2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Distributed Supervisor Synthesis for Automated Manufacturing Systems Using Petri Nets

HeSuan Hu[1], *Senior Member*, *IEEE*, Chen Chen[1], Rong Su[2], Yang Liu[3], and MengChu Zhou[4], *Fellow*, *IEEE*

***Abstract*— Due to the competition for limited resources by many concurrent processes in large scale automated manufacturing systems (AMS), one has to resolve a deadlock issue in order to reach their production goal without disruption and downtime. Monolithic resolution is a conventional approach for optimal or acceptable solutions, but suffers from computational difficulty. On the other hand, some decentralized methods are more powerful in finding approximate solutions, but most are application-dependent. By modeling AMS as Petri nets, we develop an innovative distributed control approach, which can create a trajectory leading to a desired destination and are adaptable to different kinds of constraints. Control strategies are applied to processes locally such that they can concurrently proceed efficiently. Global destinations are always reachable through the local observation upon processes without knowing external and extra information. Efficient algorithms are proposed to find such distributed controllers.**

I. Introduction

Manufacturing systems often experience dramatic changes thanks to the emergence of intensive market competition and the availability of advanced automation technology [1]–[4], [6], [9]–[12], [17]. To facilitate a quick response to a market change and allow mass product customization, automated manufacturing systems (AMS) are developed. They consist of a group of numerically-controlled machines, interconnected by a centralized control system, via loading, unloading, storage stations, and automated material handling systems [13]. They are designed so as to rapidly and inexpensively change manufacturing executions on numerous product designs in smaller quantity and with faster delivery. Manufacturers can be benefited by increasing automation, quality, and productivity, reducing cost, waste, and downtime, and authorizing customization, reconfiguration, and flexibility. The avoidance of undesired behaviors in AMS requires the synthesis of a supervisor to deal with all the possible system behavior. However, the design of a centralized supervisor can be too difficult for systems with a large scale due to state explosion [5], [19], [20], [23], [26].

Industrial AMS are featured with large scales. Each global AMS is constituted by collections of relatively small-size, local, interacting, asynchronous, and event-driven subsystems. Its execution is an interactive synchronization of these local subsystems on shared actions. Apparently, as the number of modules increases, state explosion is met when designing a monolithic and centralized supervisor. All monolithic approaches are no longer applicable because of it in the supervisor synthesis procedures [7], [8], [14]–[16].

In the framework of Petri nets, this work intends to avoid the enumerative state-space search when designing and controlling large scale systems. A method is developed to ensure the entire system's nonblockingness by checking local conditions only [7], [18]–[25]. For the sake of scalability, a local system architecture is restricted and thus the global maximal permissiveness cannot be ensured. As a result, one can obtain a suboptimal instead of an optimal solution, but the former is computationally tractable. Despite their considerable complexity, industrial systems exhibit modular structures that can be exploited to avoid explicit subsystem synchronous composition and thereby the state explosion issues. In contrast to the construction of the entire reachability graph, the static analysis of a system configuration can accurately determine the solvability of a nonblocking supervisory control problem. A dynamic search procedure can lead to a supervisor that approximates the maximal permissive solution with nonblockness. The posterior verification of nonblockingness is unnecessary because it is ensured in theory.

This paper is structured as follows. Section II reviews the basic definitions and notations of Petri nets used throughout this paper. Section III is devoted to a special class of Petri nets with a supervisory controller design. In Section IV, with a dynamic search technique, our method is proposed to synthesize a liveness enforcing supervisor in a distributed way. Concluding remarks are given in Section V.

II. Preliminaries

A Petri net is $N = (P, T, F, W)$ where P is a set of places, T is a set of transitions such that $P \cup T \neq \emptyset$, $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is a set of directed arcs

*This work was supported by the Natural Science Foundation of China under Grant Nos. 61203037 and 51305321, Fundamental Research Funds for the Central Universities Under Grant Nos. K5051304004 and K5051304021, New Century Excellent Talents in University under Grant No NCET-12-0921, and NAP under Grant Nos. M4081155.020 and M4080996.020.

[1]Hesuan Hu and Chen Chen are with School of Electro-mechanical Engineering, Xidian University, Xi'an, Shaanxi 710071, P. R. China. Hesuan Hu is also with School of Electrical and Electronic Engineering as well as School of Computer Engineering, Nanyang Technological University, Singapore 639798. hshu@mail.xidian.edu.cn

[2]Rong Su is with School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798. rsu@ntu.edu.sg

[3]Yang Liu is with School of Computer Engineering, Nanyang Technological University, Singapore 639798. yangliu@ntu.edu.sg

[4]MengChu Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, NJ 07102, USA. zhou@njit.edu

from places to transitions or from transitions to places, and $W : (P \times T) \cup (T \times P) \to \mathbb{N} = \{0, 1, 2, \ldots\}$ is the weight function on arcs and $W(x, y) = 0$ if $(x, y) \neq F$. The preset of a node $x \in P \cup T$ is defined as ${}^\bullet x = \{y \in P \cup T \mid (y, x) \in F\}$. Its postset $x^\bullet = \{y \in P \cup T \mid (x, y) \in F\}$. N is a state machine if $W : F \to \{1\}$ and $\forall t \in T$, $|{}^\bullet t| = |t^\bullet| = 1$. It is a marked graph if $W : F \to \{1\}$ and $\forall p \in P$, $|{}^\bullet p| = |p^\bullet| = 1$. N's input incidence matrix $[N^-] = [W(p_i, t_j)]$ and output one $[N^+] = [W(t_j, p_i)]$. Its incidence matrix is $[N] = [N^+] - [N^-]$. $[N_{p_i}]$ (resp., $[N^-_{p_i}]$, $[N^+_{p_i}]$) is the i-th row of $[N]$ (resp., $[N^-]$, $[N^+]$).

A marking of N is a mapping $M : P \to \mathbb{N}$. Given an initial marking M_0, we call (N, M_0) a net system or simply net. t is enabled, denoted by $M[t\rangle$, if $\forall p \in {}^\bullet t$, $M(p) \geq W(p, t)$. M' is reachable from M, denoted by $M[\sigma\rangle M'$, if there exists a firing sequence $\sigma = \langle t_1\ t_2\ \ldots\ t_n\rangle$ such that $M[t_1\rangle M_1\ \ldots\ [t_n\rangle\ M'$. $\overrightarrow{\sigma}$ is a $|T|$-dimensional firing count vector where $\overrightarrow{\sigma}(t)$ states the number of t's appearances in σ. The set of all markings reachable from M_0 is denoted by $R(N, M_0)$. (N, M_0) is bounded if $\exists k \in \mathbb{N}^+ = \mathbb{N} \setminus \{0\}$, $\forall M \in R(N, M_0)$, $\forall p \in P$, $M(p) \leq k$. $t \in T$ is live under M_0 if $\forall M \in R(N, M_0)$, $\exists M' \in R(N, M)$, $M'[t\rangle$ holds. (N, M_0) is deadlock-free if $\forall M \in R(N, M_0)$, $\exists t \in T$, $M[t\rangle$. t is dead at $M \in R(N, M_0)$ if $\nexists M' \in R(N, M)$, $\ni M'[t\rangle$ holds. It is a livelock if $\forall M \in R(N, M_0)$, $\exists \{t, t'\} \subseteq T$, $\ni M[t\rangle$ and t is dead. (N, M_0) is live if $\forall t \in T$, t is live under M_0.

