Çankaya University Journal of Science and Engineering
Volume 16, No. 2 (2019) 070–086
Date Received: October 22, 2019
Date Accepted: October 28, 2019

CUJ

Order-preserving Models for the Supervisory Control of Flexible Manufacturing Systems

Anas Nooruldeen[1], Klaus Werner Schmidt[2]

[1]*Department of Electronic and Communication Engineering, Çankaya University, Turkey,*
[2]*Department of Electrical and Electronics Engineering, Middle East Technical University, Turkey,*
e-mail: anasnooruldeen@gmail.com, schmidt@metu.edu.tr

Abstract: Flexible manufacturing systems (FMS) are characterized by the processing of different product types on the same manufacturing systems. In particular, it is possible that the paths of different products type through an FMS overlap and different product types share the same production components such as machines.

This paper develops a new modeling technique for the supervisory control of FMS in the framework of discrete event systems (DES). In particular, we consider the general case of an FMS, where different product types can share production components and production components can hold multiple products. We first point out that a suitable model for such production component needs to keep track of the product type and the order of products entering and leaving production components. Then, we develop a general method for algorithmically constructing the required order-preserving models. We further illustrate the practicability of the developed method by an application example.

Keywords: Discrete event systems, supervisory control, order-preserving models, flexible manufacturing systems.

1. Introduction

Flexible manufacturing systems (FMS) have the ability of manufacturing different types of products using a given hardware setup with various production components such as machines, robots and conveyor belts [1, 2, 3, 4, 5]. In an FMS, it is generally desired to move products along predefined paths through the FMS and use pre-specified production components for processing these products in a given sequence [6, 7, 8].

The logic control of FMS can be carried out in the framework of supervisory control for discrete event systems (DES) [9]. In this context, the desired production sequences are realized by a supervisor that is designed based on a formal DES model of the FMS and a formal specification of the production sequence. In this context, the existing literature considers various aspects regarding the supervisory control of FMS. The work in [10, 11, 12, 13] focuses on the avoidance of deadlocks or forbidden states in FMS and [6, 7, 14, 15] develop modular and hierarchical methods for the efficient computation of supervisors for FMS. Furthermore, there is recent work such as [8] that

proposes the concept of distinguishers in order to facilitate the modeling of FMS with different product types.

The subject of this paper is the modeling of an additional property of FMS that has not been addressed in previous work on the supervisory control of FMS. In particular, it is possible that production components can hold multiple products and then process them sequentially. In this case, it is required that a product that enters the production component first will also be processed first and then leave the production component first. That is, if a production component is able to process different product types, it is necessary to keep track of the different product types entering and leaving such production component. Nevertheless, this case does not appear in the existing literature. For FMS with different product types it is either the case that these product types have independent paths in the FMS [3, 8, 16], the product paths are defined such that it is not necessary to remember the product order [1, 6, 7, 14, 15, 17, 18] or the product order is not taken into account and it is implicitly assumed that the production component knows which product is currently transported [2, 10, 11], In order to address this issue, we first determine scenarios under which a suitable model for a production component needs to keep track of the product type and the order of products entering and leaving. Then, we develop a general method for algorithmically constructing the required order-preserving models of production components. This method takes into account the neighborhood relationship between production components and is applicable to production components that can hold an arbitrary number of products and that can process an arbitrary number of product types. We demonstrate the practicability of the proposed modeling method by an FMS example with multiple products and overlapping product paths.

The remainder of the paper is organized as follows. Section 2 introduces the basic notation regarding DES and supervisory control. Section 3 formulates the problem addressed in this paper based on motivating examples. The proposed order-preserving modeling method is developed in Section 4 and applied to an FMS example in Section 5. Section 6 gives conclusions and states ideas for future work.

2. Preliminaries

This section provides the necessary notation for the formal representations in the paper. Section 2.1 introduces the notions related to discrete event systems (DES) and Section 2.2 gives the background on the supervisory control theory.

2.1. Discrete Event Systems (DES)

The behavior of a DES is represented by formal languages over finite alphabets Σ. Each element $\sigma \in \Sigma$ is denoted as an event, $\Sigma^\star$ denotes the set of all finite strings over Σ and $s_1, s_2 \in \Sigma^\star$ defines

the concatenation of two strings $s_1 s_2 \in \Sigma^\star$. $s_1 \leq s$ indicates that s_1 is a *prefix* of s and $\varepsilon \in \Sigma^\star$ is the empty string. A formal *language* over Σ is a subset $L \subseteq \Sigma^\star$. $\overline{L} := \{s_1 \in \Sigma^\star | \exists s \in L \text{ s.t. } s_1 \leq s\}$ defines the *prefix closure* of L, and L is called *prefix closed* if $L = \overline{L}$.

Consider two alphabets Σ_1, Σ_2 and their union $\Sigma = \Sigma_1 \cup \Sigma_2$. Then, the *natural projection* $p_i : \Sigma^\star \to \Sigma_i^\star$, $i = 1,2$, is defined iteratively such that (1) $p_i(\varepsilon) := \varepsilon$; (2) for $s \in \Sigma^\star$, $\sigma \in \Sigma$: $p_i(s\sigma) := p_i(s)\sigma$ if $\sigma \in \Sigma_i$, or $p_i(s\sigma) := p_i(s)$ otherwise. The set-valued inverse of p_i is written as $p_i^{-1} : \Sigma_i^\star \to 2^{\Sigma^\star}$, $p_i^{-1}(t) := \{s \in \Sigma^\star | \, p_i(s) = t\}$. Using the natural projection, the *synchronous product* $L_1 || L_2 \subseteq \Sigma^\star$ of two languages $L_i \subseteq \Sigma_i^\star$ is computed as $L_1 || L_2 = p_1^{-1}(L_1) \cap p_2^{-1}(L_2) \subseteq \Sigma^\star$.

