specification. Then by employing a MC algorithm [4] the synthesized decentralized supervisors are clustered. We check if there are conflicts in each cluster and, if so, design a supervisor called a coordinator to remove them. After ensuring that each cluster is nonblocking, we employ the model abstraction approach to compute an abstract model for each nonblocking cluster. The clustering, conflicts removing, and model abstraction steps are repeated until there are no more than two clusters. Finally, it is checked if conflicts exist among the abstractions in the top-level cluster(s). If so, a coordinator is synthesized to ensure the nonblockingness. This procedure generates a set of decentralized supervisors and hierarchical coordinators that collectively guarantee the overall nonblockingness of the entire system.

In this paper, we consider the system modeled as J component agents $\mathbf{G}_1,\ldots,\mathbf{G}_J$ and I specification automata $\mathbf{E}_1,\ldots,\mathbf{E}_I (J,I \in \mathbb{N}^+)$. Our proposed approach to the heterarchical supervisory control includes the following six steps.

Step 1 Synthesize optimal and nonblocking *decentralized supervisors* $\mathbf{SUP}_i$ for each $\mathbf{E}_i$ by (1) and (2), i.e.,

$$L_m(\mathbf{SUP}_i) = \sup C(L_m(\mathbf{E}_i)||L_m(\mathbf{G}_{\mathbf{E}_i})).$$

Step 2 Compute the stochastic square matrix $\mathbf{M}$ by Steps (**S1**)-(**S3**) given above and set the *expansion* parameter α to 2 and adjust the *inflation* parameter β obtain varying clustering outcomes. Employ the MC algorithm to partition the decentralized supervisors into N clusters $\{1,\ldots,N\}$, thereby designing modular subsystems.

Step 3 Let $\mathscr{X}_l$ $(l \in \{1,\ldots,N\})$ be a set of indices of the decentralized supervisors in cluster l. The subsystem $\mathbf{SUB}_l$ is the synchronous product of all decentralized supervisors in the same cluster, i.e.

$$\mathbf{SUB}_l = ||\{\mathbf{SUP}_i | i \in \mathscr{X}_l\} \tag{11}$$

Check if there exists blocking in the subsystem $\mathbf{SUB}_l$. If so, a coordinator $\mathbf{CO}_l$ will be designed to remove the blocking states by (3). Then the nonblocking subsystem is obtained by

$$\mathbf{SUB}_l = ||_{i\in\mathscr{X}_l}(\mathbf{SUP}_i)||\mathbf{CO}_l.$$

Step 4 Synthesize an *abstraction* $\mathbf{ABS}_l$ for the subsystem $\mathbf{SUB}_l$ by

$$\mathbf{ABS}_l = P_l(\mathbf{SUB}_l),$$

where $P_l : \Sigma^*_{\mathbf{SUB}_l} \to \Sigma^*_{\mathbf{ABS}_l}$ is a natural projection from the event set of $\mathbf{SUB}_l$, written as $\Sigma_{\mathbf{SUB}_l}$, to the event set of $\mathbf{ABS}_l$: $\Sigma_{\mathbf{ABS}_l} := \Sigma_o \cap \Sigma_{\mathbf{SUB}_l}$ with a shared event set $\Sigma_o := \{\sigma \in \Sigma | (\exists n_1, n_2 \in \{1,\ldots,N\})\ n_1 \neq n_2\ \&\ \sigma \in \Sigma_{\mathbf{SUB}_{n_1}} \cap \Sigma_{\mathbf{SUB}_{n_2}}\}$. The event set $\Sigma_{\mathbf{ABS}_n}$ is extended to make P_n satisfy both natural observer and OCC properties. Abstractions $\mathbf{ABS}_1,...,\mathbf{ABS}_N$ thus are obtained.

Step 5 Repeat the **Step 2** to **Step 4** until there are no more than two clusters.

Step 6 Check if there are any conflicts among the abstractions in the top-level cluster(s). If such conflicts exist, generate a coordinator to guarantee nonblockingness.

Note that if the result of clustering remains the same as before, i.e., if the number of clusters has not decreased compared to the previous clustering, we will reduce the inflation value β. Additionally, after the first round of clustering, we partition the set of abstractions (rather than the set of decentralized supervisors as in the first time). Hence the binary matrix $\mathbf{P}$ in **Step 2** is constructed by $\mathbf{P}(l,j) = 1$ if the event set $\Sigma_{\mathbf{ABS}_l}$ of abstraction $\mathbf{ABS}_l$ and the event set $\Sigma_{\mathbf{G}_j}$ of plant component $\mathbf{G}_j$ are not disjoint; otherwise $\mathbf{P}(l,j) = 0$.

3.2 Theoretical results

By the above procedure, we obtain I decentralized supervisors $\mathbf{SUP}_1,...,\mathbf{SUP}_I$ and H coordinators $\mathbf{CO}_1,\ldots,\mathbf{CO}_H$ (the number of coordinators is case-dependent). We conclude that the synchronized behavior of these supervisors and coordinators is identical to the behavior of the monolithic supervisor $\mathbf{SUP}$ and the global nonblocking is ensured by the obtained supervisors and coordinators.

Theorem 1 *Suppose that the proposed procedure terminates after $L(\geq 1)$ levels of clustering/abstraction, and in each level $l \in \{1,\ldots,L\}$ there are N_l clusters $\mathbf{SUB}_{n_l}$ ($n_l \in \{1,\ldots,N_l\}$). Assume that the procedure generates I decentralized supervisors $\mathbf{SUP}_i$ ($i \in \{1,\ldots,I\}$) and H coordinators $\mathbf{CO}_h$ ($h \in \{1,\ldots,H\}$). If for every $l \in \{1,\ldots,L\}, n_l \in \{1,\ldots,N_l\}$, P_{n_l} is an $L_m(\mathbf{SUB}_{n_l})$-observer and OCC for $L_m(\mathbf{SUB}_{n_l})$, then*

$$(||_{i\in\{1,\ldots,I\}}L(\boldsymbol{SUP}_i))||(||_{h\in\{1,\ldots,H\}}L(\boldsymbol{CO}_h))||L(\boldsymbol{G}) = L(\boldsymbol{SUP})$$
$$(||_{i\in\{1,\ldots,I\}}L_m(\boldsymbol{SUP}_i))||(||_{h\in\{1,\ldots,H\}}L_m(\boldsymbol{CO}_h))||L_m(\boldsymbol{G}) = L_m(\boldsymbol{SUP})$$

To prove Theorem 1, we need the following proposition from [7].

Proposition 1 ([7]) *Consider I decentralized supervisors $\boldsymbol{SUP}_i$ ($i \in \{1,\ldots,I\}$) and H coordinators $\mathbf{CO}_h$ ($h \in \{1,\ldots,H\}$) based on abstractions. If for every i, P_i is an $L_m(\boldsymbol{SUP}_i)$-observer and OCC for $L_m(\boldsymbol{SUP}_i)$, then*

$$\overline{(||_{i\in\{1,\ldots,I\}}L(\boldsymbol{SUP}_i))||(||_{h\in\{1,\ldots,H\}}L(\boldsymbol{CO}_h))} =$$
$$\overline{(||_{i\in\{1,\ldots,I\}}L_m(\boldsymbol{SUP}_i))||(||_{h\in\{1,\ldots,H\}}L_m(\boldsymbol{CO}_h))}$$
$$(||_{i\in\{1,\ldots,I\}}L(\boldsymbol{SUP}_i))||(||_{h\in\{1,\ldots,H\}}L(\boldsymbol{CO}_h))||L(\boldsymbol{G}) = L(\boldsymbol{SUP})$$
$$(||_{i\in\{1,\ldots,I\}}L_m(\boldsymbol{SUP}_i))||(||_{h\in\{1,\ldots,H\}}L_m(\boldsymbol{CO}_h))||L_m(\boldsymbol{G}) = L_m(\boldsymbol{SUP})$$

Proof of Theorem 1. Consider a system with I specification models.

Level 1: By step 1, we get I decentralized supervisors $\mathbf{SUP}_1,\ldots,\mathbf{SUP}_I$ which are partitioned into N_1 clusters by step 2. Suppose there are $\mathscr{X}_{n_1}$ ($n_1 \in \{1,\ldots,N_1\}$) decentralized supervisors in cluster n_1. A subsystem $\mathbf{SUB}_{n_1}$ ($n_1 \in \{1,\ldots,N_1\}$) is synthesized for cluster n_1 by $\mathbf{SUB}_{n_1} = ||\{\mathbf{SUP}_i | i \in \mathscr{X}_{n_1}\}$. The subsystem is updated by $\mathbf{SUB}_{n_1} = (||_{i\in\mathscr{X}_{n_1}}(\mathbf{SUP}_i))||\mathbf{CO}_{n_1}$ to remove the blocking states in $||\{\mathbf{SUP}_i | i \in \mathscr{X}_{n_1}\}$. At this level, each abstraction $\mathbf{ABS}_{n_1}$ ($n_1 \in \{1,\ldots,N_1\}$) is obtained by projecting $\mathbf{SUB}_{n_1}$ as described in step 4, i.e.,

$$L(\mathbf{ABS}_{n_1}) = P_{n_1}L(\mathbf{SUB}_{n_1}), L_m(\mathbf{ABS}_{n_1}) = P_{n_1}L_m(\mathbf{SUB}_{n_1}).$$

Since the projection P_{n_1} in step 4 is an $L_m(\mathbf{SUB}_{n_1})$-observer and OCC for $L_m(\mathbf{SUB}_{n_1})$, by Proposition 1 we have

$$\overline{(||_{i\in\{1,\ldots,I\}}L(\mathbf{SUP}_i))||_{n_1\in\{1,\ldots,l_1\leq N_1\}}L(\mathbf{CO}_{n_1})} =$$
$$\overline{(||_{i\in\{1,\ldots,I\}}L_m(\mathbf{SUP}_i))||_{n_1\in\{1,\ldots,l_1\}}L_m(\mathbf{CO}_{n_1})}$$
$$(||_{i\in\{1,\ldots,I\}}L(\mathbf{SUP}_i))||_{n_1\in\{1,\ldots,l_1\}}L(\mathbf{CO}_{n_1})||L(\mathbf{G}) = L(\mathbf{SUP})$$
$$(||_{i\in\{1,\ldots,I\}}L_m(\mathbf{SUP}_i))||_{n_1\in\{1,\ldots,l_1\}}L_m(\mathbf{CO}_{n_1})||L_m(\mathbf{G}) = L_m(\mathbf{SUP})$$