A P- (resp., T-) vector is a column vector $I : P$ (resp., $J : T) \to \mathbb{Z}$ indexed by P (resp., T), where $\mathbb{Z}$ is the set of integers. A P-vector $I \neq \mathbf{0}$ becomes a P-invariant if $[N]^T \cdot I = \mathbf{0}$, where $\mathbf{0}$ means a vector of zeros. A P-invariant is called a P-semiflow if $I \geq \mathbf{0}$. $\|I\| = \{p \in P \mid I(p) \neq 0\}$ is called the support of I. For economy of space, $\sum_{p \in P} M(p) \cdot p$ (resp., $\sum_{p \in P} I(p) \cdot p$, $\sum_{t \in T} J(t) \cdot t$) is used to denote vector M (resp., I, J). (N, M_0) is conservative (resp., consistent) if $\exists I > \mathbf{0}$ (resp., $\exists J > \mathbf{0}$) so that $I^T \cdot [N] = \mathbf{0}^T$ (resp., $[N] \cdot J^T = \mathbf{0}$). A circuit is an ordered set $\langle x_1, x_2, \ldots, x_n\rangle$ such that: 1) $\{x_1, x_2, \ldots, x_n\} \subseteq P \cup T$; 2) $\forall i \in \mathbb{N}_{n-1}$, $x_{i+1} \in x_i^\bullet$; 3) $\forall \{i, j\} \subseteq \mathbb{N}_n$ except $\{i, j\} = \{1, n\}$, $x_i \neq x_j$; and 4) $x_1 = x_n$.

A nonempty set $S \subseteq P$ (resp., $Q \subseteq P$) is a siphon (resp., trap) if ${}^\bullet S \subseteq S^\bullet$ (resp., $Q^\bullet \subseteq {}^\bullet Q$). A strict minimal siphon is a siphon containing neither other siphon nor trap. $M(p)$ indicates the number of tokens in p under M. p is marked by M if $M(p) > 0$. The sum of tokens in S is denoted by $M(S)$, where $M(S) = \sum_{p \in S} M(p)$. A subset $S \subseteq P$ is marked by M if $M(S) > 0$. A siphon is undermarked if $\nexists t \in S^\bullet$ can fire.

Let D be a set. A multiset α over D is defined as a mapping $\alpha : D \to \mathbb{N}$ and can be represented as $\alpha = \sum_{d \in D} \alpha(d) \cdot d$. We denote $Bag(D) = \{\alpha |\ \alpha$ is a multiset of $D\}$. In $Bag(D)$, the following operators are defined. If α, $\alpha' \in Bag(D)$, then: 1) $\alpha \geq \alpha'$ iff $\alpha(d) \geq \alpha'(d)$, $\forall d \in D$; 2) $\alpha + \alpha' = \sum_{d \in D} (\alpha(d) + \alpha'(d)) \cdot d$; 3) $max(\alpha, \alpha') = \sum_{d \in D} (max(\alpha(d), \alpha'(d))) \cdot d$; 4) $\alpha - \alpha' = \sum_{d \in D} (\alpha(d) - \alpha'(d)) \cdot d$. Without abuse of notations, a multi-set can also be denoted as $\sum_{d \in D} \alpha(d)$. This overlaps the economical notation of a vector without causing any confusion.

III. Petri Net Modeling of AMS

For better understanding, we focus throughout this paper on a special class of Petri nets, namely, Feedback System of Sequential Systems with Shared Resources (FS^4R). Nevertheless, this does not necessarily compromise the applicability of our proposed method. In fact, it can be used in more general systems without extra effort. In FS^4R, various job types are modeled by state machines. The availability of various resources is modeled by tokens in their corresponding resource places. Since there is no special limitation upon the resource quantity and types at each operation stage, an FS^4R models a general resource allocation mechanism. Let $\mathbb{N}_L = \{1, 2, \ldots, L\}$ and $\mathbb{N}_K = \{1, 2, \ldots, K\}$. An AMS has a set of resource types $\mathbb{R} = \{r_i, i \in \mathbb{N}_L\}$ and handles a set of process types $\mathbb{J} = \{\mathcal{J}_j, j \in \mathbb{N}_K\}$. Every resource type r_i is further characterized by its capacity $C_i \in \mathbb{N}^+$. Processing requirements of process type $\mathcal{J}_j$ are defined by a set of concurrent and/or sequential stages. Each process stage, say k, modeled by a place p_{jk} is associated with a conjunctive resource requirement, expressed by an m-dimensional vector $a_{p_{jk}}$ with $a_{p_{jk}}[i]$, $i \in \mathbb{N}_L$, indicating how many units of resource r_i are required to support the execution of the stage denoted by p_{jk}.

Definition 1: An FS^4R is a strongly-connected generalized pure Petri net $N = (P, T, F, W)$ where:

1) $P = P_B \cup P_E \cup P_A \cup P_R$ is a partition such that: a) P_B, P_E, P_A, and P_R are called beginning, ending, activity or operation, and resource places, respectively; b) $P_B = \{p_{b_i}, i \in \mathbb{N}_K\}$ and $P_E = \{p_{e_i}, i \in \mathbb{N}_K\}$; c) $P_A = \cup_{i \in \mathbb{N}_K} P_{A_i}$, where for each $i \in \mathbb{N}_K$, $P_{A_i} \neq \emptyset$, and $\forall i, j \in \mathbb{N}_K$, $i \neq j$, $P_{A_i} \cap P_{A_j} = \emptyset$; and d) $P_R = \{r_i, i \in \mathbb{N}_K\}$, $n > 0$;

2) $T = \cup_{i \in \mathbb{N}_K} T_i \cup \{t_{f_i}\}$, where for each $i \in \mathbb{N}_K$, $T_i \neq \emptyset$, and for each $i, j \in \mathbb{N}_K$, $i \neq j$, $T_i \cap T_j = \emptyset$;

3) For each $i \in \mathbb{N}_K$, subnet $\overline{N}^i = N \mid (\{p_{b_i}\} \cup \{p_{e_i}\} \cup P_{A_i}, T_i \cup \{t_{f_i}\}, F_i, W_i)$ is a strongly connected state machine, except for $W(t_{f_i}, p_{b_i}) = W(p_{e_i}, t_{f_i}) = M_0(p_{0_i})$, such that every cycle contains p_{b_i}, p_{e_i}, and t_{f_i};

4) For each $r \in P_R$, $\exists$ a unique minimal P-semiflow $X_r \in \mathbb{N}^{|P|}$ such that $\{r\} = \|X_r\| \cap P_R$, $P_B \cap \|X_r\| = \emptyset$, $P_E \cap \|X_r\| = \emptyset$, $P_A \cap \|X_r\| \neq \emptyset$, and $X_r(r) = 1$; and

5) $P_A = \cup_{r \in P_R} (\|X_r\| \setminus \{r\})$.