We model a DES as a finite state automaton $G = (X, \Sigma, \delta, x_0, X_\mathrm{m})$, with the finite set of *states* X; the finite alphabet of *events* Σ; the partial *transition function* $\delta : X \times \Sigma \to X$; the *initial state* $x_0 \in X$ and the set of *marked states* $X_\mathrm{m} \subseteq X$. Hereby, $\delta(x, \sigma)!$ is written if δ is defined at (x, σ) and we extend the transition function δ to a partial function on $X \times \Sigma^\star$ in the usual way. The behavior of G is given by its *closed language* $L(G) := \{s \in \Sigma^\star | \delta(x_0, s)!\}$ and its *marked language* $L_\mathrm{m}(G) := \{s \in L(G) | \delta(x_0, s) \in X_\mathrm{m}\}$. The synchronous composition $G_1 || G_2$ of two automata G_1, G_2 is defined in the usual way [9] such that $L(G_1 || G_2) = L(G_1) || L(G_2)$.

2.2. Supervisory Control

In supervisory control, we write $\Sigma = \Sigma_\mathrm{c} \dot{\cup} \Sigma_\mathrm{u}$ for *controllable* (Σ_c) and *uncontrollable* (Σ_u) events. We say that an automaton $S = (Q, \Sigma, \nu, q_0, Q_\mathrm{m})$ is a *supervisor* for a plant G if S can only disable events in Σ_c. In particular, it must hold for all $s \in L(G) \cap L(S)$ and $\sigma \in \Sigma_\mathrm{u}$ with $s\sigma \in L(G)$ that also $s\sigma \in L(S)$. Then, $L(G) || L(S)$ and $L_\mathrm{m}(G) || L_\mathrm{m}(S)$ represent the closed and marked behavior of the closed-loop system $G || S$, respectively.

A language $K \subseteq L_\mathrm{m}(G)$ is said to be controllable for $L(G)$ and Σ_u if $\overline{K}\Sigma_\mathrm{u} \cap L(G) \subseteq \overline{K}$. There exists a supervisor S such that $L_\mathrm{m}(G||S) = K$ if and only if K is controllable for $L(G)$ and Σ_u [**?**]. In case, K is not controllable for $L(G)$ and Σ_u, the supervisor will implement the *supremal controllable sublanguage* of K. We write $L_m(S||G) = \mathrm{SupC}(K, G, \Sigma_\mathrm{u})$. It is ensured that such supervisor is nonblocking and maximally permissive if $\mathrm{SupC}(K, G, \Sigma_\mathrm{u}) \neq \emptyset$ [19].

3. Problem Statement

The main topic of this paper is the generation of DES models that retain the order of products for production systems. To this end, Section 3.1 first motivates the problem setting based on an illustrative example. Then, Section 3.2 states the research problem addressed in this paper.

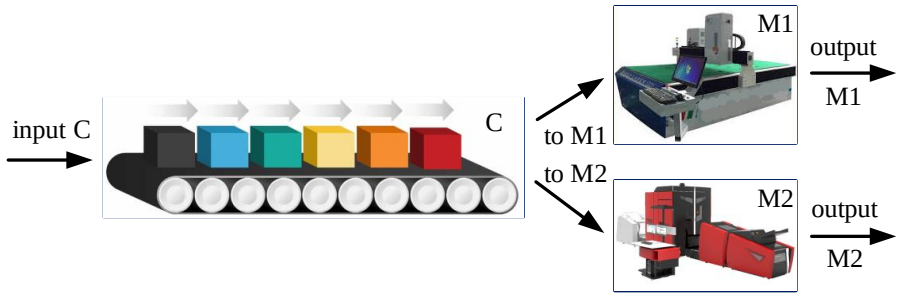

FIGURE 1. Example system with one conveyor (C) and two machines (M_1 and M_2).

3.1. Motivation

In this paper, we focus on the flow of products through a production system. Consider for example the simple production system with one conveyor belt and two machines in Fig. 1.

We study the case where the conveyor belt (C) is potentially long and can hold multiple products simultaneously. Products are fed to C from the left and are then transported to one of the machines M1 or M2 on the right hand side of the figure. Each product is then processed in the respective machine and leaves the example system on the right hand side.

In this simple setup, the common automaton model G_C of the conveyor and the automata models G_{M1} and G_{M2} of the machines are as given in Fig. 2. That is, the conveyor receives products with the event `inC` and delivers products towards the machines with the event `outC`. If C can hold at most n products simultaneously (that is, the capacity of C is n), G_C has $n+1$ states to keep memory of the number of products in C. The machines M1 and M2 obtain products with the event `inM1` and `inM2`, respectively. After processing products are delivered to the outside with `outM1` and `outM2`.

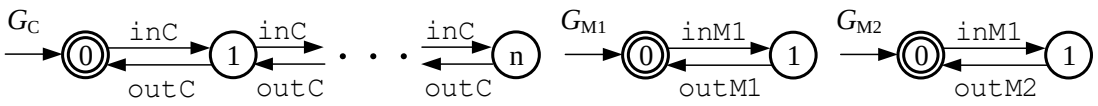

FIGURE 2. Automata models for the machines M1 and M2.