Level 2: When the number of clusters is greater than two, we do further clustering for abstractions $\mathbf{ABS}_1,\ldots,\mathbf{ABS}_{N_1}$. Suppose that we get N_2 clusters at the second level and there are $\mathscr{X}_{n_2}$ ($n_2 \in \{1,\ldots,N_2\}$) abstractions in cluster n_2. For each cluster a corresponding subsystem $\mathbf{SUB}_{n_2} = ||\{\mathbf{ABS}_{n_1} | n_1 \in \mathscr{X}_{n_2}\}(n_2 \in \{1,\ldots,N_2\})$ is formed by step 3. Similarly, the subsystem is updated to remove the blocking states in $||\{\mathbf{ABS}_{n_1} | n_1 \in \mathscr{X}_{n_2}\}$ by

$$\mathbf{SUB}_{n_2} = (||_{n_1\in\mathscr{X}_{n_2}}(\mathbf{ABS}_{n_1}))||\mathbf{CO}_{n_2}.$$

Then, an abstraction $\mathbf{ABS}_{n_2}$ ($n_2 \in \{1,\ldots,N_2\}$) as in step 4 is synthesized by natural projection P_{n_2} for $\mathbf{SUB}_{n_2}$, i.e.,

$$L(\mathbf{ABS}_{n_2}) = P_{n_2}L(\mathbf{SUB}_{n_2}),\ L_m(\mathbf{ABS}_{n_2}) = P_{n_2}L_m(\mathbf{SUB}_{n_2})$$

By Proposition 1, the projection P_{n_2} in step 4 is an $L_m(\mathbf{SUB}_{n_2})$-observer and OCC for $L_m(\mathbf{SUB}_{n_2})$, so we obtain that

$$\begin{array}{l}(||_{i\in\{1,\ldots,I\}}L(\mathbf{SUP}_i))||_{n_1\in\{1,\ldots,l_1\leq N_1\}}L(\mathbf{CO}_{n_1})||_{n_2\in\{1,\ldots,l_2\leq N_2\}}L(\mathbf{CO}_{n_2}) = \\ \overline{(||_{i\in\{1,\ldots,I\}}L_m(\mathbf{SUP}_i))||_{n_1\in\{1,\ldots,l_1\}}L_m(\mathbf{CO}_{n_1})||_{n_2\in\{1,\ldots,l_2\leq N_2\}}L_m(\mathbf{CO}_{n_2})} \\ (||_{i\in\{1,\ldots,I\}}L(\mathbf{SUP}_i))||_{n_1\in\{1,\ldots,l_1\}}L(\mathbf{CO}_{n_1})||_{n_2\in\{1,\ldots,l_2\leq N_2\}}L(\mathbf{CO}_{n_2}) \\ ||L(\mathbf{G}) = L(\mathbf{SUP})(||_{i\in\{1,\ldots,I\}}L_m(\mathbf{SUP}_i))||_{n_1\in\{1,\ldots,l_1\}}L_m(\mathbf{CO}_{n_1}) \\ ||_{n_2\in\{1,\ldots,l_2\leq N_2\}}L_m(\mathbf{CO}_{n_2})||L_m(\mathbf{G}) = L_m(\mathbf{SUP})\end{array}$$

Level L: Consider that the procedure ends with clustering $L(\geq 1)$ times. At the top level L, suppose there are $\mathscr{X}_{n_L}$ $(n_L \in \{1,\ldots,N_L\})$ abstractions in cluster n_L. Here $N_L \leq 2$ by step 5. For each cluster a subsystem $\mathbf{SUB}_{n_L}$ $(n_L \in \{1,\ldots,N_L\})$ is formed by $\mathbf{SUB}_{n_L} = ||\{\mathbf{ABS}_{n_{(L-1)}} | n_{(L-1)} \in \mathscr{X}_{n_L}\}$ and updated by

$$\mathbf{SUB}_{n_L} = (||_{n_{(L-1)}\in\mathscr{X}_{n_L}}(\mathbf{ABS}_{n_{(L-1)}}))||\mathbf{CO}_{n_L}$$

to remove the blocking states in step 3. Then, an abstraction $\mathbf{ABS}_L$ $(n_L \in \{1,\ldots,N_L\})$ is synthesized by natural projection P_{n_L} for $\mathbf{SUB}_{n_L}$, i.e.,

$$\begin{array}{r}L(\mathbf{ABS}_{n_{(L)}}) = P_{n_{(L)}}L(\mathbf{SUB}_{n_{(L)}}) \\ L_m(\mathbf{ABS}_{n_{(L)}}) = P_{n_{(L)}}L_m(\mathbf{SUB}_{n_{(L)}}).\end{array}$$

Since for every $l \in \{1,\ldots,L\}, n_l \in \{1,\ldots,N_l\}$, P_{n_l} is an $L_m(\mathbf{SUB}_{n_l})$-observer and OCC for $L_m(\mathbf{SUB}_{n_l})$, by proposition 1 we obtain that

$$\begin{array}{l}(||_{i\in\{1,\ldots,I\}}L(\mathbf{SUP}_i))||_{n_1\in\{1,\ldots,l_1\leq N_1\}}L(\mathbf{CO}_{n_1})\ldots||_{n_L\in\{1,\ldots,l_L\leq N_L\}}L(\mathbf{CO}_{n_L}) = \\ \overline{(||_{i\in\{1,\ldots,I\}}L_m(\mathbf{SUP}_i))||_{n_1\in\{1,\ldots,l_1\}}L_m(\mathbf{CO}_{n_1})\ldots||_{n_L\in\{1,\ldots,l_L\leq N_L\}}L_m(\mathbf{CO}_{n_L})} \\ (||_{i\in\{1,\ldots,I\}}L(\mathbf{SUP}_i))||_{n_1\in\{1,\ldots,l_1\}}L(\mathbf{CO}_{n_1})\ldots||_{n_L\in\{1,\ldots,l_L\leq N_L\}}L(\mathbf{CO}_{n_L}) \\ ||L(\mathbf{G}) = L(\mathbf{SUP})(||_{i\in\{1,\ldots,I\}}L_m(\mathbf{SUP}_i))||_{n_1\in\{1,\ldots,l_1\}}L_m(\mathbf{CO}_{n_1})\ldots \\ ||_{n_L\in\{1,\ldots,l_L\leq N_L\}}L_m(\mathbf{CO}_{n_L})||L_m(\mathbf{G}) = L_m(\mathbf{SUP})\end{array}$$

which implies that

$$\begin{array}{l}(||_{i\in\{1,\ldots,I\}}L(\mathbf{SUP}_i))||_{h\in\{n_1,\ldots,n_L\}}L(\mathbf{CO}_h) = \\ \overline{(||_{i\in\{1,\ldots,I\}}L_m(\mathbf{SUP}_i))||_{h\in\{n_1,\ldots,n_L\}}L_m(\mathbf{CO}_h)} \\ (||_{i\in\{1,\ldots,I\}}L(\mathbf{SUP}_i))||_{h\in\{n_1,\ldots,n_L\}}L(\mathbf{CO}_h)||L(\mathbf{G}) = L(\mathbf{SUP}) \\ (||_{i\in\{1,\ldots,I\}}L_m(\mathbf{SUP}_i))||_{h\in\{n_1,\ldots,n_L\}}L_m(\mathbf{CO}_h)||L_m(\mathbf{G}) = L_m(\mathbf{SUP})\end{array}$$

where $n_1 + n_2 + \cdots + n_L = H$. ◇

Theorem 1 asserts that decentralized supervisors and coordinators designed by our proposed method jointly achieve global optimal and nonblocking controlled behavior. Thereby, the computational effort of the supervisor synthesis for large-scale DES is reduced.

4 Case studies

In this section, the proposed method is demonstrated with two cases, where the first case is a control problem for automatic guided vehicles (AGVs)[13] and the second case is a large-scale system Production Cell [1]. For the AGV systems, the monolithic supervisor is computable, therefore it can be confirmed that the joint behavior of decentralized supervisors by our method is identical to the monolithic controlled behavior.

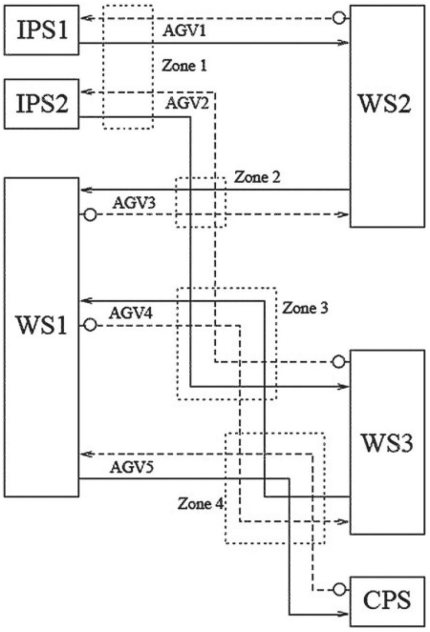

Fig. 1. AGVs system, taken from [3]

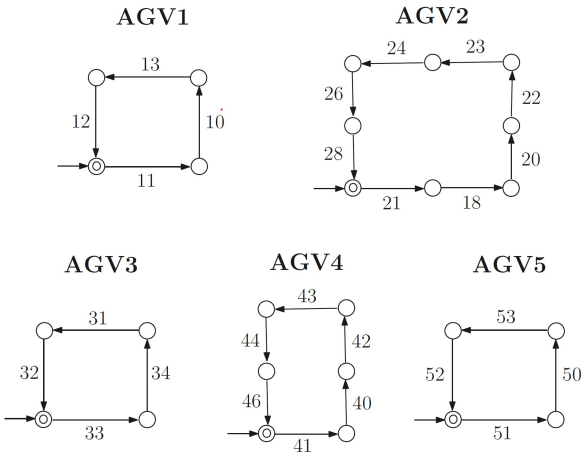

Fig. 2. Plant components of AGV, taken from [3]

4.1 Automatic Guided Vehicles

The AGV system consists of five automatic guided vehicles **AGV1**, ...,**AGV5** serving a manufacturing workcell described in [13] [3].