In N, $\forall p_0 \in P_0$, $M_0(p_0)$ indicates the maximum number of products that are allowed to be concurrently manufactured in a process initialized by p_0. $\forall p \in P_A$, $M(p) > 0$ means an ongoing operation modeled by p. $\forall r \in P_R$, $M_0(r)$ denotes the capacity of resource type r. From Definition 1, FS^4R is evidently conservative and consistent.

Definition 2: M_0 is an acceptable initial marking in N if (1) $M_0(p_{b_i}) = W(t_{f_i}, p_{b_i})$, $\forall p_{b_i} \in P_B$, $\forall i \in \mathbb{N}_K$; (2) $M_0(p) = 0$, $\forall p \in P_A \cup P_E$; and (3) $M_0(r) \geq X_r(p)$, $\forall r \in P_R$, $\forall p \in P_A$.

Given an arbitrary marking $M \in R(N, M_0)$, a transition t is M-process-enabled if $M({}^\bullet t \cap P_A) > 0$. Note that $|{}^\bullet t \cap P_A| = 1$ by definition. Correspondingly, t is M-resource-

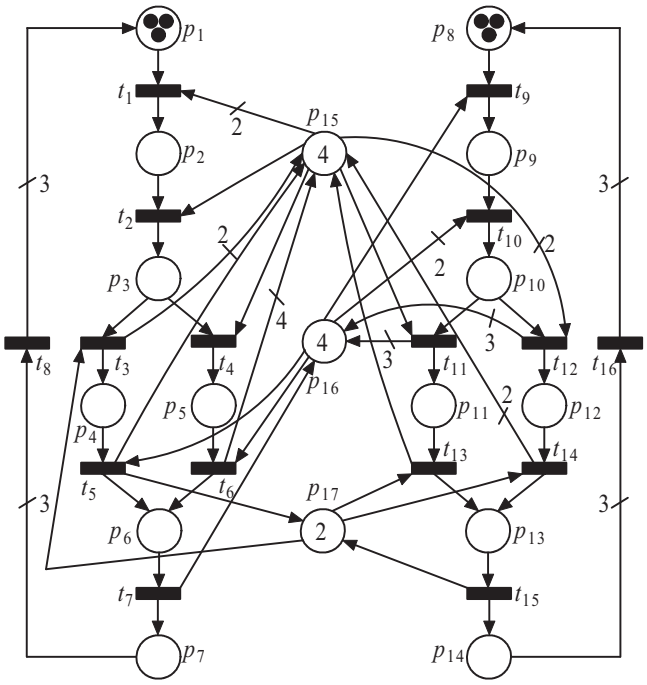

Fig. 1. An example FS^4R net.

enabled by $r \in {}^\bullet t \cap P_R$ if $M(r) \geq W(r, t)$. In the rest of this paper, (N, M_0) is an acceptably marked FS^4R.

Definition 3: $(\overline{N}^i, \overline{M}_0^i)$ is the i-th sub-process net system (process net, for short) such that $|\overline{M}_0^i| = |P_{A_i}| + 2$.

Definition 4: Let $r \in P_R$ be a resource place in (N, M_0). The set of its holders is the support of a minimal P-semiflow X_r without r, i.e., $H(r) = \|X(r)\| \setminus \{r\}$.

Clearly, $H(r)$ contains only operation places due to $\|X(r)\| \cap P_R = \{r\}$. Let $S_R = S \cap P_R$ and $S_A = S \cap P_A$.

Definition 5: Let S be a siphon that can be undermarked in (N, M_0). Its token takers, $\widetilde{H}(S) = \cup_{r \in S_R} H(r) - S$, is the set of the places that correspond to the holders of the resources in S but do not belong to S.

Suppose $H_{S_R} = \cup_{r \in S_R} H(r)$. We have $\widetilde{H}(S) = H_{S_R} \setminus S = (H_{S_R} \cap P_A) \setminus S_A$.

Thanks to their special structure, FS^4R can describe AMS in which each product is manufactured via sequential and/or concurrent manufacturing processes. It is composed of a set of process nets $\overline{N}^i$, $i \in \mathbb{N}_K$, which are in one-to-one correspondence with a product and its related manufacturing processes. More specifically, each $\overline{N}^i$ can be decomposed into an acyclic graph and a feedback transition t_{f_i}. The operations together with their interactions required by a process are represented by the activity places and transitions involved in the respective acyclic graph of $\overline{N}^i$. A set of transitions with a same ingoing place correspond to the initialization of a number of flexible routes, while the ones with a same outgoing places correspond to common termination. The initial marking of a beginning place p_{b_i} corresponds to the number of products that are allowed in the process at a time. Place p_{e_i} is designated as the final destination of all finished process instances. Firing t_{f_i} models the repetitive production of a new batch of process type i. Places in P_R are used to model various resource types. Their marking during the evolution of a Petri net corresponds to the number of available resources in the modeled AMS. In particular, their initial markings define the capacities of the corresponding resource types.

TABLE I
SUPERVISOR FOR PETRI NET IN FIG. 1

i	$M_0(p_c)$	${}^\bullet p_c$
		$p_c^\bullet$
1	8	$\{2 \cdot t_3, 2 \cdot t_5, 4 \cdot t_6, 2 \cdot t_{11}, t_{12}, t_{13}, 2 \cdot t_{14}\}$
		$\{2 \cdot t_1, t_2, t_4, t_9, 2 \cdot t_{10}\}$
2	6	$\{2 \cdot t_2, 3 \cdot t_3, 6 \cdot t_4, 3 \cdot t_{11}, 3 \cdot t_{12}\}$
		$\{2 \cdot t_2, t_3, 4 \cdot t_4, t_9, 2 \cdot t_{10}\}$
3	2	$\{2 \cdot t_2\}$
		$\{t_2\}$
4	2	$\{2 \cdot t_{11}, 2 \cdot t_{12}\}$
		$\{t_9, t_{10}\}$