Although such models are frequently used in the modeling of manufacturing systems, flexible manufacturing systems (FMS) [20] as well as reconfigurable manufacturing systems (RMS) [21], we argue that these models are not suitable if there are different product types (with different processing requirements). To this end, we next discuss four relevant scenarios and their effect on the conveyor model as depicted in Fig. 3.

3.1.1. Scenario with a single product type and capacity of one product on C. Fig. 3 (a) shows the case where there is a single product type and C has capacity for a single product. It is further

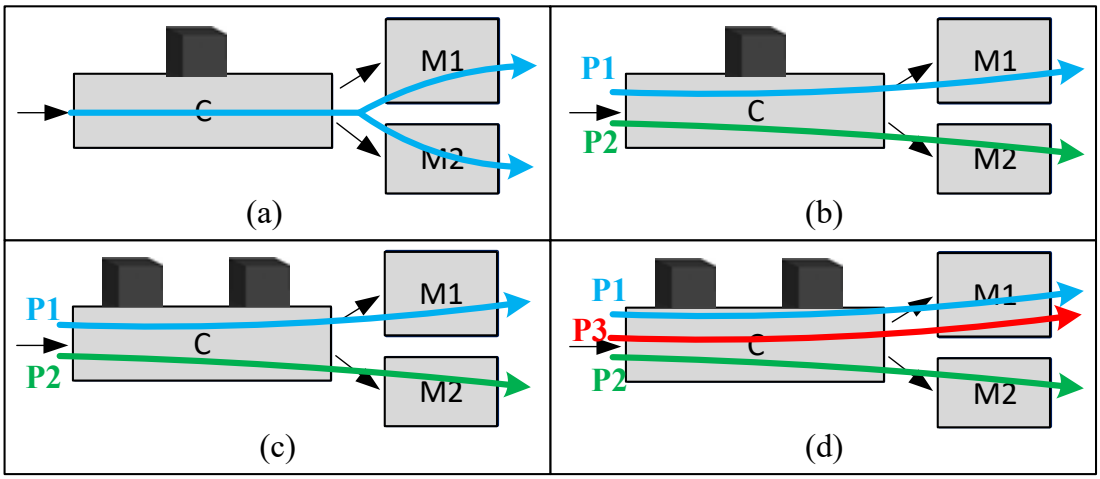

FIGURE 3. Conveyor belt: (a) One product type and capacity one; (b) two product types and capacity one; (c) two product types and capacity two; (d) three product types and capacity two.

desired that any product is delivered to one of the machines arbitrarily. In this case, it needs to be taken into account that (i) there is no difference between products that enter C; (ii) any product that enters C can leave to two different machines. Accordingly, input of a product to C can be modeled by the event `inC`, whereas product output has to be modeled by two different events `C-M1` (from C to M1) and `C-M2` (from C to M2) as is shown in Fig. 4. Similarly, the model of the machines in Fig. 2 has to be adjusted such that `inM1` is replaced by `C-M1` and `inM2` is replaced by `C-M2` in order to match the events defined for C.

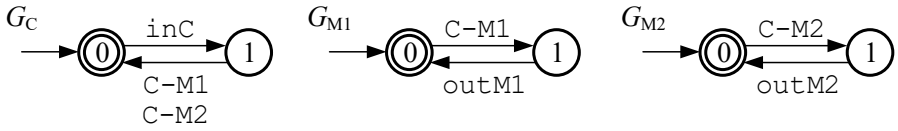

FIGURE 4. Models for the case in Fig. 3 (a).

3.1.2. Scenario with two product types and capacity of one product on C. We next investigate the scenario in Fig. 3 (b). Here, the conveyor can hold a single product but there are two product types. Product P1 (blue) needs to be processed by M1, whereas product P2 (green) has to be processed by M2. In this case, it needs to be taken into account that (i) different products enter C; (ii) depending on the product type in C, the product leaves to M1 or M2. That is, the model of C in Fig. 4 is not suitable since it cannot distinguish products entering C. In order to distinguish the product type, a refinement of the event alphabet of G_C is required by introducing separate events for the different product types. That is, instead of $\Sigma_C = \{\texttt{inC}, \texttt{C-M1}, \texttt{C-M2}\}$, we use $\Sigma_C = \{\texttt{inC}_{\texttt{P1}}, \texttt{inC}_{\texttt{P2}}, \texttt{C-M1}, \texttt{C-M2}\}$. In addition, the model needs to respect the order of products entering and leaving C. In particular, it is not possible that P2 leaves C (`C-M2`) after P1 enters

C ($\texttt{inC}_{\texttt{P1}}$) and vice versa. A suitable model for this scenario is shown in Fig. 5. Here, G_{C} expresses that each event $\texttt{C-Mi}$ is only possible after the respective event $\texttt{inC}_{\texttt{Pi}}$, $\mathrm{i} \in \{1,2\}$.

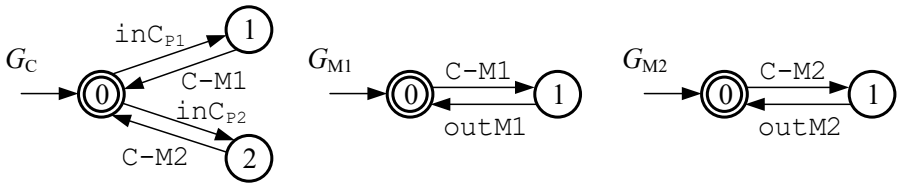

FIGURE 5. Models for the case in Fig. 3 (b).