As displayed in Fig.1 the workcell consists of two input stations **IPS1, IPS2**; three workstations **WS1, WS2, WS3**; one completed parts station **CPS**; and four shared zones for the AGVs. The generator models of the AGVs are displayed in Fig.2, and Table 1 lists the interpretation of events.

Table 1
Interpretation of events

Event	**AGV**	Interpretation
11	1	Leaves **WS2** and enters Zone 1
10		Exits Zone 1 and loads from **IPS1**
13		Re-enters Zone 1
12		Exits Zone 1, unloads to **WS2** and parks
21	2	Leaves **WS3** and enters Zone 3
18		Exits Zone 3 and enters Zone 2
20		Exits Zone 2 and enters Zone 1
22		Exits Zone 1 and loads from **IPS2**
23		Re-enters Zone 1
24		Exits Zone 1 and re-enters Zone 2
26		Exits Zone 2 and re-enters Zone 3
28		Exits Zone 3, unloads to **WS3** and parks
33	3	Leaves **WS1** and enters Zone 2
34		Exits Zone 2 and loads from **WS2**
31		Re-enters Zone 2
32		Exits Zone 2, unloads to **WS1** and parks
41	4	Leaves **WS1** and enters Zone 3
40		Exits Zone 3 and enters Zone 4
42		Exits Zone 4 and loads from **WS3**
43		Re-enters Zone 4
44		Exits Zone 4 and re-enters Zone 3
46		Exits Zone 3, unloads at **WS1** and parks
51	5	Leaves **CPS** and enters Zone 4
50		Exits Zone 4 and loads from **WS1**
53		Re-enters Zone 4
52		Exits Zone 4, unloads to **CPS** and parks

For this system, the control specifications imposed on the plant are as follows:

- Each of the four shared zones should be occupied by at most one AGV at a time.
- Only one of **AGV1, AGV2** can be loaded at a time in two input stations **IPS1, IPS2**.
- Only one part can be processed at a time by each of **WS2**, **WS3**, while **WS1** can assemble just two parts (a Type1 and a Type2) at a time into a complete part. Three workstations must be protected against overflow and underflow.

These specifications are modeled by nine automata shown in Fig.4.1 :

$$\mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Z}_3, \mathbf{Z}_4, \mathbf{WS}_{13}, \mathbf{WS}_{14}, \mathbf{WS}_2, \mathbf{WS}_3, \mathbf{IPS}$$

In the following, we employ this case to illustrative the steps of our proposed method.

By **Step 1**: We synthesize decentralized supervisors for these nine specifications in the following. Let the event set of the specification $\mathbf{Z}_1$ be $\Sigma_{\mathbf{Z}_1}$. Firstly, we obtain the decentralized plant $\mathbf{G}_{\mathbf{Z}_1}$ which is the synchronous product of the **AGV**s that share events with $\Sigma_{\mathbf{Z}_1}$, namely $\mathbf{G}_{\mathbf{Z}_1} := (\mathbf{AGV1}||\mathbf{AGV2})$. Then we compute the decentralized supervisor $\mathbf{SUP}_{\mathbf{Z}_1}$ such that

$$L_m(\mathbf{SUP}_{\mathbf{Z}_1}) = \sup C(\mathbf{Z}_1||L_m(\mathbf{G}_{\mathbf{Z}_1}))$$

Similarly, we get nine decentralized supervisors

$$\mathbf{SUP}_{\mathbf{Z}_1}, \mathbf{SUP}_{\mathbf{Z}_2}, \mathbf{SUP}_{\mathbf{Z}_3}, \mathbf{SUP}_{\mathbf{Z}_4}, \mathbf{SUP}_{\mathbf{WS}_{13}},$$

$$\mathbf{SUP}_{\mathbf{WS}_{14}}, \mathbf{SUP}_{\mathbf{WS}_2}, \mathbf{SUP}_{\mathbf{WS}_3}, \mathbf{SUP}_{\mathbf{IPS}}.$$

By **Step 2**: We create a DMM $\mathbf{P}$ to record the dependencies between models $\mathbf{AGV1}, .., \mathbf{AGV5}$ and nine decentralized supervisors, which is shown below.

Then, we multiply $\mathbf{P}$ with its tranpose matrix $\mathbf{P}^\top$ to get the DSM shown in Table 3.

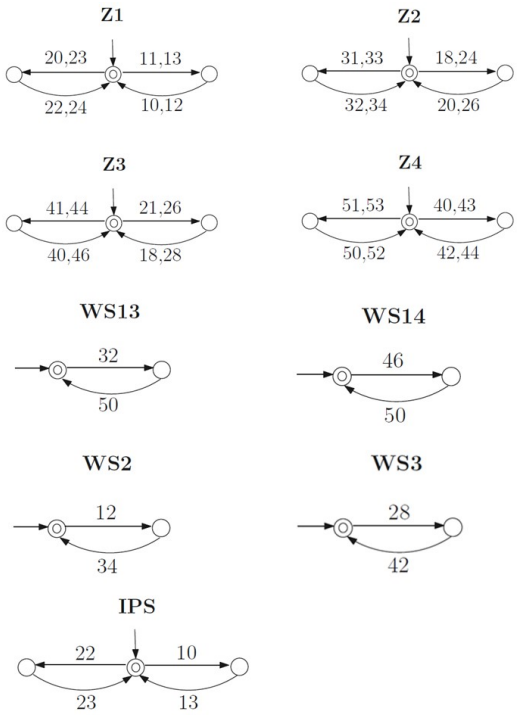

Fig. 3. Specifications of AGVs, taken from [3]

Table 2
The DMM $\mathbf{P}$ of the AGV system.

		$\mathbf{AGV}_1$	$\mathbf{AGV}_2$	$\mathbf{AGV}_3$	$\mathbf{AGV}_4$	$\mathbf{AGV}_5$
$\mathbf{Z}_1$	$\mathbf{SUP}_{\mathbf{Z}_1}$	1	1	0	0	0
$\mathbf{Z}_2$	$\mathbf{SUP}_{\mathbf{Z}_2}$	0	1	1	0	0
$\mathbf{Z}_3$	$\mathbf{SUP}_{\mathbf{Z}_3}$	0	1	0	1	0
$\mathbf{Z}_4$	$\mathbf{SUP}_{\mathbf{Z}_4}$	0	0	0	1	1
$\mathbf{WS}_{13}$	$\mathbf{SUP}_{\mathbf{WS}_{13}}$	0	0	1	0	1
$\mathbf{WS}_{14}$	$\mathbf{SUP}_{\mathbf{WS}_{14}}$	0	0	0	1	1
$\mathbf{WS}_2$	$\mathbf{SUP}_{\mathbf{WS}_2}$	1	0	1	0	0
$\mathbf{WS}_3$	$\mathbf{SUP}_{\mathbf{WS}_3}$	0	1	0	1	0
IPS	$\mathbf{SUP}_{\mathbf{IPS}}$	1	1	0	0	0

Table 3
The DSM of the AGV system: $\mathbf{P_D} = \mathbf{P} \times \mathbf{P}^\top$

	$\mathbf{SUP}_{\mathbf{Z}_1}$	$\mathbf{SUP}_{\mathbf{Z}_2}$	$\mathbf{SUP}_{\mathbf{Z}_3}$	$\mathbf{SUP}_{\mathbf{Z}_4}$	$\mathbf{SUP}_{\mathbf{WS}_{13}}$	$\mathbf{SUP}_{\mathbf{WS}_{14}}$	$\mathbf{SUP}_{\mathbf{WS}_2}$	$\mathbf{SUP}_{\mathbf{WS}_3}$	$\mathbf{SUP}_{\mathbf{IPS}}$
$\mathbf{SUP}_{\mathbf{Z}_1}$	2	1	1	0	0	0	1	1	2
$\mathbf{SUP}_{\mathbf{Z}_2}$	1	2	1	0	1	0	1	1	1
$\mathbf{SUP}_{\mathbf{Z}_3}$	1	1	2	1	0	1	0	2	1
$\mathbf{SUP}_{\mathbf{Z}_4}$	0	0	1	2	1	2	0	1	0
$\mathbf{SUP}_{\mathbf{WS}_{13}}$	0	1	0	1	2	1	1	0	0
$\mathbf{SUP}_{\mathbf{WS}_{14}}$	0	0	1	2	1	2	0	1	0
$\mathbf{SUP}_{\mathbf{WS}_2}$	1	1	0	0	1	0	2	0	1
$\mathbf{SUP}_{\mathbf{WS}_3}$	1	1	2	1	0	1	0	2	1
$\mathbf{SUP}_{\mathbf{IPS}}$	2	1	1	0	0	0	1	1	2