Under the assumption that $P_B = \{p_1, p_8\}$, $P_E = \{p_7, p_{14}\}$, $P_{A_1} = \{p_2 - p_6\}$, $P_{A_2} = \{p_9 - p_{13}\}$, $P_R = \{p_{15} - p_{17}\}$, $t_{f_1} = t_8$, and $t_{f_2} = t_{16}$, Fig. 1 shows an FS^4R example representing an AMS consisting of three resource types $r_1 - r_3$ with capacities $C_1 = C_2 = 4$, and $C_3 = 2$, and handling job types $\mathcal{J}_1$ and $\mathcal{J}_2$. $\mathcal{J}_1$ (resp., $\mathcal{J}_2$) is defined by a set of partially ordered job stages $\{p_1 - p_7\}$ (resp., $\{p_8 - p_{14}\}$). The conjunctive resource requirements associated with various job stages are as follows: $a_{p_2} = 2 \cdot p_{15}$, $a_{p_3} = 3 \cdot p_{15}$, $a_{p_4} = p_{15} + p_{17}$, $a_{p_5} = 4 \cdot p_{15}$, $a_{p_6} = p_{16}$, $a_{p_9} = p_{16}$, $a_{p_{10}} = 3 \cdot p_{16}$, $a_{p_{11}} = p_{15}$, $a_{p_{12}} = p_{15}$, and $a_{p_{13}} = p_{17}$. The P-semiflows corresponding to the resources are: $X_1 = 2 \cdot p_2 + 3 \cdot p_3 + p_4 + 4 \cdot p_{15} + p_{11} + p_{12} + p_{15}$, $X_2 = p_6 + p_9 + 3 \cdot p_{12} + p_{16}$, and $X_3 = p_4 + p_{13} + p_{17}$. There are two sub-process net systems, i.e., $(\overline{N}^1, \overline{M}_0^1)$ and $(\overline{N}^2, \overline{M}_0^2)$. $(\overline{N}^1, \overline{M}_0^1)$ is composed by $\{p_1 - p_7\}$ and $\{t_1 - t_8\}$ along with their related arcs. $\overline{M}_0^1 = 3 \cdot p_1$. $(\overline{N}^2, \overline{M}_0^2)$ is composed by $\{p_8 - p_{14}\}$ and $\{t_9 - t_{16}\}$ along with their related arcs. $\overline{M}_0^2 = 3 \cdot p_8$. Obviously, this Petri net model allows multiple units of resource acquisition at each operation stage as well as flexible routes.

Siphon-based liveness analysis is crucial for most Petri net models; however, it is of difficulty to derive such structural objects owing to their complexity. In this work, we simplify this problem by restricting the behavior of FS^4R to trajectories ensuring the liveness and deadlock-freeness.

Take the net shown in Fig. 1 as an example. Totally, there are four siphons, i.e., $S_1 = \{p_6, p_{13}, p_{15} - p_{17}\}$, $S_2 = \{p_3, p_6, p_{11}, p_{12}, p_{15}, p_{16}\}$, $S_3 = \{p_3 - p_5, p_{11}, p_{12}, p_{15}\}$, and $S_4 = \{p_6, p_{10}, p_{16}\}$. To avoid the undermarkedness of each siphon, a general exclusive mutual constraint (GMEC) is applied to its complementary part. After a GMEC is given, a monitor p_c along with its outgoing and ingoing arcs as well as initial marking can be calculated with the technique in [2]. Table I shows the resultant supervisor that is a simple integration of all monitors. Obviously, such a supervisor is

derived in a global way as searching the entire net for all siphons is necessary. Moreover, the control mechanism is also implemented in a global way because it works exactly as a coordinator dispatching all resources to different processes according to certain predetermined rules. As the system size increases, the number of siphons increases exponentially. Thus, such a design and control method may fail in large scale systems. A distributed solution becomes necessary.

IV. Distributed Control of FS^4R

Given an FS^4R, its evolution represents the flow of tokens from the beginning places to destination ones. The nonblockingness property is guaranteed through undermarkedness of no siphons, which can be realized with conventional static supervision techniques. Structure-based and global specifications are involved to avoid the siphons' undermarkedness in a centralized and global way. All resources are assumed reliable; thereby, systems are not contingency-prone. Failures may appear in practical systems whose structures are distributed and behaviors are dynamic. Challenging issues are broken-down resources. It is questionable regarding the adaptability of a conventional static, global, centralized, and failure-sensitive mechanism. In practice, supervisors are expected to be robust, adaptive, and tolerant to various contingencies including resource failure and configuration modification. As a consequence, deadlock resolution techniques must be dynamic and fault-tolerant. Instead of checking the global structures, the proposed approach aims to resolve the problem in a decentralized and local way. Rather than failure sensitive, the resultant systems become fault-tolerant.

A. Distributed Admissibility-enforcing Control

Given an FS^4R, each of process $\overline{N}^i$ is executed independently without the constraints from resources. As a result, the state $M \in R(N, M_0)$ is not necessarily a Cartesian state product of different processes. Moreover, deadlock may occur owing to the competition among processes for finite resources, further reducing the number of states from the set of the Cartesian product. Take the Petri net in Fig. 1 as an example. Without the resource constraint, the two sub-process net systems and $(\overline{N}^2, \overline{M}_0^2)$, involve 84 states, respectively. Their Cartesian product is $84 \times 84 = 7056$. After considering the resource constraints, the number of state is reduced to 2076. For example, although $3 \cdot p_2 + 2 \cdot p_{10}$ is a component of the Cartesian product, its reachability is impossible because it requires 6 copies of r_1 and 6 copies of r_2 while either has its capacity being bounded by 4.

As known, the computation of the set of reachable states is an NP-complete hard problem as one ought to enumerate all states and determine their reachability one by one. This approach intends to avoid such an enumeration. To achieve so, we introduce the concept of admissibility, which approximates the reachability but significantly reduces the computational complexity.

Definition 6: Given an FS^4R (N, M_0), $\mathcal{M}$ is a admissible state if $\forall r \in P_R$, $\sum_{p \in P_A \cap \| X_r \|} X_r(p) \cdot \mathcal{M}(p) \leq M_0(r)$. $\mathcal{R}(N, M_0)$ denotes the universal set of $\mathcal{M}$.

Apparently, regarding a state's admissibility, the only criterion is the usage of resources. In the sequel, $\mathcal{M}$ is used to denote a component in the set of admissible states $\mathcal{R}(N, M_0)$. The admissibility property must hold for each component belonging to the set of reachable states; however, an admissible state does not necessarily belong to the set of reachable states. In short, we have $R(N, M_0) \subseteq \mathcal{R}(N, M_0)$. Thus, the below result is true.

Proposition 1: Given an FS^4R (N, M_0), $M \in R(N, M_0)$. We have $M \in \mathcal{R}(N, M_0)$.

Proposition 1 implies that the admissible state set can well cover the reachable state set. The former one is quite easy to produce and verify because we only need to check the availability of various resources. As a benefit, we can neglect the entire system's global structure. In the sequel, we introduce the notion of successor and the algorithm to produce the admissible graph. Since the enabledness condition is substituted by the admissibility condition, the algorithm becomes concise and a superset of the reachable state set is obtained.