3.1.3. Scenario with two product types and capacity for two products on C. A more complicated modeling problem is encountered in the scenario in Fig. 3 (c). Here, two different product types P1 and P2 are produced and C can hold up to two products. In this case, it needs to be taken into account that (i) different products enter C; (ii) depending on the product type in C, the product leaves to M1 or M2; (iii) the order of products entering and leaving C must be preserved in the model. Similar to the previous case, it is again necessary to refine the event alphabet of G_{C} as $\Sigma_{\mathrm{C}} = \{\texttt{inC}_{\texttt{P1}}, \texttt{inC}_{\texttt{P2}}, \texttt{C-M1}, \texttt{C-M2}\}$. In addition, the model has to remember the order in which products enter the system since products have to leave C in the same order. A suitable model for this purpose is shown in Fig. 6. Here, G_{C} has one state for each possible product combination on C as is indicated by the color code in the respective states (P1 – blue; P2 – green). For example, it can be seen that the input order P1, P2 of products (string $\texttt{inC}_{\texttt{P1}}\ \texttt{inC}_{\texttt{P2}}$) leads to a state where only P1 can exit C to M1 (event $\texttt{C-M1}$), whereas P2 has to wait until P1 leaves C.

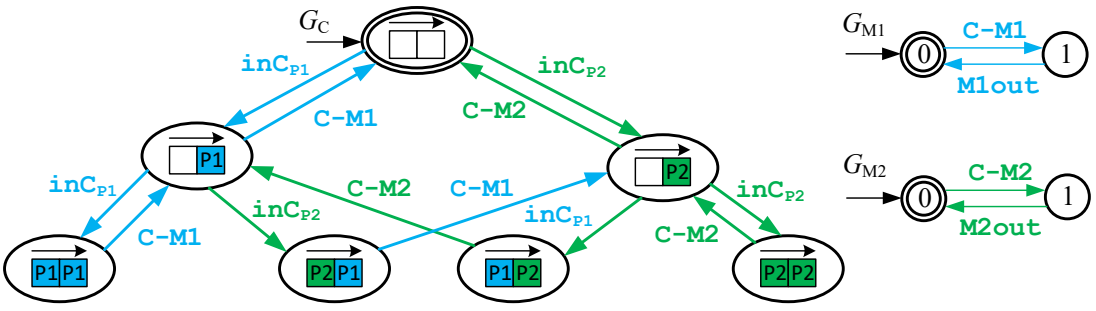

FIGURE 6. Models for the the case in Fig. 3 (c).

3.1.4. Scenario with three product types and capacity for two products on C. The next example shows that the modeling problem need not be restricted to a single component of a production system. To this end, we consider the scenario in Fig. 3 (d). Here, there are three product types P1, P2, P3 and C has a capacity of two products. P2 is delivered to M2, whereas both P1

and P3 are delivered to M1. That is, P1 and P3 share the same path within the example system but might follow a different path after leaving the example system. In this case, it needs to be taken into account that (i) different products enter C; (ii) depending on the product type in C, the product leaves to M1 or M2; (iii) the order of products entering and leaving C and M1 must be preserved in the model. Since there are three products in C and two products in M1, we refine the respective alphabets as $\Sigma_{\mathrm{C}} = \{\mathtt{inC}_{\mathtt{P1}}, \mathtt{inC}_{\mathtt{P2}}, \mathtt{inC}_{\mathtt{P3}}, \mathtt{C\text{-}M1}_{\mathtt{P1}}, \mathtt{C\text{-}M1}_{\mathtt{P3}}, \mathtt{C\text{-}M2}_{\mathtt{P2}}\}$, $\Sigma_{\mathrm{M1}} = \{\mathtt{C\text{-}M1}_{\mathtt{P1}}, \mathtt{C\text{-}M1}_{\mathtt{P3}}, \mathtt{outM1}_{\mathtt{P1}}, \mathtt{outM1}_{\mathtt{P3}}\}$ and $\Sigma_{\mathrm{M2}} = \{\mathtt{C\text{-}M2}_{\mathtt{P2}}, \mathtt{outM2}_{\mathtt{P2}}\}$. In addition, the model has to remember the order in which three different products enter C (capacity 2) and the order in which two different products enter M1 (capacity 1). A suitable model for this purpose is shown in Fig. 7. Similar to Fig. 6, G_{C} has one state for each possible product combination on C. Since there are more product types, more combinations have to be considered. Finally, the model for M1 has the same structure as the model for C in Fig. 5 since there are two products types and the capacity is one.