To turn the DSM $\mathbf{P_D}$ into a probability matrix $\mathbf{M}$ for the clustering step, we do the column normalization. The result is:

$$\mathbf{M} = \begin{pmatrix} 0.2500 & 0.1250 & 0.1111 & 0 & 0 & 0 & 0.1667 & 0.1111 & 0.2500 \\ 0.1250 & 0.2500 & 0.1111 & 0 & 0.1667 & 0 & 0.1667 & 0.1111 & 0.1250 \\ 0.1250 & 0.1250 & 0.2222 & 0.1429 & 0 & 0.1429 & 0 & 0.2222 & 0.1250 \\ 0 & 0 & 0.1111 & 0.2857 & 0.1667 & 0.2857 & 0 & 0.1111 & 0 \\ 0 & 0.1250 & 0 & 0.1429 & 0.3333 & 0.1429 & 0.1667 & 0 & 0 \\ 0 & 0 & 0.1111 & 0.2857 & 0.1667 & 0.2857 & 0 & 0.1111 & 0 \\ 0.1250 & 0.1250 & 0 & 0 & 0.1667 & 0 & 0.3333 & 0 & 0.1250 \\ 0.1250 & 0.1250 & 0.2222 & 0.1429 & 0 & 0.1429 & 0 & 0.2222 & 0.1250 \\ 0.2500 & 0.1250 & 0.1111 & 0 & 0 & 0 & 0.1667 & 0.1111 & 0.2500 \end{pmatrix}$$

This $\mathbf{M}$ is used as input of MCL for clustering the decentralized supervisors. For this, we set up the *inflation* parameter β of MCL to different values in order to get different clustering results. For example, when we set $\beta = 4$, we obtain four clusters:

Cluster 1	Cluster 2	Cluster 3	Cluster 4
$\mathbf{SUP_{WS_{13}}}$	$\mathbf{SUP_{Z_4}}$	$\mathbf{SUP_{Z_3}}$	$\mathbf{SUP_{Z_1}}$
	$\mathbf{SUP_{WS_{14}}}$	$\mathbf{SUP_{WS_3}}$	$\mathbf{SUP_{Z_2}}$
			$\mathbf{SUP_{WS_2}}$
			$\mathbf{SUP_{IPS}}$

By **Step 3**: The components of each cluster are combined using synchronous product to create the subsystem model **SUB**, namely:

- $\mathbf{SUB}_1 = \mathbf{SUP_{WS_{13}}}$
- $\mathbf{SUB}_2 = \mathbf{SUP_{Z_4}} || \mathbf{SUP_{WS_{14}}}$
- $\mathbf{SUB}_3 = \mathbf{SUP_{Z_3}} || \mathbf{SUP_{WS_3}}$
- $\mathbf{SUB}_4 = \mathbf{SUP_{Z_1}} || \mathbf{SUP_{Z_2}} || \mathbf{SUP_{WS_2}} || \mathbf{SUP_{IPS}}$

We check the nonblokingness inside of each subsystem and design the coordinator supervisor if conflict exists. The subsystems $\mathbf{SUB}_1, \mathbf{SUB}_2, \mathbf{SUB}_3, \mathbf{SUB}_4$ are nonblocking, so there is no need to design coordinator to remove the conflict inside each subsystem.

By **Step 4**: Then we perform the subsystem abstraction of each subsystem and obtain $\mathbf{ABS}_1, \mathbf{ABS}_2, \mathbf{ABS}_3, \mathbf{ABS}_4$. We employ the synthesis procedure of $\mathbf{SUB}_1$ to introduce this step. Firstly, the shared event set Σ_o of the four subsystems is given by

$$\Sigma_o := \{\sigma \in \Sigma | \sigma \in (\Sigma_{\mathbf{SUB}_1} \cap \Sigma_{\mathbf{SUB}_2} \cap \Sigma_{\mathbf{SUB}_3} \cap \Sigma_{\mathbf{SUB}_4})\},$$

where $\Sigma_{\mathbf{SUB}_i}$ is the event set of $\mathbf{SUB}_i$ for $i \in [1,4]$. Let $\Sigma_{\mathbf{ABS}_1}$ be the event set of $\mathbf{ABS}_1$. Then we get that $\Sigma_{\mathbf{ABS}_1} := \Sigma_o \cap \Sigma_{\mathbf{SUB}_1}$. $\mathbf{ABS}_1$ thus can be generated by a natural projection $P_1 : \Sigma^*_{\mathbf{SUB}_1} \to \Sigma^*_{\mathbf{ABS}_1}$. Check and extend P_1 to ensure P_1 satisfies the natural observer and OCC.

Return back to **Step 2**: We perform the clustering again for the four abstractions $\mathbf{ABS}_1, ..., \mathbf{ABS}_4$ and five plant components $\mathbf{AGV1}, ..., \mathbf{AGV5}$. Construct a new DMM $\mathbf{P}$ to record the dependencies between abstractions and plant components. The matrices $\mathbf{P}$, $\mathbf{P}_D$ and $\mathbf{M}$ for this second level clustering are shown below.

Table 4
The DMM $\mathbf{P}$ of the **AGVs** and abstractions

	$\mathbf{AGV}_1$	$\mathbf{AGV}_2$	$\mathbf{AGV}_3$	$\mathbf{AGV}_4$	$\mathbf{AGV}_5$
$\mathbf{ABS}_1$	0	0	1	0	1
$\mathbf{ABS}_2$	0	0	0	1	1
$\mathbf{ABS}_3$	0	1	0	1	0
$\mathbf{ABS}_4$	1	1	1	0	0

Table 5
$\mathbf{P_D} = \mathbf{P} \times \mathbf{P}^{\top}$

	$\mathbf{ABS}_1$	$\mathbf{ABS}_2$	$\mathbf{ABS}_3$	$\mathbf{ABS}_4$
$\mathbf{ABS}_1$	2	1	0	1
$\mathbf{ABS}_2$	1	2	1	0
$\mathbf{ABS}_3$	0	1	2	1
$\mathbf{ABS}_4$	1	0	1	3

$$\mathbf{M} = \begin{pmatrix} 0.50 & 0.25 & 0 & 0.20 \\ 0.25 & 0.50 & 0.25 & 0 \\ 0 & 0.25 & 0.50 & 0.20 \\ 0.25 & 0 & 0.25 & 0.60 \end{pmatrix}$$

Then we apply the clustering algorithm again using the same $\beta(=4)$.

Cluster 1	Cluster 2	Cluster 3	Cluster 4
$\mathbf{ABS}_1$	$\mathbf{ABS}_2$	$\mathbf{ABS}_3$	$\mathbf{ABS}_4$

As shown above, the cluster number does not reduce compared to the previous clustering. Therefore, we reduce β value by 0.5 (i.e. $\beta = 3.5$), and show the result in the following.

Cluster 1	Cluster 2
$\mathbf{ABS}_2$	$\mathbf{ABS}_1$
	$\mathbf{ABS}_3$
	$\mathbf{ABS}_4$

By **Step 3**: After the new clustering, we get 2 new clusters in this upper layer. Specifically, there is only one abstraction in cluster 1, so $\mathbf{ABS}_2$ is the new subsystem directly. For cluster 2, however, there are three abstractions $\mathbf{ABS}_1$, $\mathbf{ABS}_3$, $\mathbf{ABS}_4$. We thus get a new subsystem for cluster 2 by the synchronous product of these three abstractions, i.e.,

- $\mathbf{SUB}_{layer2,1} = \mathbf{ABS}_2$
- $\mathbf{SUB}_{layer2,2} = \mathbf{ABS}_1 || \mathbf{ABS}_3 || \mathbf{ABS}_4$

Check possible conflict inside $\mathbf{SUB}_{layer2,1}, \mathbf{SUB}_{layer2,2}$ and get that they are nonblocking.

By **Step 4**: Then we abstract $\mathbf{SUB}_{layer2,1}$ and $\mathbf{SUB}_{layer2,2}$ again, and get two abstractions $\mathbf{ABS}_{layer2,1}$ and $\mathbf{ABS}_{layer2,2}$. We treat these two abstractions as one cluster and find that there exist conflicts between these two abstractions. A coordinator supervisor **CO** thus be computed to remove the conflicts. The resulting control architecture for the AGV system is shown in Fig. 4.

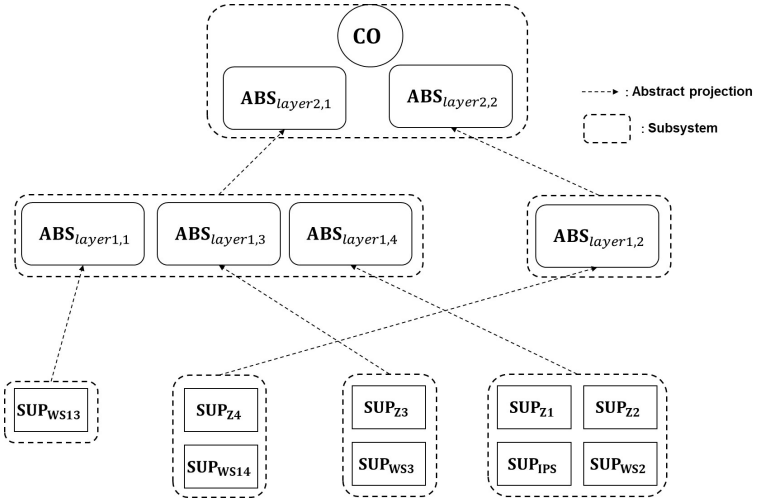

Fig. 4. Control architecture resulted from our approach

Our result consists of nine decentralized supervisors $\mathbf{SUP_{Z_1}}$, $\mathbf{SUP_{Z_2}}$, $\mathbf{SUP_{Z_3}}$, $\mathbf{SUP_{Z_4}}$, $\mathbf{SUP_{WS_{13}}}$, $\mathbf{SUP_{WS_{14}}}$, $\mathbf{SUP_{WS_2}}$, $\mathbf{SUP_{WS_3}}$, $\mathbf{SUP_{IPS}}$, and one coordinator **CO**. For the initial parameter $\beta = 4$, the resulting control architecture has two abstraction levels, in which the coordinator **CO** has 10 states.

β	Cluster numbers of decentralized supervisor	The number of abstraction levels	Coordinator (state numbers)
4	4	2	**CO** (10)

For this AGV system, the standard centralized approach is still possible. The monolithic supervisor **SUP** has 4406 states. So we can confirm that the joint behavior of the supervisors and the coordinator is the same as the monolithic controlled behavior, i.e.,

$$\mathbf{SUP_{Z_1}}||...||\mathbf{SUP_{IPS}}||\mathbf{CO} = \mathbf{SUP}$$

We change the clustering *inflation* β and find that the number of cluster increases as β increases. The following table is the result obtained based on the assumption $2.5 \leq \beta \leq 10$, which includes the number that the decentralized supervisors are clustered, the number of abstraction, and the state number of the coordinator.