Definition 7: $\mathcal{P}(\mathcal{M})$ is an activity distribution function, to map each admissible state to a vector of activities in progress at this state. Moreover, $\mathcal{P}_i(\mathcal{M})$ is an activity distribution function at P_{A_i} to map each admissible state to a vector of activities in progress in P_{A_i}. Thus, we also denote $\mathcal{P}(\mathcal{M})$ as $[\mathcal{P}_1(\mathcal{M})^T \ \mathcal{P}_2(\mathcal{M})^T \ \ldots \ \mathcal{P}_K(\mathcal{M})^T]^T$.

Definition 8: $\mathscr{P}(\mathcal{M})$ is an activity distribution function to map each admissible state to a multi-set of activities in progress at this state. Moreover, $\mathscr{P}_i(\mathcal{M})$ is an activity distribution function at P_{A_i} to map each admissible state to a multi-set of activities in progress at process P_{A_i}. Thus, we also denote $\mathscr{P}(\mathcal{M})$ as $\{\mathscr{P}_1(\mathcal{M}), \mathscr{P}_2(\mathcal{M}), \ldots, \mathscr{P}_K(\mathcal{M})\}$.

Definition 9: $\mathscr{R}(\mathcal{M})$ (resp., $\mathscr{R}^{-1}(\mathcal{M})$) is a resource utilization (resp., availability) function to map each admissible state to a multi-set of resources utilized (resp., available) at this state.

Proposition 2: Given an FS^4R, we have $\mathscr{R}(\mathcal{M}) + \mathscr{R}^{-1}(\mathcal{M}) = \mathscr{R}^{-1}(\mathcal{M}_0)$.

Definition 10: $\mathscr{M}(\mathcal{M})$ is a state function to map each admissible state to a multi-set.

Proposition 3: $\mathscr{M}(\mathcal{M}) - \mathscr{P}(\mathcal{M}) + \mathscr{R}(\mathcal{M}) = \mathscr{R}^{-1}(\mathcal{M}_0)$.

Definition 11: Given an FS^4R (N, M_0), $\forall \{\mathcal{M}, \mathcal{M}'\} \subset \mathcal{R}(N, M_0)$, $\mathcal{M}'$ is said to be the successor of $\mathcal{M}$, which is denoted as $\mathcal{M} \stackrel{t_{i_j}}{\rightarrow} \mathcal{M}'$ where $i \in \mathbb{N}_K$ and $j \in \mathbb{N}_{|P_{A_i}|}$ such that $\mathcal{P}_i(\mathcal{M}) \in \mathcal{R}(\overline{N}^i, M_0^i)$ and $\mathcal{P}_i(\mathcal{M}') \in \mathcal{R}(\overline{N}^i, M_0^i)$, $\mathcal{P}_i(\mathcal{M}) \ [\ t_{i_j} \ \rangle \ \mathcal{P}_i(\mathcal{M}')$, $\mathcal{P}_{k \neq i}(\mathcal{M}) \ [\ t_{i_j} \ \rangle \ \mathcal{P}_{k \neq i}(\mathcal{M}')$ where t_{i_j} means the j-th transition in the i-th process.

In the sequel, an algorithm is developed to define the structure of the distributed controllers.

Algorithm 1: Generation of the Admissible Global Reachability Set and its Successor Relationship

Input: An FS^4R (N, M_0) constituted by $(\overline{N}^i, \overline{M}_0^i)$ and $\{r_j\}$ such that $i \in \mathbb{N}_K$ and $j \in \mathbb{N}_L$

Output: All the Admissible Global States and their Successor Relationship

begin

1) $\mathcal{R}(N, M_0)$ is the set of admissible global states with $\mathcal{M} \in \mathcal{R}(N, M_0)$ as a component;
2) $\mathcal{M}_0 = M_0 \in \mathcal{R}(N, M_0)$ is the initial state;
3) $\mathcal{M} \stackrel{t_{i_j}}{\rightarrow} \mathcal{M}'$ iff $\mathcal{M}'$ is an admissible global successor of $\mathcal{M}$ such that $\mathcal{P}_i(\mathcal{M}) \stackrel{t_{i_j}}{\rightarrow} \mathcal{P}_i(\mathcal{M}')$, where $i \in \mathbb{N}_K$, $j \in \mathbb{N}_{|P_{A_i}|}$. Moreover, $\forall r \in P_R$, $X_r^T \cdot M \leq M_0(r)$ and $X_r^T \cdot M' \leq M_0(r)$;

end

The above algorithm defines the structure of a controller. Actually, it successfully merges the behavior of both local controllers and global coordinator. The system evolution well behaves according to the local evolution feasibility and global admissibility. More importantly, the last item can be utilized to define various behavior constraints upon the system evolution which is actually a proper subset of $\mathcal{R}(N, M_0)$. The generation of $R(N, M_0)$ requires the verification at each state the nonblocking issues and enabledness of all transitions. The above algorithm can well avoid the such time-consuming verification operations as all states are represented on the basis of set theory and logic operations.

Given an FS^4R, the obtained behavior is no longer in the form of a plant model and some separated controllers. In fact, it is realized by one or several possible trajectories, i.e., $\mathcal{M}_0 \stackrel{t_{z_1}}{\rightarrow} \mathcal{M}_{z_1} \stackrel{t_{z_2}}{\rightarrow} \mathcal{M}_{z_2} \stackrel{t_{z_3}}{\rightarrow} \ldots \mathcal{M}_{z_{k-1}} \stackrel{t_{z_k}}{\rightarrow} \mathcal{M}_{z_k} \ldots$ with $z_i = x_i y_i$ where $x_i \in \mathbb{N}_K$, $y_i \in \mathbb{N}_{max\{|P_{A_{x_i}}|\}}$. In the sequel, this trajectory can be simplified as $\mathcal{M}_0 \stackrel{\sigma}{\rightarrow} \mathcal{M}_{z_k} \ldots$, where $\sigma = \langle t_{z_1}\ t_{z_2}\ \ldots\ t_{z_k} \rangle$.

Compared with any other properties, liveness is of paramount significance. This is because only it can ensure all processes to reach their destinations without any blocking issues. In the sequel, it is realized through model predictive control techniques. This means that before the execution of each step, we must ensure that the corresponding process can reach their destination without any blocking issue.

According to the above statement, we need to establish a distributed algorithm to determine the feasibility of each step's execution. As known, it can be quite time-consuming to search a deadlock free trajectory in the entire reachability graph. In the case where the limited-step look-ahead technique is adopted, the upper bound of the step limitation is actually undecidable in general cases and thus deadlock state cannot be completely removed in theory. Moreover, there is no unified principle to follow such that we can produce an effective and efficient look-ahead technique. Next, we present an efficient method to tackle the issue.