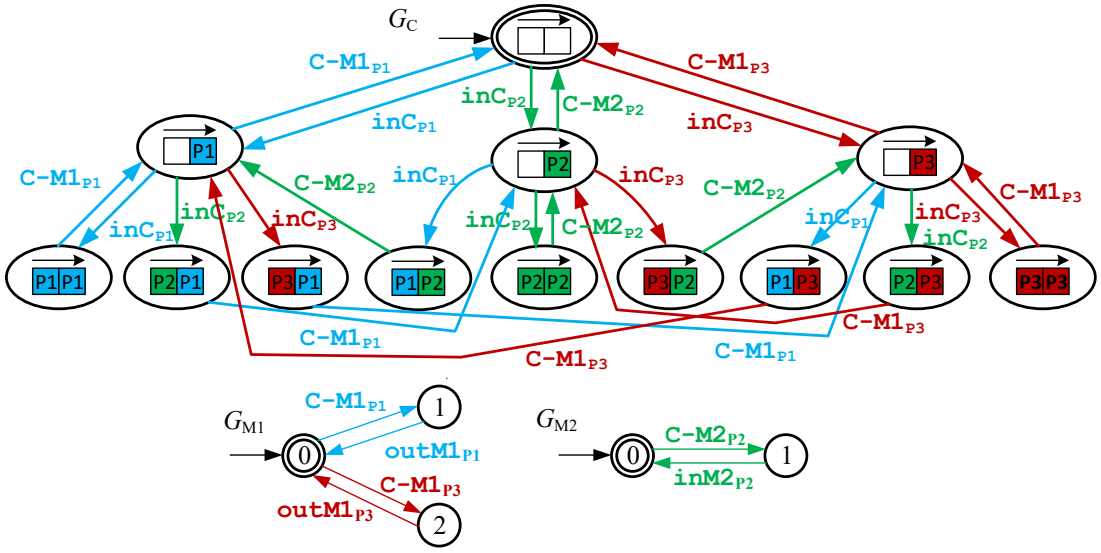

FIGURE 7. Models for the the case in Fig. 3 (d).

3.2. Discussion and Research Problem

The main purpose of the example scenarios in Fig. 3 is to analyze the effect of different product types and different capacities of production components on the structure of the respective automaton model. In particular, it can be observed from the scenario in Fig. 3 (b) that DES models need to distinguish products if there is more than one product type. In addition, the scenario in Fig. 3 (c) indicates that the DES model needs to remember the order of products if there are multiple

product types and the capacity of the production component is greater than one. Finally, the scenario in Fig. 3 (d) shows that the information about capacity and different product types has to be incorporated in the DES models of all relevant production components.

In this context, we note that there is a vast literature on the control of FMS and RMS with potentially different product types [1, 2, 3, 6, 7, 8, 10, 11, 14, 15, 16, 17, 18]. Nevertheless, none of the existing studies considers the case of DES models that preserve the order of products.

Regarding FMS and RMS with different product types, the literature generally considers two special cases regarding the system specification. In the first case, it is assumed that the different product types travel on different paths through the system and hence do not pass the same production components [3, 8, 16]. Accordingly, it is not necessary to find models that distinguish products. In the second case, the paths of different products can intersect such that there are production components which are passed by multiple products [1, 2, 6, 7, 10, 11, 14, 15, 17, 18]. In one line of work, the specifications are always chosen such that it is not necessary to remember the product order [1, 6, 7, 14, 15, 17, 18]. More precisely, different products never share a path with production components with a capacity greater than one. In another line of work [2, 10, 11], the product order is not taken into account and it is implicitly assumed that it is know which product travels between production components. In a real application this knowledge can only be obtained by sensors that detect the prototype everywhere in the production system, which is impractical.

In view of the above discussion, the main objective of this paper is a systematic approach to the modeling of production systems with different product types and production components with a capacity that is greater than one. In particular, Section 4 formalizes the notion of an order-preserving DES model and proposes an algorithmic procedure to construct such model. Moreover, Section 5 illustrates the applicability of the proposed modeling technique by the application to FMS supervisory control.

4. Order-preserving DES Models

Based on the discussion in the previous section, this section systematically addresses the problem of modeling production systems with multiple product types and production components with a capacity greater than one. Section 4.1 formalizes the notion of an order-preserving DES model proposes an algorithm for determining such models. Section 4.2 discusses properties of the order-preserving model.

4.1. Modular Production Systems and Order-preserving Models

We consider modular production systems (MPSs) with a set $\mathscr{C} = \{\mathrm{C1}, \ldots, \mathrm{Cm}\} \cup \{\mathrm{I}, \mathrm{O}\}$ of m production components and the virtual input component I and output component O. We assume

that the MPS is processing a set of n products $\mathscr{P} = \{\mathrm{P1}, \ldots, \mathrm{Pn}\}$ and the main aim of this paper is to model the flow of products in such production system. To this end, we introduce $\mathscr{P}_{\mathrm{Ci}} \subseteq \mathscr{P}$ as the set of products that pass component Ci. Considering production components $\mathrm{Ci}, \mathrm{Cj} \in \mathscr{C}$ such that $\mathrm{Ci} \neq \mathrm{Cj}$, we define $\mathscr{P}_{\mathrm{Ci,Cj}} \subseteq \mathscr{P}_{\mathrm{ci}}$ as the set of products leaving component Ci to Cj and $\mathscr{P}_{\mathrm{Cj,Ci}} \subseteq \mathscr{P}_{\mathrm{ci}}$ as the set of products arriving at Ci from component Cj. In particular, $\mathscr{P}_{\mathrm{Ci,Cj}} = \emptyset$ if no products are transported from Ci to Cj and it must hold that

$$\bigcup_{\mathrm{Cj} \in \mathscr{C}} \mathscr{P}_{\mathrm{Cj,Ci}} = \bigcup_{\mathrm{Cj} \in \mathscr{C}} \mathscr{P}_{\mathrm{Ci,Cj}}. \tag{1}$$

That is, all products entering component Ci should also be able to leave Ci.