β	Cluster numbers for decentralized supervisor	The number of abstraction	Coordinator (state numbers)
2.5	2	1	**CO** (64)
3	3	2	**CO** (51)
3.5	3	2	**CO** (27)
4	4	2	**CO** (10)
4.5	4	2	**CO** (10)
5	4	2	**CO** (10)
5.5	5	3	**CO** (10)
6	5	3	**CO** (10)
6.5	5	3	**CO** (10)
7	5	3	**CO** (10)
7.5	5	3	**CO** (10)
8	6	3	**CO** (10)
8.5	6	3	**CO** (10)
9	6	3	**CO** (10)
9.5	6	3	**CO** (10)
10	6	3	**CO** (10)

Comparison of coordinators

As shown in Table 4.1, there are 13 coordinators having 10 states for inflation $\beta = 4$ to $\beta = 10$. It is verified that these 13 coordinators are isomorphic. The coordinator $\mathbf{CO}_{(10)}$ is shown in Fig.6. On the other hand, relying on engineering insight for system decomposition, it is known that the coordinator $\mathbf{CO}_{(7)}$ for the same AGV system designed by the approach in [7] has 7 states which is shown in Fig.4.5. We now make a brief comparison between $\mathbf{CO}_{(7)}$ and $\mathbf{CO}_{(10)}$.

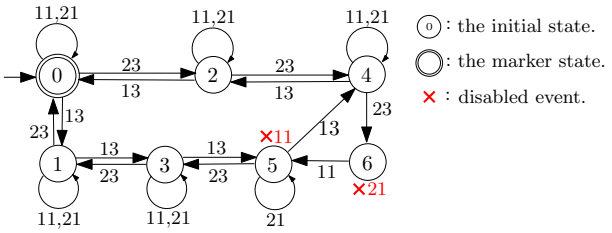

Fig. 5. 7-state coordinator by approach in [7]

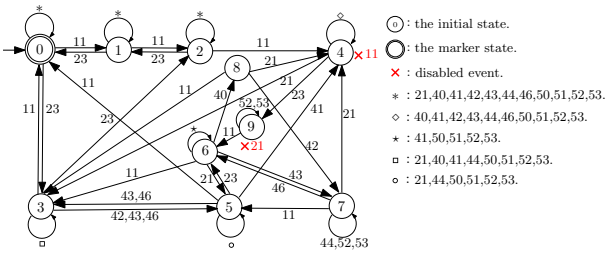

Fig. 6. 10-state coordinator by our approach

The control logic of $\mathbf{CO}_{(7)}$ is:
1. Disable event 11 (**AGV1** leaves **WS2** and enters Zone 1) if event 13 (**AGV1** re-enters Zone 1) has occurred three more times than event 23 (**AGV2** re-enters Zone 1).
2. Disable event 21 (**AGV2** leaves **WS3** and enters Zone 3) if event 23 has occurred three more times than event 13.

The control logic of $\mathbf{CO}_{(10)}$ is:
1. Disable event 11 if event 11 has occurred three more times than event 23.
This first control logic of $\mathbf{CO}_{(10)}$ is the same as $\mathbf{CO}_{(7)}$. The reason is that for the automaton of **AGV1** shown in Fig.2, events 11 and 13 can only occur cyclically, so observing event 11 is equivalent to observing event 13 for the purpose of controlling 11.
2. Disable event 21 if event 23 has occurred three more times than event 11.

4.2 Production Cell

The Production Cell system [1],[3] consists of nine asynchronous component agents: Stock, Feed Belt, Elevating Rotary Table, Rotary Base, Arm1, Arm2, Press, Deposit Belt, and Crane. The cell processes workpieces, called 'blanks', having state size of order 10^7. The system is shown Fig.7.

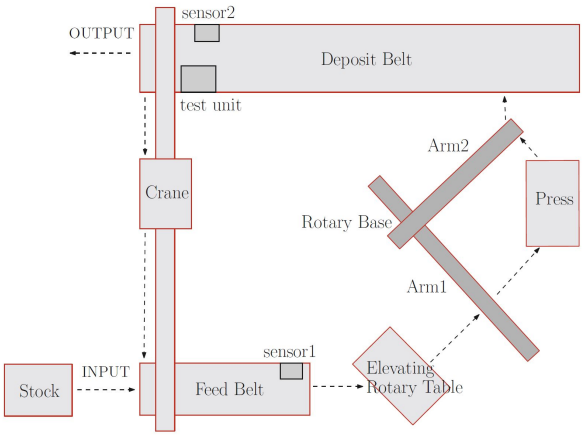

Fig. 7. Overview of Production Cell configuration, taken from [3]

These nine asynchronous component agents are modeled by eleven component automata, and there are eighteen control specifications. Please see Chap.5 in [3] for the detailed description of each automaton and event. We construct an abstraction based hierarchical architecture for Production Cell system by our approach.

By **Step 1**: We synthesize a decentralized supervisor for each specification. Then we get eighteen decentralized supervisors.

Components	Specifications	Decentralized supervisors
St	**FB1 FB2**	$\mathbf{SUP_{FB1}}$ $\mathbf{SUP_{FB2}}$
FB	**Ta1 Ta2**	$\mathbf{SUP_{Ta1}}$ $\mathbf{SUP_{Ta2}}$
Ta_V	**Ta3 Ta4**	$\mathbf{SUP_{Ta3}}$ $\mathbf{SUP_{Ta4}}$
Ta_H	**A1T Pr1**	$\mathbf{SUP_{A1T}}$ $\mathbf{SUP_{Pr1}}$
Pr	**Pr2 A1P**	$\mathbf{SUP_{Pr2}}$ $\mathbf{SUP_{A1P}}$
Ro	**A2P R1**	$\mathbf{SUP_{A2P}}$ $\mathbf{SUP_{R1}}$
A1	**R2 R3**	$\mathbf{SUP_{R2}}$ $\mathbf{SUP_{R3}}$
A2	**R4 DB1**	$\mathbf{SUP_{R4}}$ $\mathbf{SUP_{DB1}}$
DB	**DB2 DB3**	$\mathbf{SUP_{DB2}}$ $\mathbf{SUP_{DB3}}$
Cr_V		
Cr_H		

By **Step 2**: Next we create a DMM **P** to record the dependencies between these eleven component models and eighteen decentralized supervisors. The matrix **P** is shown below.

Table 6
The DMM **P** of the Production Cell system.

	St	**FB**	**Ta_V**	**Ta_H**	**Pr**	**Ro**	**A1**	**A2**	**DB**	**Cr_V**	**Cr_H**
$\mathbf{SUP_{FB1}}$	1	1	0	0	0	0	0	0	0	1	1
$\mathbf{SUP_{FB2}}$	1	1	0	0	0	0	0	0	0	1	1
$\mathbf{SUP_{Ta1}}$	0	1	1	0	0	0	0	0	0	0	0
$\mathbf{SUP_{Ta2}}$	0	0	1	0	0	0	1	0	0	0	0
$\mathbf{SUP_{Ta3}}$	0	1	0	1	0	0	0	0	0	0	0
$\mathbf{SUP_{Ta4}}$	0	0	0	1	0	0	1	0	0	0	0
$\mathbf{SUP_{A1T}}$	0	0	1	0	0	1	0	0	0	0	0
$\mathbf{SUP_{Pr1}}$	0	0	0	0	1	0	1	0	0	0	0
$\mathbf{SUP_{Pr2}}$	0	0	0	0	1	0	0	1	0	0	0
$\mathbf{SUP_{A1P}}$	0	0	0	0	1	1	1	0	0	0	0
$\mathbf{SUP_{A2P}}$	0	0	0	0	1	1	0	1	0	0	0
$\mathbf{SUP_{R1}}$	0	0	0	0	0	1	1	0	0	0	0
$\mathbf{SUP_{R2}}$	0	0	0	0	0	1	0	1	0	0	0
$\mathbf{SUP_{R3}}$	0	0	0	0	0	1	1	0	0	0	0
$\mathbf{SUP_{R4}}$	0	0	0	0	0	1	0	1	0	0	0
$\mathbf{SUP_{DB1}}$	0	0	0	0	0	0	0	1	1	1	1
$\mathbf{SUP_{DB2}}$	0	0	0	0	0	0	0	1	1	0	0
$\mathbf{SUP_{DB3}}$	0	0	0	0	0	0	0	0	1	1	1

After this step, we multiply **P** with its tranpose matrix $\mathbf{P}^\top$ to get the DSM shown in Table 7.