B. Distributed Design of Liveness-enforcing Controller

The above discussions suggest that an admissible state is unnecessarily an either good or real one. First, it might be a livelock or deadlock which involves a circular wait among several processes occupying certain resources. Second, it might be a pseudo state which is never reachable owing to the negligence of some transitions' fireability. As a consequence, a supervisor is expected to simultaneously remove livelock, deadlock, and pseudo states. Take the Petri net in Fig. 1 as an instance. A firing sequence, say $\langle t_1\ t_2\ t_9\ t_9\ t_9\ t_9 \rangle$, results in a marking $p_1 + p_2 + 3 \cdot p_9 + p_{16} + 2 \cdot p_{17}$, under which a siphon $S = \{p_3, p_6, p_{11}, p_{12}, p_{15}, p_{16}\}$ is undermarked and no process can proceed despite the existence of some available resources.

Conventional methods necessitate the supervisory control of all siphons and/or bad states in order to synthesize supervisors. Due to their astrometrical quantity, our strategy pursues for their avoidance. As an alternative, we propose a method with the aid of a look-ahead technique. Priori to each step's execution along its process, the other processes temporarily stagnate so as to yield for its accomplishment. In other words, our focus is upon an arbitrarily or intentionally selected process rather than others. In this particular process, a predictive evaluation is conducted upon a token to determine whether it can proceed from its current location to the destination place or a place with the maximum resource usage. If so, this token is moved to its next step; otherwise, it keeps standstill regardless its legality to further proceed. Thanks to their significance, critical places stand as the notion for these destination places and the places with the maximum resource usage.

Definition 12: Given an FS^4R, $\mathscr{C} \subseteq \cup_{i \in \mathbb{N}_K} P_{A_i} \cup P_B \cup P_E$ are a set of critical places if $a_p = \mathbf{0}^T$, $\forall p \in P_A \cup P_B \cup P_E$ or $\forall i \in \mathbb{N}_K$, $a_p \geq a_{p'}$, $\forall p' \in P_{A_i} \cup P_B \cup P_E$.

At an abstract level, the above definition provides a distributed method to produce a firing sequence which ensures nonblockingness. Apparently, such firing sequences are not unique thanks to the selection randomness upon the imminently progressing process. A different selection sequence or criterion results in a distinct firing sequence and control scheme. Any job is of independence and capability to navigate the entire system from the initial state to the destination one. From the viewpoint of pure logic, the increase of concurrency usually improves the system performance. Eventually, the following are some remarks upon the desired control strategy.

First, places except resources are distinguished by critical and normal ones. The former are the places with minimum (zero, in fact) or maximum resource usages whereas the latter are the remaining places. From the viewpoint of resource acquisition, enough resources are available behind the place with minimum resource usage and no more resources are required behind the place with maximum resource usage. Critical places' existence is guaranteed because no resource is occupied at the beginning and destination places. By intuition, they behave like temporary buffers for the processes.

Second, systems' tractability is ensured by their initial markings. When $\forall j \in \mathbb{N}_K \setminus \{i\}$ processes stagnate at their current states, the i-th one can reach the closest critical place. In the worst case, all the processes except the i-th one stagnate at their initial states, the i-th one can reach its destination place according to Definition 2. After so, all resources are released such that another token in the same process or another process can proceed similarly. Eventually, all tokens can reach their destination places after these two steps execute repeatedly.

Third, a liveness-enforcing supervisor is promised to exist

by FS^4R's special system structure and its acceptable initial marking. On the basis of the above statements, one can easily establish a liveness enforcing supervisory control scheme. Before it leaves its current state, every process must guarantee the reachability of the closest critical place. A prominent principle is that only one step is allowed for any process execution when it leaves its current state and proceeds towards its destination. This philosophy is valid even when sufficient resources are available to support certain token in a process to reach its destination place.

At any system's initial marking, all processes are at their critical places because only the beginning places are marked and no resources are employed. By using the above-mentioned predictive control technique, all processes proceed alternatively and concurrently from their beginning places to destination ones. To the end, a trajectory is produced which can accomplish such an objective. Superficially, our strategy is similar to the lookahead deadlock resolution policy. Nevertheless, they are different to much extent. For the former, the prerequisite for its applicability is the generation of the entire reachability graph. Owing to the undecidability of each path's length, there is no way either precisely or approximately to anticipate the number of look-ahead steps. Our method requires no information from the reachability graph. Comparatively, the look-ahead step is beforehand determined by each process' length, which promises the efficiency of a deadlock-free trajectory's identification.

For each searching process, one and only one trajectory can be produced. As a consequence, this method does not ensure the controlled system's maximal permissiveness. Note that maximal permissiveness is of only theoretical significance for a large-scale system. It provides only all the potential and feasible trajectories along which a system may follow. No information is provided to reach the destination along the shortest path or with the minimum duration. As a consequence, system performance may not benefit from the so-called maximal permissiveness. In contrast, our method dynamically provides an effective criterion ensuring the system's liveness during its real-time evolution. It may be somewhat less permissive owing to the prohibition of some trajectories leading to no deadlocks.

Definition 13: Given an FS^4R, $\mathscr{C}$ is a set of critical places. $\forall p \in P_B \cup P_E \cup P_A$, its neighbor is defined as a set $\mathscr{N}(p) = \{p_{i_0}\ p_{i_1}\ \ldots\ p_{i_k}\} \subseteq P_B \cup P_E \cup P_A$ such that $\langle p_{i_0}\ t_{i_1}\ p_{i_1}\ \ldots\ t_{i_k}\ p_{i_k}\rangle$ is a path where $p_{i_0} = p$, $p_{i_k} \in \mathscr{C}$, and $\forall j \in \mathbb{N}_{l-1}$, $p_{i_j} \notin \mathscr{C}$.

Definition 13 determines that $\mathscr{N}(p)$ contains a series of places which starts and ends with two distinct critical places. Apparently, these neighbors partition all processes into segments which are uninterruptedly connected with critical places as their joints. Our control strategy is to transmit each token along these segments sequentially. Between any two contiguous segments, a critical place performs like an interchange station under which a token can sojourn temporarily to ascertain further progress' feasibility and determine the next move's direction. These segments behave as the bridges carrying each token from one critical place to another.