For the example in Fig. 1, we have $\mathscr{C} = \{\mathrm{C}, \mathrm{M1}, \mathrm{M2}, \mathrm{I}, \mathrm{O}\}$. Then, the sets $\mathscr{P}_{\mathrm{Ci,Cj}}$ depend on the product flow. Consider for example the scenario in Fig. 3 (d). Here, $\mathscr{P}_{\mathrm{I,C}} = \{\mathrm{P1}, \mathrm{P2}, \mathrm{P3}\}$, $\mathscr{P}_{\mathrm{C,M1}} = \{\mathrm{P1}, \mathrm{P3}\}$, $\mathscr{P}_{\mathrm{C,M2}} = \{\mathrm{P2}\}$, $\mathscr{P}_{\mathrm{M1,O}} = \{\mathrm{P1}, \mathrm{P3}\}$, $\mathscr{P}_{\mathrm{M2,O}} = \{\mathrm{P2}\}$. The remaining sets are $\mathscr{P}_{\mathrm{I,M1}} = \mathscr{P}_{\mathrm{I,M2}} = \mathscr{P}_{\mathrm{I,O}} = \mathscr{P}_{\mathrm{O,I}} = \mathscr{P}_{\mathrm{O,M1}} = \mathscr{P}_{\mathrm{O,M2}} = \mathscr{P}_{\mathrm{O,C}} = \mathscr{P}_{\mathrm{M1,I}} = \mathscr{P}_{\mathrm{M2,I}} = \mathscr{P}_{\mathrm{C,I}} = \emptyset$.

Using the information about the product transport between production components, we suggest to define appropriate alphabets for the DES model of each production component. Specifically, for any $\mathrm{Ci} \in \mathscr{C}$ we introduce the input alphabet $\Sigma^{\mathrm{in}}_{\mathrm{Ci}}$ and the output alphabet $\Sigma^{\mathrm{out}}_{\mathrm{Ci}}$ as

$$\Sigma^{\mathrm{in}}_{\mathrm{Ci}} = \bigcup_{\mathrm{Cj} \in \mathscr{C}} \bigcup_{\mathrm{Pk} \in \mathscr{P}_{\mathrm{Cj,Ci}}} \mathtt{Cj\text{-}Ci}_{\mathtt{Pk}} \cup \bigcup_{\mathrm{Pk} \in \mathscr{P}_{\mathrm{I,Ci}}} \mathtt{inCi}_{\mathtt{Pk}}, \tag{2}$$

$$\Sigma^{\mathrm{out}}_{\mathrm{Ci}} = \bigcup_{\mathrm{Cj} \in \mathscr{C}} \bigcup_{\mathrm{Pk} \in \mathscr{P}_{\mathrm{Ci,Cj}}} \mathtt{Ci\text{-}Cj}_{\mathtt{Pk}} \cup \bigcup_{\mathrm{Pk} \in \mathscr{P}_{\mathrm{Ci,O}}} \mathtt{outCi}_{\mathtt{Pk}}. \tag{3}$$

In words, $\Sigma^{\mathrm{in}}_{\mathrm{Ci}}$ contains events for each product transport from neighboring components (including the input component I) and for each possible product type. Similarly, $\Sigma^{\mathrm{out}}_{\mathrm{Ci}}$ contains events for each product transport to neighboring components (including the output component O) with each possible product type. Then, the overall alphabet of any production component $\mathrm{Ci} \in \mathscr{C}$ is $\Sigma_{\mathrm{Ci}} = \Sigma^{\mathrm{in}}_{\mathrm{Ci}} \cup \Sigma^{\mathrm{out}}_{\mathrm{Ci}}$.

We note that the alphabets of the models in Section 3.1 are selected according to (2) and (3). For example, we have $\Sigma^{\mathrm{in}}_{\mathrm{C}} = \{\mathtt{inC}_{\mathtt{P1}}, \mathtt{inC}_{\mathtt{P2}}, \mathtt{inC}_{\mathtt{P3}}\}$ and $\Sigma^{\mathrm{out}}_{\mathrm{C}} = \{\mathtt{C\text{-}M1}_{\mathtt{P1}}, \mathtt{C\text{-}M2}_{\mathtt{P2}}, \mathtt{C\text{-}M1}_{\mathtt{P3}}\}$ for component C in Fig. 7.

Finally, we write c_{Ci} for the product capacity of component $\mathrm{Ci} \in \mathscr{C}$ (maximum number of products that can be in component Ci simultaneously).

Using c_{Ci}, Σ_{Ci}, $\mathscr{P}_{\mathrm{Cj,Ci}}$ and $\mathscr{P}_{\mathrm{Ci,Cj}}$ for $\mathrm{Cj} \in \mathscr{C}$, we define a general automaton model $G_{\mathrm{Ci}} = (X_{\mathrm{Ci}}, \Sigma_{\mathrm{Ci}}, \delta_{\mathrm{Ci}}, x_{0,\mathrm{Ci}}, X_{\mathrm{m,Ci}})$ for a production component Ci with multiple product types and a product capacity that is greater than one. Referring to G_{C} in Fig. 6 and 7, we note that each state of the automaton model represents a possible sequence of products entering the component. For a

production component Ci, with the capacity c_{Ci}, the state set X_{Ci} is hence given by

$$X_{\mathrm{Ci}} = \{\mathrm{E}\} \cup \bigcup_{l=1}^{c_{\mathrm{Ci}}} \Big(\bigtimes_{k=1}^{l} \mathscr{P}_{\mathrm{Ci}} \Big). \tag{4}$$

Hereby, E is the empty state and the Cartesian product $\times$ with the set $\mathscr{P}_{\mathrm{Ci}}$ represents the combinations of products in the component. Then, the initial state is given as $x_{0,\mathrm{Ci}} = \mathrm{E}$ and the set of marked states is $X_{\mathrm{m,Ci}} = \{\mathrm{E}\}$ since it is always desired to go back to the empty state of a production component in order to complete all production tasks.