Table 7
The DSM of the Production Cell system. $\mathbf{P_D} = \mathbf{P} \times \mathbf{P}^{\top}$

	$\mathbf{SUP_{FB1}}$	$\mathbf{SUP_{FB2}}$	$\mathbf{SUP_{Ta1}}$	$\mathbf{SUP_{Ta2}}$	$\mathbf{SUP_{Ta3}}$	$\mathbf{SUP_{Ta4}}$	$\mathbf{SUP_{A1T}}$	$\mathbf{SUP_{Pr1}}$	$\mathbf{SUP_{Pr2}}$	$\mathbf{SUP_{A1P}}$	$\mathbf{SUP_{A2P}}$	$\mathbf{SUP_{R1}}$	$\mathbf{SUP_{R2}}$	$\mathbf{SUP_{R3}}$	$\mathbf{SUP_{R4}}$	$\mathbf{SUP_{DB1}}$	$\mathbf{SUP_{DB2}}$	$\mathbf{SUP_{DB3}}$
$\mathbf{SUP_{FB1}}$	4	4	1	0	1	0	0	0	0	0	0	0	0	0	0	2	0	2
$\mathbf{SUP_{FB2}}$	4	4	1	0	1	0	0	0	0	0	0	0	0	0	0	2	0	2
$\mathbf{SUP_{Ta1}}$	1	1	2	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
$\mathbf{SUP_{Ta2}}$	0	0	1	2	0	1	1	1	0	1	0	1	0	1	0	0	0	0
$\mathbf{SUP_{Ta3}}$	1	1	1	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0
$\mathbf{SUP_{Ta4}}$	0	0	0	1	1	2	0	1	0	1	0	1	0	1	0	0	0	0
$\mathbf{SUP_{A1T}}$	0	0	1	1	0	0	2	0	0	1	1	1	1	1	1	0	0	0
$\mathbf{SUP_{Pr1}}$	0	0	0	1	0	1	0	2	1	2	1	1	0	1	0	0	0	0
$\mathbf{SUP_{Pr2}}$	0	0	0	0	0	0	0	1	2	1	2	0	1	0	1	1	1	0
$\mathbf{SUP_{A1P}}$	0	0	0	1	0	1	1	2	1	3	2	2	1	2	1	0	0	0
$\mathbf{SUP_{A2P}}$	0	0	0	0	0	0	1	1	2	2	3	1	2	1	2	1	1	0
$\mathbf{SUP_{R1}}$	0	0	0	1	0	1	1	1	0	2	1	2	1	2	1	0	0	0
$\mathbf{SUP_{R2}}$	0	0	0	0	0	0	1	0	1	1	2	1	2	1	2	1	1	0
$\mathbf{SUP_{R3}}$	0	0	0	1	0	1	1	1	0	2	1	2	1	2	1	0	0	0
$\mathbf{SUP_{R4}}$	0	0	0	0	0	0	1	0	1	1	2	1	2	1	2	1	1	0
$\mathbf{SUP_{DB1}}$	2	2	0	0	0	0	0	0	1	0	1	0	1	0	1	4	2	3
$\mathbf{SUP_{DB2}}$	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	2	2	1
$\mathbf{SUP_{DB3}}$	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	3	1	3

To turn the DSM $\mathbf{P_D}$ into a probability matrix $\mathbf{M}$ for the clustering step, we do the column normalization. The result is given by:

$$\mathbf{M} = \begin{pmatrix}
0.2857 & 0.2857 & 0.1429 & 0 & 0.1667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1176 & 0 & 0.1818 \\
0.2857 & 0.2857 & 0.1429 & 0 & 0.1667 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1176 & 0 & 0.1818 \\
0.0714 & 0.0714 & 0.2857 & 0.1111 & 0.1667 & 0 & 0.1000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.1429 & 0.2222 & 0 & 0.1250 & 0.1000 & 0.1000 & 0 & 0.0588 & 0 & 0.0769 & 0 & 0.0769 & 0 & 0 & 0 & 0 \\
0.0714 & 0.0714 & 0.1429 & 0 & 0.3333 & 0.1250 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.1111 & 0.1667 & 0.2500 & 0 & 0.1000 & 0 & 0.0588 & 0 & 0.0769 & 0 & 0.0769 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.1429 & 0.1111 & 0 & 0 & 0.2000 & 0 & 0 & 0.0588 & 0.0588 & 0.0769 & 0.0769 & 0.0769 & 0.0769 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.1111 & 0 & 0.1250 & 0 & 0.2000 & 0.1000 & 0.1176 & 0.0588 & 0.0769 & 0 & 0.0769 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1000 & 0.2000 & 0.0588 & 0.1176 & 0 & 0.0769 & 0 & 0.0769 & 0.0588 & 0.1111 & 0 \\
0 & 0 & 0 & 0.1111 & 0 & 0.1250 & 0.1000 & 0.2000 & 0.1000 & 0.1765 & 0.1176 & 0.1538 & 0.0769 & 0.1538 & 0.0769 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.1000 & 0.1000 & 0.2000 & 0.1176 & 0.1765 & 0.0769 & 0.1538 & 0.0769 & 0.1538 & 0.0588 & 0.1111 & 0 \\
0 & 0 & 0 & 0.1111 & 0 & 0.1250 & 0.1000 & 0.1000 & 0 & 0.1176 & 0.0588 & 0.1538 & 0.0769 & 0.1538 & 0.0769 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.1000 & 0 & 0.1000 & 0.0588 & 0.1176 & 0.0769 & 0.1538 & 0.0769 & 0.1538 & 0.0588 & 0.1111 & 0 \\
0 & 0 & 0 & 0.1111 & 0 & 0.1250 & 0.1000 & 0.1000 & 0 & 0.1176 & 0.0588 & 0.1538 & 0.0769 & 0.1538 & 0.0769 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.1000 & 0 & 0.1000 & 0.0588 & 0.1176 & 0.0769 & 0.1538 & 0.0769 & 0.1538 & 0.0588 & 0.1111 & 0 \\
0.1429 & 0.1429 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1000 & 0 & 0.0588 & 0 & 0.0769 & 0 & 0.0769 & 0.2353 & 0.2222 & 0.2727 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1000 & 0 & 0.0588 & 0 & 0.0769 & 0 & 0.0769 & 0.1176 & 0.2222 & 0.0909 \\
0.1429 & 0.1429 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1765 & 0.1111 & 0.2727
\end{pmatrix}$$

This $\mathbf{M}$ is used as input of MCL for clustering the decentralized supervisors. Note that in this Production Cell system, the computation is feasible only when $inflation\ \beta = 20$. After first clustering, the eighteen decentralized supervisors are grouped into five clusters:
Cluster1 :$\mathbf{SUP_{DB1}}$, $\mathbf{SUP_{DB2}}$, $\mathbf{SUP_{DB3}}$.
Cluster2 :$\mathbf{SUP_{Pr2}}$, $\mathbf{SUP_{A2P}}$, $\mathbf{SUP_{R2}}$, $\mathbf{SUP_{R4}}$.
Cluster3 :$\mathbf{SUP_{Ta2}}$, $\mathbf{SUP_{Ta4}}$, $\mathbf{SUP_{A1T}}$, $\mathbf{SUP_{Pr1}}$, $\mathbf{SUP_{A1P}}$, $\mathbf{SUP_{R1}}$, $\mathbf{SUP_{R3}}$.
Cluster4 :$\mathbf{SUP_{Ta1}}$.
Cluster5 :$\mathbf{SUP_{FB1}}$, $\mathbf{SUP_{FB2}}$, $\mathbf{SUP_{Ta3}}$.

By **Step 3**: We synchronize the automata in the same cluster, and obtain five subsystems $\mathbf{SUB}_{layer1,1}$,..., $\mathbf{SUB}_{layer1,5}$:

- $\mathbf{SUB}_{layer1,1} = \mathbf{SUP_{DB1}}||\mathbf{SUP_{DB2}}||\mathbf{SUP_{DB3}}$.
- $\mathbf{SUB}_{layer1,2} = \mathbf{SUP_{Pr2}}||\mathbf{SUP_{A2P}}||\mathbf{SUP_{R2}}||\mathbf{SUP_{R4}}$.
- $\mathbf{SUB}_{layer1,3} = \mathbf{SUP_{Ta2}}||\mathbf{SUP_{Ta4}}||\mathbf{SUP_{A1T}}||\mathbf{SUP_{Pr1}}||\mathbf{SUP_{A1P}}$ $||\mathbf{SUP_{R1}}||\mathbf{SUP_{R3}}$.
- $\mathbf{SUB}_{layer1,4} = \mathbf{SUP_{Ta1}}$.
- $\mathbf{SUB}_{layer1,5} = \mathbf{SUP_{FB1}}||\mathbf{SUP_{FB2}}||\mathbf{SUP_{Ta3}}$.

Check conflicts inside each subsystem and find that there are two subsystems with conflicts, so we design the coordinators $\mathbf{CO}_1$ and $\mathbf{CO}_2$ to remove conflicts in $\mathbf{SUB}_{layer1,2}$ and $\mathbf{SUB}_{layer1,3}$, respectively.

By **Step 4**: We abstract the subsystems and get five abstraction models $\mathbf{ABS}_{layer1,1}, ..., \mathbf{ABS}_{layer1,5}$.

Return back to **Step 2**: Construct a new DMM **P** to record the dependencies between abstractions and plant components. The matrices **P**, $\mathbf{P_D}$ and **M** for this second level clustering are shown below.

Table 8
The DMM **P** of abstractions on layer 1.

	St	**FB**	**Ta_V**	**Ta_H**	**Pr**	**Ro**	**A1**	**A2**	**DB**	**Cr_V**	**Cr_H**
$\mathbf{ABS}_{layer1,1}$	0	0	0	0	0	0	0	1	1	1	1
$\mathbf{ABS}_{layer1,2}$	0	0	0	0	1	1	0	1	0	0	0
$\mathbf{ABS}_{layer1,3}$	0	0	1	1	1	1	1	0	0	0	0
$\mathbf{ABS}_{layer1,4}$	0	1	1	0	0	0	0	0	0	0	0
$\mathbf{ABS}_{layer1,5}$	1	1	0	1	0	0	0	0	0	1	1

Table 9
The DSM of of abstractions on layer 1. $\mathbf{P_D} = \mathbf{P} \times \mathbf{P}^{\top}$

	$\mathbf{ABS}_{layer1,1}$	$\mathbf{ABS}_{layer1,2}$	$\mathbf{ABS}_{layer1,3}$	$\mathbf{ABS}_{layer1,4}$	$\mathbf{ABS}_{layer1,5}$
$\mathbf{ABS}_{layer1,1}$	4	1	0	0	2
$\mathbf{ABS}_{layer1,2}$	1	3	2	0	0
$\mathbf{ABS}_{layer1,3}$	0	2	5	1	1
$\mathbf{ABS}_{layer1,4}$	0	0	1	2	1
$\mathbf{ABS}_{layer1,5}$	2	0	1	1	5

$$\mathbf{M} = \begin{pmatrix} 0.5714 & 0.1667 & 0 & 0 & 0.2222 \\ 0.1429 & 0.5000 & 0.2222 & 0 & 0 \\ 0 & 0.3333 & 0.5556 & 0.2500 & 0.1111 \\ 0 & 0 & 0.1111 & 0.5000 & 0.1111 \\ 0.2857 & 0 & 0.1111 & 0.2500 & 0.5556 \end{pmatrix}$$

By **Step 3**: By clustering these five abstractions, we get 3 clusters as shown below.