Definition 14: Given an FS^4R (N, M_0), $\{\mathcal{M}, \mathcal{M}'\} \subseteq \mathcal{R}(N, M_0)$, $\mathcal{M} \stackrel{t_{i_j}}{\rightarrow} \mathcal{M}'$, $\exists\{p\} = {}^\bullet t_{i_j} \cap \{P_B \cup P_E \cup P_A\}$, $\mathcal{M}(p) \geq 1$. $\exists\{p'\} = t_{i_j}^\bullet \cap \{P_B \cup P_E \cup P_A\}$, $\mathcal{M}(p') \geq 1$. p's neighbor is $\mathscr{N}(p) = \{p_{i_j}\ p_{i_{j+1}}\ \ldots\ p_{i_{j+k}}\} \subseteq P_B \cup P_E \cup P_A$ associated to a path $\langle p_{i_j}\ t_{i_j}\ p_{i_{j+1}}\ \ldots\ t_{i_{j+k+1}}\ p_{i_{j+k}}\rangle$ where $p_{i_j} = p$, $p_{i_{j+1}} = p'$, $p_{i_{j+k}} \in \mathscr{C}$, and $\forall l \in \mathbb{N}_{k-1}$, $p_{i_{j+l}} \notin \mathscr{C}$. $\mathscr{M} = \{\mathcal{M}^{<0>}, \mathcal{M}^{<1>}, \ldots, \mathcal{M}^{<k>}\}$ is $\mathcal{M}$'s Operation Downstream Neighborhood (ODN) such that $\mathcal{M}^{<0>} = \mathcal{M}'$, $\mathcal{M}^{<0>} \stackrel{t_{i_{j+2}}}{\rightarrow} \mathcal{M}^{<1>} \stackrel{t_{i_{j+3}}}{\rightarrow}, \ldots, \stackrel{t_{i_{j+k+1}}}{\rightarrow} \mathcal{M}^{<k>}$.

In the framework of FS^4R, $\mathscr{C}$ can be distinguished into two categories. The first one is the beginning and destination places under which no resources are occupied. The second one is the places under which maximum resources are occupied. In the former case, sufficient resources are available to transmit a token in an arbitrarily selected process to its destination place when all the other processes are in halt. In the latter case, no more resources are required because the current place already acquires the maximum number of resources. Under either situation, a common feature is that a token in one process can proceed to its closest critical place. Hence, our control strategy can be described as follows.

Algorithm 2: Generation of a Liveness-enforcing Supervisor in Form of a Firing Sequence
Input: An FS^4R (N, M_0)
Output: A Series of Live Global States and their Successor Relationship
begin

1) $\mathcal{L}(N, M_0)$ is a set of admissible global states;
2) $\mathcal{M}_0 \in \mathcal{L}(N, M_0)$ is the initial state;
3) $\mathcal{M} \stackrel{t_{i_j}}{\rightarrow} \mathcal{M}'$ is a liveness-enforcing successor relationship if:
   a) $\{\mathcal{M}, \mathcal{M}'\} \subseteq \mathcal{R}(N, M_0)$. $\mathcal{M} \stackrel{t_{i_j}}{\rightarrow} \mathcal{M}'$ is an admissible global successor such that $i \in \mathbb{N}_K$ and $j \in \mathbb{N}_{|P_{A_i}|}$;
   b) For $\forall \mathcal{M}^{<k>} \in \mathscr{M}$, $\mathscr{R}(\mathcal{M}^{<k>}) + \mathscr{R}(\mathcal{M}) \leq \mathscr{R}^{-1}(\mathcal{M}_0)$, where $\mathscr{M}$ is $\mathcal{M}'$s ODN and $k \in \mathbb{N}_{|\mathscr{M}|}$.

end

This supervisor synthesis algorithm has two features. First, it works in a decidable predictive way. One and only one step is allowed to execute along a process even if sufficient resources are available to consecutively transfer a token to its corresponding destination place. Sufficient resources are available for this process' progress if all other processes halt during this execution step. Second, the algorithm proceeds with no need to check the global information. Each step's execution is determined by the sufficiency of its related resources. In other words, there is no need to know the status of all the other processes except the one to execute. As a consequence, the entire algorithm performs in a completely distributed way since only local information is required. This implies a significant decrease in the communication amount between the supervisor and processes. In the sequel, the correctness of this algorithm is established.

Lemma 1: Given an FS^4R, $\mathcal{M}_0 = M_0 \in \mathcal{R}(N, M_0)$ and $\Omega = \cup_{i=1}^{|\Omega|}\{\mathcal{M}\}$ such that $\mathcal{M}_0 \xrightarrow{\sigma} \mathcal{M}$ holds according to Algorithm 2. Then, we have $\Omega \subseteq \mathcal{R}(N, M_0)$.

Lemma 2: Given an FS^4R, $\mathcal{M}_0 = M_0 \in \mathcal{R}(N, M_0)$ and $\Omega = \cup_{i=1}^{|\Omega|}\{\mathcal{M}\}$ such that $\mathcal{M}_0 \xrightarrow{\sigma} \mathcal{M}$ holds according to Algorithm 2. Then, we have $\Omega \subseteq R(N, M_0)$.

Lemma 3: Given an FS^4R, $\mathcal{M}_0 = M_0 \in \mathcal{R}(N, M_0)$ and $\Omega = \cup_{i=1}^{|\Omega|}\{\mathcal{M}\}$ such that $\mathcal{M}_0 \xrightarrow{\sigma} \mathcal{M}$ holds according to Algorithm 2. Then, $\nexists t \in \sigma$, t is dead.

Theorem 1: Given an FS^4R, $\mathcal{M}_0 = M_0 \in \mathcal{R}(N, M_0)$ and $\Omega = \cup_{i=1}^{|\Omega|}\{\mathcal{M}\}$ such that $\mathcal{M}_0 \xrightarrow{\sigma} \mathcal{M}$ holds according to Algorithm 2. The synthesized supervisor is liveness-enforcing.

To achieve higher concurrency, each token is associated with an identity number. When there are G number of tokens in total, each can be represented by a label $o(x)^{<y_x>}$ where $x \in \mathbb{N}_G$ means a token's identity number and $y_x \in max_{i\in\mathbb{N}_K}\{|P_{A_i}|\}$ means how many steps the x-th token has moved forward. At M_0, $\forall x \in \mathbb{N}_G$, we have $y_x = 0$. Once the x-th token proceeds for one step, $y_x := y_x + 1$. More concurrency implies that more tokens proceed with the same pace. In another words, between any two tokens, i.e., x_i and x_j, their largest difference must be minimized. In the optimality theory, this corresponds to an objective function $min\{max|y_{x_i} - y_{x_j}|\}$ where $\{x_i, x_j\} \subseteq \mathbb{N}_G$.

Consider the FS^4R as an example. With the aid of Algorithm 2, we can easily derive a liveness-enforcing supervisor which is represented as a firing sequence, i.e., $\langle t_1\ t_9\ t_2\ t_{10}\ t_3\ t_{11}\ t_5\ t_{13}\ t_7\ t_{15}\ t_1\ t_9\ t_2\ t_{10}\ t_4\ t_{12}\ t_6\ t_{14}\ t_7\ t_{15}\ t_1\ t_9\ t_2\ t_{10}\ t_3\ t_{12}\ t_5\ t_{13}\ t_7\ t_{15}\ t_8\ t_{16}\rangle$. Compared with the supervisor in Table I, our supervisor does not need any global information during its synthesis and execution processes.