For example, the state set for component C in Fig. 6 with the products in $\mathscr{P}_{\mathrm{C}} = \{\mathrm{P1}, \mathrm{P2}\}$ is found as $X_{\mathrm{C}} = \{\mathrm{E}, \mathrm{P1}, \mathrm{P2}, (\mathrm{P1}, \mathrm{P1}), (\mathrm{P1}, \mathrm{P2}), (\mathrm{P2}, \mathrm{P1}), (\mathrm{P2}, \mathrm{P2})\}$. Then, $x_{0,\mathrm{C}} = \mathrm{E}$ and $X_{\mathrm{m,C}} = \{\mathrm{E}\}$.

It remains to determine the transition relation δ_{Ci}. To this end, we first observe that any input event in $\Sigma_{\mathrm{Ci}}^{\mathrm{in}}$ adds one new product to the production component and each output event in $\Sigma_{\mathrm{Ci}}^{\mathrm{out}}$ removes the oldest product from the production component. That is, assuming that the current state of Ci is $(\mathrm{Pk}, \cdots, \mathrm{Pl})$ and an input event with product Pm occurs, the new state is $(\mathrm{Pm}, \mathrm{Pk}, \cdots, \mathrm{Pl})$ (adding the new product PM). Similarly, assuming that the current state of Ci is $(\mathrm{Pk}, \cdots, \mathrm{Pl}, \mathrm{Pm})$, then only output events with product Pm are possible and the new state is $(\mathrm{Pk}, \cdots, \mathrm{Pl})$ (taking out the "oldest" product Pm). Using this observation, the transition relation is defined for any state $x \in X_{\mathrm{Ci}}$ and $\sigma_{\mathrm{Pm}} \in \Sigma_{\mathrm{Ci}}$ as

$$\delta_{\mathrm{Ci}}(x, \sigma_{\mathrm{Pm}}) = \begin{cases} (\mathrm{Pm}) & \text{if } x = \mathrm{E} \text{ and } \sigma_{\mathrm{Pm}} \in \Sigma_{\mathrm{Ci}}^{\mathrm{in}} \\ (\mathrm{Pm}, \mathrm{Pk}, \cdots, \mathrm{Pl}) & \text{if } x = (\mathrm{Pk}, \cdots, \mathrm{Pl}) \text{ and } \sigma_{\mathrm{Pm}} \in \Sigma_{\mathrm{Ci}}^{\mathrm{in}} \\ \mathrm{E} & \text{if } x = (\mathrm{Pm}) \text{ and } \sigma_{\mathrm{Pm}} \in \Sigma_{\mathrm{Ci}}^{\mathrm{out}} \\ (\mathrm{Pk}, \cdots, \mathrm{Pl}) & \text{if } x = (\mathrm{Pk}, \cdots, \mathrm{Pl}, \mathrm{Pm}) \text{ and } \sigma_{\mathrm{Pm}} \in \Sigma_{\mathrm{Ci}}^{\mathrm{out}} \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{5}$$

Consider for example the model G_{C} in Fig. 6. Here, the transitions $\delta_{\mathrm{C}}(\mathrm{E}, \mathtt{inC_{P1}}) = (\mathrm{P1})$, $\delta_{\mathrm{C}}((\mathrm{P1}), \mathtt{inC_{P2}}) = (\mathrm{P2}, \mathrm{P1})$, $\delta_{\mathrm{C}}((\mathrm{P1}, \mathtt{outC_{P1}}) = \mathrm{E}$ and $\delta_{\mathrm{C}}((\mathrm{P1}, \mathrm{P2}), \mathtt{outC_{P2}}) = (\mathrm{P1})$ are introduced according to the rules in (5).

4.2. Order-preserving Models

We recall that the main objective of this paper is the modeling of production components that can hold multiple products with different types, while preserving the order to products entering and leaving the component. For the first time in the literature, the model introduced in Section 4.1 fulfill this purpose. In particular, we point out that the label of each state in G_{Ci} keeps track of the order of products entering the component Ci. In the tuple $(\mathrm{Pk}, \ldots, \mathrm{Pl})$, old products appear on the right, whereas new products appear on the left. On the one hand, any new product Pm entering the component with an input event $\sigma_{\mathrm{Pm}} \in \Sigma_{\mathrm{Ci}}^{\mathrm{in}}$ is added to the state label from the left, to obtain the next state $(\mathrm{Pm}, \mathrm{Pk}, \ldots, \mathrm{Pl})$ in (5). Hence, indeed, the state label always correctly characterizes the

order of incoming products. On the other hand, any product that leaves the system at some state $(\mathrm{Pk},\ldots,\mathrm{Pl},\mathrm{Pm})$ must be the oldest product Pm on the right of the tuple as stated in (5). Hence, products indeed leave the component in the same order as entering the component. Accordingly, we denote the model in Section 4.1 as an *order-preserving* DES model.