Cluster 1	Cluster 2	Cluster 3
$\mathbf{ABS}_{layer1,5}$	$\mathbf{ABS}_{layer1,2}$	$\mathbf{ABS}_{layer1,1}$
	$\mathbf{ABS}_{layer1,3}$	
	$\mathbf{ABS}_{layer1,4}$	

The components of each cluster are combined using synchronous product to create the subsystems $\mathbf{SUB}_{layer2,1}, \ldots,$ $\mathbf{SUB}_{layer2,3}$. Since cluster 1 and cluster 3 only have one abstraction, abstractions $\mathbf{ABS}_{layer1,1}$ and $\mathbf{ABS}_{layer1,5}$ are already the new subsystems respectively:

- $\mathbf{SUB}_{layer2,1} = \mathbf{ABS}_{layer1,5}$
- $\mathbf{SUB}_{layer2,2} = \mathbf{ABS}_{layer1,1} || \mathbf{ABS}_{layer1,3} || \mathbf{ABS}_{layer1,4}$
- $\mathbf{SUB}_{layer2,3} = \mathbf{ABS}_{layer1,1}$

The second subsystem $\mathbf{SUB}_{layer2,2}$ has conflicts, so a coordinator $\mathbf{CO}_3$ is designed to make $\mathbf{SUB}_{layer2,2}$ nonblocking.

By **Step 4**: These three subsystems $\mathbf{SUB}_{layer2,1}, .., \mathbf{SUB}_{layer2,3}$ in layer 2 are further abstracted into three abstractions $\mathbf{ABS}_{layer2,1}$, $\mathbf{ABS}_{layer2,2}$, $\mathbf{ABS}_{layer2,3}$.

Return back to **Step 2**: For this upper layer, we also construct a new DMM **P** to record the dependencies between abstractions $\mathbf{ABS}_{layer2,1}$, $\mathbf{ABS}_{layer2,2}$, $\mathbf{ABS}_{layer2,3}$ and plant components. The matrices **P**, $\mathbf{P_D}$ and **M** for this third level clustering are shown below.

Table 10
The DMM **P** of abstractions on layer 2.

	St	**FB**	**Ta_V**	**Ta_H**	**Pr**	**Ro**	**A1**	**A2**	**DB**	**Cr_V**	**Cr_H**
$\mathbf{ABS}_{layer2,1}$	1	1	0	1	0	0	0	0	0	1	1
$\mathbf{ABS}_{layer2,2}$	0	1	0	1	0	0	0	1	0	0	0
$\mathbf{ABS}_{layer2,3}$	0	0	0	0	0	0	0	1	1	1	1

Table 11
The DSM of of abstractions on layer 2. $\mathbf{P_D} = \mathbf{P} \times \mathbf{P}^{\top}$

	$\mathbf{ABS}_{layer2,1}$	$\mathbf{ABS}_{layer2,2}$	$\mathbf{ABS}_{layer2,3}$
$\mathbf{ABS}_{layer2,1}$	5	2	2
$\mathbf{ABS}_{layer2,2}$	2	3	1
$\mathbf{ABS}_{layer2,3}$	2	1	4

$$\mathbf{M} = \begin{pmatrix} 0.5556 & 0.3333 & 0.2857 \\ 0.2222 & 0.5000 & 0.1429 \\ 0.2222 & 0.1667 & 0.5714 \end{pmatrix}$$

By **Step 3**: Perform the clustering again for these three abstractions and we obtain two clusters:

Cluster 1	Cluster 2
$\mathbf{ABS}_{layer2,1}$	$\mathbf{ABS}_{layer2,3}$
$\mathbf{ABS}_{layer2,2}$	

For cluster 1, we perform synchronous product of these two abstractions to obtain a new subsystem $\mathbf{SUB}_{layer3,1}$. There is only one abstraction in cluster 2, so $\mathbf{ABS}_{layer2,3}$ already is the new subsystem $\mathbf{SUB}_{layer3,2}$.

- $\mathbf{SUB}_{layer3,1} = \mathbf{ABS}_{layer2,1} || \mathbf{ABS}_{layer2,3}$
- $\mathbf{SUB}_{layer3,2} = \mathbf{ABS}_{layer2,2}$

Each of these two subsystems $\mathbf{SUB}_{layer3,1}, \mathbf{SUB}_{layer3,2}$ is checked to be nonblocking, so no new coordinator is designed.

By **Step 4**: We further abstract $\mathbf{SUB}_{layer3,1}$ and $\mathbf{SUB}_{layer3,2}$ to get abstractions $\mathbf{ABS}_{layer3,1}$ and $\mathbf{ABS}_{layer3,2}$. We treat these two abstractions as one cluster. In this cluster, the two abstractions are conflicting, so $\mathbf{CO}_4$ is designed to remove the conflicts.

The design procedure ends here since the number of clusters is no more than two. There are eighteen decentralized supervisors and four coordinators $\mathbf{CO}_1, ..., \mathbf{CO}_4$. The state number of the coordinators is shown Table 12. We also give the models of $\mathbf{CO}_1, ..., \mathbf{CO}_4$ which are shown in Fig.8 and Fig.9. The system construct obtained by our approach is shown in Fig.10.

Table 12
Four coordinators are computed

$\beta = 20$	$\mathbf{CO_1}$	$\mathbf{CO_2}$	$\mathbf{CO_3}$	$\mathbf{CO_4}$
State numbers	3	3	2	16

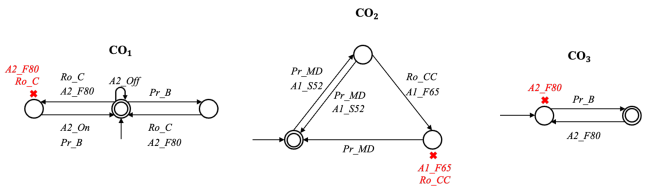

Fig. 8. Coordinators $\mathbf{CO}_1$, $\mathbf{CO}_2$, $\mathbf{CO}_3$ by our approach

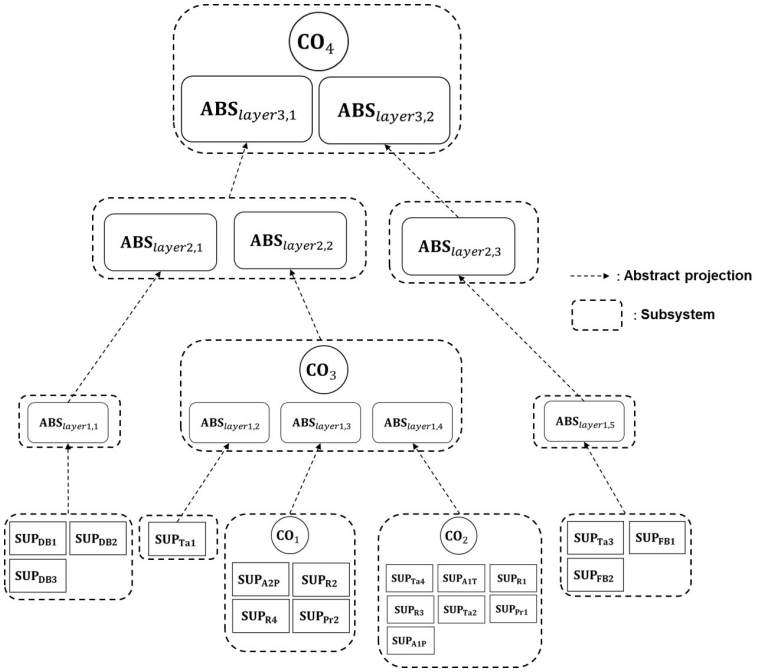

Fig. 10. System construct by our approach

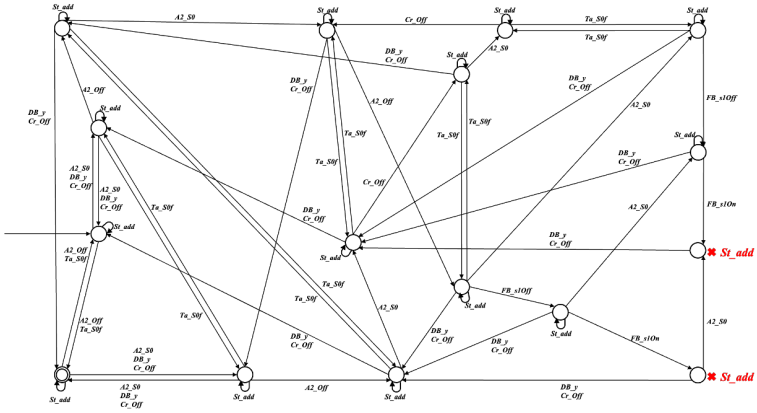

Fig. 9. Coordinator $\mathbf{CO}_4$ by our approach

Comparison of coordinators Here we compare the coordinators $\mathbf{CO}_{01}$ and $\mathbf{CO}_{02}$ designed by the approach in [7] and our obtained coordinators $\mathbf{CO}_1, ..., \mathbf{CO}_4$, where $\mathbf{CO}_{01}$ and $\mathbf{CO}_{02}$ are shown in Fig.11 and Fig.12.