V. Conclusion

This work focuses on the synthesis of distributed liveness-enforcing supervisors of automated manufacturing systems allowing both flexible process routes and multiple resource acquisition operations. Deadlocks can be avoided in a dynamical way and distributed manner. The admissibility of each execution step is determined by actively checking the availability of sufficient resources in a local way rather than by passively accepting the instructions of execution or prohibition from a supervisor in a global way. As a consequence, the communication amount decreases substantially because the execution of one step is completely independent from the statuses of other processes. All siphons are implicitly controlled and all deadlocks are avoided. This technique can reduce the computation burden to compute these supervisor structures. Our resulting supervisor is simple, distributed, dynamical, adaptive, and fault-tolerant. The generalization of the research results to more complex systems is expected.

References

[1] A. Aybar, "Decentralized structural control approach for Petri nets," *Control Cybern.*, vol. 36, no. 1, pp. 143–159, Jan. 2007.

[2] K. Barkaoui and I. Abdallah, "A deadlock prevention method for a class of FMS," in *Proc. IEEE Int. Conf. Syst., Man, and Cybern.*, pp. 4119–4124, Vancouver, British Columbia, Canada, Oct. 1995.

[3] F. Basile, A. Giua, and C. Seatzu, "Some new results on supervisory control of Petri nets with decentralized monitor places," in *Proc. Intern. Fed. Autom. Contr.*, Seoul, Korea, Jul. 2008, pp. 531–536.

[4] F. Basile, A. Giua, and C. Seatzu, "Decentralized supervisory control of Petri nets with monitor places," in *Proc. IEEE Conf. Autom. Sci. Eng.*, Arizona, USA, Sept. 2007, pp. 7–13.

[5] P. Darondean, "Distributed implementations of Ramadge-Wonham supervisory control with Petri nets," in *Proc. IEEE Conf. Dec. Contr.*, Seville, Spain, Dec. 2005, pp. 2017–2112.

[6] J. Ezpeleta, J. Colom, and J. Martínez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. Robot. Autom.*, vol. 11, no. 2, pp. 173–184, Apr. 1995.

[7] R. Ferrari, T. Parisini, and M. Polycarpou, "Distributed fault detection and isolation of large-scale discrete-time nonlinear systems: An adaptive approximation approach," *IEEE Trans. Auto. Contr.*, vol. 57, no. 2, pp. 275–290, Feb. 2012.

[8] M. Franceschelli, A. Giua, and C. Seatzu, "Distributed averaging in sensor networks based on broadcast gossip algorithms," *IEEE Sens. Journ.*, vo. 11, no. 3, pp. 808–817, Mar. 2011.

[9] H. S. Hu, M. C. Zhou, and Z. W. Li, "Liveness enforcing supervision of video streaming systems using non-sequential Petri nets," *IEEE Trans. Multi.*, vol. 11, no. 8, pp. 1457–1465, Nov. 2009.

[10] H. S. Hu, M. C. Zhou, and Z. W. Li, "Algebraic synthesis of timed supervisor for automated manufacturing systems using Petri nets," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 3, pp. 549-557, Jul. 2010.

[11] H. Hu, M. C. Zhou, Z. W. Li, and Y. Tang, "Deadlock-free control of automated manufacturing systems with flexible routes and assembly operations using Petri nets," *IEEE Trans. Ind. Inform.*, Vol. 9, No. 1, pp. 109–121, Feb. 2013.

[12] H. Hu and Y. Liu, "Supervisor simplification for AMS based on Petri nets and inequality analysis," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 1, pp. 66-77, Jan. 2014.

[13] M. Iordache and P. Antsaklis, "Synthesis of supervisor enforcing general linear constraints in Petri nets," *IEEE Trans. Autom. Control*, vol. 48, no. 11, pp. 2036–2039, Nov. 2003.

[14] G. Jiroveanu, R. Boel, and B. Bordbar, "On-line monitoring of large Petri net models under partial observation," *Discrete Event Dyn. Sys.: Theor. and Appl.*, vol. 18, no. 3, pp. 323–354, Sept. 2008.

[15] J. Julvez and R. Boel, "A continuous Petri net approach for model predictive control of traffic systems," *IEEE Trans. Syst., Man, Cybern. A. Syst., Humans*, vol. 40, no. 4, pp. 686–697, Jul. 2010.

[16] J. Komenda, T. Masopust, and J. H. V. Schuppen, "Supervisory control synthesis of discrete-event systems using a coordination scheme," *Automatica*, vol. 48, no. 2, pp. 247–254, Feb. 2012.

[17] Z. W. Li and M. C. Zhou, Deadlock Resolution in Automated Manufacturing Systems: A Petri net Approach, Springer, 2009.

[18] D. Perez and H. Barbera, "Modelling distributed transportation systems composed of flexible automated guided vehicles in flexible manufacturing systems," *IEEE Trans. Ind. Inform.*, vol. 6, no. 2, pp. 166–180, May 2010.

[19] R. Su, J. H. V. Schuppen, and J. Rooda, "Maximally permissive coordinated distributed supervisory control of nondeterministic discrete-event systems," *Automatica*, vol. 48, no. 7, pp. 1237–1247, Jul. 2012.

[20] R. Su, J. H. V. Schuppen, and J. Rooda, "Aggregative synthesis of distributed supervisors based on automaton abstraction," *IEEE Trans. Autom. Contr.*, vol. 55, no. 7, pp. 1627–1640, Jul. 2010.

[21] R. Su, J. H. V. Schuppen, and J. E. Rooda, "The synthesis of time optimal supervisors by using heaps-of-pieces," *IEEE Trans. Autom. Contr.*, vol. 57, no. 1, pp. 105–118, Jan. 2012.

[22] R. Su, J. H. V. Schuppen, J. E. Rooda, and A. T. Hofkamp, "Nonconflict check by using sequential automaton abstractions based on weak observation equivalence," *Automatica*, vol. 46, no. 6, pp. 968–978, Jun. 2010.

[23] K. Schmidt and C. Breindl, "Maximally permissive hierarchical control of decentralized discrete event systems," *IEEE Trans. Autom. Contr.*, vol. 56, no. 4, pp. 723–737, Apr. 2011.

[24] C. R. Vazquez, J. H. V. Schuppen, and M. Silva, "A modular-coordinated control for continuous Petri nets," in *Proc. IFAC World Congress*, Milano, Italy, Aug. 2011, pp. 6029–6035.

[25] L. Wang, C. Mahulea, J. Julvez, and M. Silva, "Minimum-time decentralized control of choice-free continuous," *Nonlinear Analysis: Hybrid Sys.*, vol. 7, no. 1, pp. 39–53, Jan. 2013.

[26] A. Vahidi, M. Fabian, and B. Lennartson, "Efficient supervisory synthesis of large systems," *Control Eng. Pract.*, vol. 14, no. 10, pp. 1157–1167, Oct. 2006.