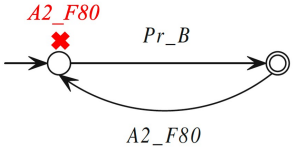

Fig. 11. Coordinator $\mathbf{CO}_{01}$ by approach in [7]

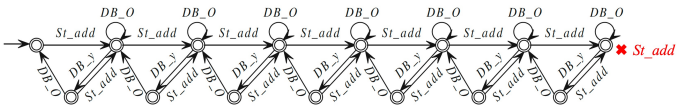

Fig. 12. Coordinator $\mathbf{CO}_{02}$ by approach in [7]

The control logic of $\mathbf{CO}_{01}$ shows that the event $A2\_F80$ (Arm2 extends its length to 80) is disabled before event Pr_B occurs (Press descends back to bottom and prepares to unload the forged blank to Arm2), to make Arm2 to stay at its initial state during the first work cycle of Press.

The control logic of $\mathbf{CO}_{02}$ is to disable event St_add (adding a blank into the cell) if and when there are 7 blanks. The reason is that the total capacity of the production cell is eight and the whole cell contains a loop with material feedback: faulty blanks are transported by Crane from DBelt back to FBelt. Moreover, the feedback event DB_n (the blank fails the check) is uncontrollable, so at least one empty slot must be maintained in the cell.

For the four coordinators $\mathbf{CO}_1, ..., \mathbf{CO}_4$ synthesized by our approach, the control logic and the comparison with $\mathbf{CO}_{01}$ and $\mathbf{CO}_{02}$ are given in the following.

The control logic of $\mathbf{CO}_1$:
1. When event Ro_C occurs (RBase turns clockwise back to 40 degrees), the event $A2\_F80$ is disabled before the event Pr_B occurs. Since event Ro_C occurs, Press is not in its first work cycle. Therefore, the difference between $\mathbf{CO}_{01}$ and $\mathbf{CO}_1$ is that: $\mathbf{CO}_1$ ensures that if RBase is at 40 degrees, Arm2 can extend its length to 80 only when Press descends back to bottom to avoid the collision, instead of forcing Arm2 to stay put at its initial state during the first work cycle of Press by $\mathbf{CO}_{01}$. Note that this control logic is in fact redundant because this collision prevention is already ensured by supervisor $\mathbf{SUP_{A2P}}$.
2. Disable event Ro_C when Arm2 has extended its length to 80 (event $A2\_F80$) but not placed a blank onto DBelt yet. This control logic is not contained in $\mathbf{CO}_{01}$ and $\mathbf{CO}_{02}$. Note that this second control of $\mathbf{CO}_1$ is also redundant because supervisor $\mathbf{SUP_{R4}}$ already has this control logic.

Next, the control logic of $\mathbf{CO}_2$:
1. Disable event Ro_CC (RBase turns clockwise to 90 degrees) when Arm1 has extended its length to 65 (event $A2\_F65$), for preventing the collision.
2. Disable event $A2\_F65$ to avoid the collision circumstances that RBase is at 90 degrees (event Ro_CC occurs) and the length of Arm1 is longer than 37 ($A2\_S52$ occurs).
These two control logic prevent Arm1 from collision, which are also contained in supervisor $\mathbf{SUP_{A1P}}$.

As shown in Fig.8, the control logic of $\mathbf{CO}_3$ is same as $\mathbf{CO}_{01}$ which disables event $A2\_F80$ before event Pr_B occurs.

The control logic of $\mathbf{CO}_4$:
When DBelt already holds two blanks (event $A2\_Off$ occurs twice) and ERTable already holds one blank (event FB_s1Off occurs), disable event St_add if there exists a new blank in Fbelt (event FB_s1On occurs).

This control logic is similar to $\mathbf{CO}_{02}$ that $\mathbf{CO}_4$ ensures that Fbelt has one empty slot at least such that the faulty blanks can be transported by Crane from DBelt back to FBelt. The difference is that $\mathbf{CO}_{02}$ only observes how many blanks are added in the PC and how many blanks are passed the test (event DB_y) or outputted from the system (event DB_O). There are eight states in $\mathbf{CO}_{02}$ to execute its control logic. However, the $\mathbf{CO}_4$ (16 states) needs to observe the blank number in Fbelt, ERTable, and DBelt to determine if the event St_add needs to be disabled to prevent the system from being 'chocked'.

5 Conclusions

This paper has considered the nonblocking heterarchical supervisory control problem of large-scale DES. A streamlined algorithm has been proposed to solve this problem, which automatically conducts horizontal decomposition and vertical aggregation of the system and achieves the global nonblocking. There are three inputs of the algorithm: plant component models, specifications models, and a clustering parameter. Without any engineering insight into system structure, the outputs are decentralized supervisors and coordinators whose joint behavior has been proved to be globally nonblocking and maximal permissive. Moreover, we have demonstrated our proposed approach by two large-scale benchmark examples. In the future, we aim to generalize our approach to finite state machine models and systems under partial observations.

References

[1] *Formal Development of Reactive Systems - Case Study Production Cell*, Berlin, Heidelberg, 1995. Springer-Verlag.

[2] Christine Agethen. *A Hierarchical Control Architecture for ω-Languages*. FAU University Press, 2017.

[3] Kai Cai and W. M. Wonham. *Supervisor Localization: A Top-Down Approach to Distributed Control of Discrete-Event Systems*. Springer International Publishing, 2016.

[4] Mike Danilovic and Tyson Browning. Managing complex product development projects with design structure matrices and domain mapping matrices. *International Journal of Project Management*, 25:300–314, 2007.

[5] Max H De Queiroz and Jose ER Cury. Modular supervisory control of large scale discrete event systems. In *Discrete Event Systems: Analysis and Control*, pages 103–110. Springer, 2000.

[6] Stijn Dongen. Graph clustering via a discrete uncoupling process. siam j matrix anal appl, 30:121-141. *SIAM J. Matrix Analysis Applications*, 30:121–141, 01 2008.

[7] Lei Feng and W. M. Wonham. Supervisory control architecture for discrete-event systems. *IEEE Transactions on Automatic Control*, 53(6):1449–1461, 2008.

[8] Martijn Goorden, Joanna van de Mortel-Fronczak, Michel Reniers, Wan Fokkink, and Jacobus Rooda. Structuring multilevel discrete-event systems with dependence structure matrices. *IEEE Transactions on Automatic Control*, 65(4):1625–1639, 2020.

[9] Martijn Goorden, Joanna van de Mortel-Fronczak, Michel Reniers, Martin Fabian, Wan Fokkink, and Jacobus Rooda. Model properties for efficient synthesis of nonblocking modular supervisors. *Control Engineering Practice*, 112:104–830, 2021.

[10] Martijn Goorden, Joanna van de Mortel-Fronczak, Michel Reniers, Wan Fokkink, and Jacobus Rooda. Structuring multilevel discrete-event systems with dependence structure matrices. *IEEE Transactions on Automatic Control*, 65(4):1625–1639, 2019.

[11] Martijn Goorden, Joanna van de Mortel-Fronczak, Michel Reniers, Wan Fokkink, and Jacobus Rooda. Vu research portal. *Discrete Event Dynamic Systems*, 31:317–348, 2021.

[12] Martijn A Goorden, Martin Fabian, Joanna M van de Mortel-Fronczak, Michel A Reniers, Wan J Fokkink, and Jacobus E Rooda. Compositional coordinator synthesis of extended finite automata. *Discrete Event Dynamic Systems*, 31(3):317–348, 2021.

[13] L.E. Holloway and B.H. Krogh. Synthesis of feedback control logic for a class of controlled petri nets. *IEEE Transactions on Automatic Control*, 35(5):514–523, 1990.

[14] Jan Komenda, Tomáš Masopust, and Jan van Schuppen. Control of an engineering-structured multilevel discrete-event system. *Proceedings of International Workshop on Discrete Event Systems*, pages 103–108, 05 2016.

[15] Jan Komenda, Tomáš Masopust, and Jan H van Schuppen. Multilevel coordination control of modular des. In *52nd IEEE Conference on Decision and Control*, pages 6323–6328. IEEE, 2013.

[16] Sahar Mohajerani, Robi Malik, and Martin Fabian. A framework for compositional nonblocking verification of extended finite-state machines. *Discrete Event Dynamic Systems*, 9, 09 2015.

[17] Patrícia N Pena, José ER Cury, and Stéphane Lafortune. Verification of nonconflict of supervisors using abstractions. *IEEE Transactions on Automatic Control*, 54(12):2803–2815, 2009.

[18] Klaus Schmidt and Christian Breindl. Maximally permissive hierarchical control of decentralized discrete event systems. *IEEE Transactions on Automatic Control*, 56(4):723–737, 2011.

[19] Donald V. Steward. The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management*, EM-28:71–74, 1981.

[20] R. Su, Jan Schuppen, and J. Rooda. Aggregative synthesis of distributed supervisors based on automaton abstraction. *IEEE Transactions on Automatic Control*, 55:1627 – 1640, 08 2010.

[21] Rong Su, Jan H Van Schuppen, and Jacobus E Rooda. Maximally permissive coordinated distributed supervisory control of nondeterministic discrete-event systems. *Automatica*, 48(7):1237–1247, 2012.

[22] Stijn van Dongen. Graph clustering by flow simulation. *PhD thesis, University of Utrecht*, 2000.

[23] T. Wilschut, L. F. P. Etman, J. E. Rooda, and I. J. B. F. Adan. Multilevel Flow-Based Markov Clustering for Design Structure Matrices. *Journal of Mechanical Design*, 139(12), 2017.

[24] Kai Wong and Walter Wonham. Modular control and coordination of discrete-event systems. *Discrete Event Dynamic Systems*, 8:247–297, 10 1998.

[25] W Murray Wonham, Kai Cai, et al. Supervisory control of discrete-event systems, 2019.

[26] W Murray Wonham and Peter J Ramadge. On the supremal controllable sublanguage of a given language. *SIAM Journal on Control and Optimization*, 25(3):637–659, 1987.

[27] Junjun Yang, Kaige Tan, Lei Feng, and Zhiwu Li. A model-based deep reinforcement learning approach to the nonblocking coordination of modular supervisors of discrete event systems. *Information Sciences*, 630:305–321, 2023.

[28] Hao Zhong and Walter Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Transactions on Automatic Control*, 35:1125 – 1134, 11 1990.