As a further interesting feature of the proposed model, it is possible to determine the number of states $|X_{\mathrm{Ci}}|$ depending on the product capacity c_{Ci} and the number of products $|\mathscr{P}_{\mathrm{Ci}}|$. Inspecting (4), it holds that

$$|X_{\mathrm{Ci}}| = 1 + \sum_{k=1}^{c_{\mathrm{Ci}}} |\mathscr{P}_{\mathrm{Ci}}|^k = \sum_{k=0}^{c_{\mathrm{Ci}}} |\mathscr{P}_{\mathrm{Ci}}|^k. \tag{6}$$

In particular, for each possible number k of products in the component with $0 \leq k \leq n_{\mathrm{Ci}}$, there are $|\mathscr{P}_{\mathrm{Ci}}|^k$ combinations of product types.

For example, an order-preserving model for a production component Ci with capacity $n_{\mathrm{Ci}} = 2$ and $|\mathscr{P}_{\mathrm{Ci}}| = 3$ product types, the model has $1+3+9 = 13$ states (see Fig. 7) and the model of a production component with capacity $n_{\mathrm{Ci}} = 4$ and $|\mathscr{P}_{\mathrm{Ci}}| = 2$ product types has $1+2+2^2+2^3+2^4 = 31$ states.

5. Application Example

In this section, we apply the modeling technique introduced in Section 4 to an FMS example that is adapted from [2, 10]. Section 5.1 describes the components of the example system and Section 5.2 determines suitable automata models. The supervisor computation for the FMS is discussed in Section 5.3.

5.1. FMS Example

The FMS example consists of 3 robots R1, R2, R3 and two machines M1 and M2. That is, $\mathscr{C} = \{\mathrm{R1},\mathrm{R2},\mathrm{R3},\mathrm{M1},\mathrm{M2},\mathrm{I},\mathrm{O}\}$ including the virtual input and output components. Hereby, it is assumed that the capacity of the robots is $c_{\mathrm{R1}} = c_{\mathrm{R2}} = c_{\mathrm{R3}} = 1$ whereas the machines can hold up to $c_{\mathrm{M1}} = c_{\mathrm{M2}} = 2$ products. There are two product types P1 and P2 that arrive at the FMS from the virtual input component I and leave the FMS to the virtual output component O. P1 is picked from I by R3, moved to M2, then transported to M1 by R2 and finally transported to O by R1. P2 is picked from I by R1, moved to M1, then transported to M2 by R2 and finally transported to O by R3. That is, both product paths overlap and both products are processed by M1 and M2, whose capacity is two. Hence, the models for these machines need to keep track of the product order.

5.2. Order-preserving Model for the Example System

In order to model the FMS, we first determine the sets $\mathscr{P}_{\mathrm{Ci,Cj}}$ that characterize the product exchange between neighboring components. It holds that $\mathscr{P}_{\mathrm{I,R3}} = \{\mathrm{P1}\}$, $\mathscr{P}_{\mathrm{R3,M2}} = \{\mathrm{P1}\}$, $\mathscr{P}_{\mathrm{M2,R2}} =$

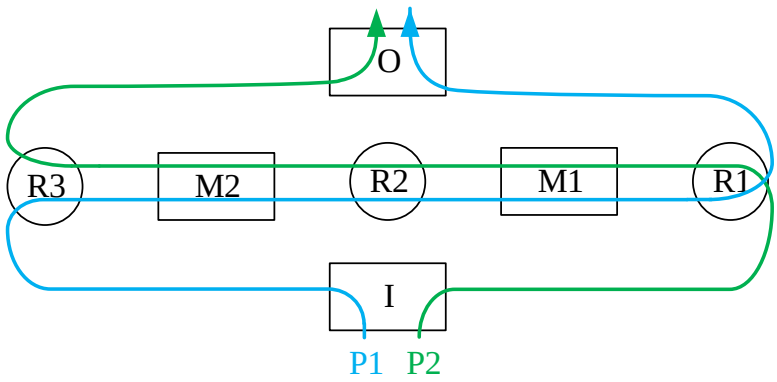

FIGURE 8. Schematic of the FMS.

$\{P1\}$, $\mathscr{P}_{R2,M1} = \{P1\}$, $\mathscr{P}_{M1,R1} = \{P1\}$, $\mathscr{P}_{R1,O} = \{P1\}$, $\mathscr{P}_{I,R1} = \{P2\}$, $\mathscr{P}_{R1,M1} = \{P2\}$, $\mathscr{P}_{M1,R2} = \{P2\}$, $\mathscr{P}_{R2,M2} = \{P2\}$, $\mathscr{P}_{M2,R3} = \{P2\}$, $\mathscr{P}_{R3,O} = \{P2\}$. All remaining sets are empty.

Accordingly, it is possible to define the input and output alphabets of different production components according to (2) and (3) as $\Sigma^{in}_{R1} = \{\texttt{inR1}_{P2}, \texttt{M1-R1}_{P1}\}$, $\Sigma^{out}_{R1} = \{\texttt{R1-M1}_{P2}, \texttt{outR1}_{P1}\}$, $\Sigma^{in}_{R2} = \{\texttt{M2-R2}_{P1}, \texttt{M1-R2}_{P2}\}$, $\Sigma^{out}_{R2} = \{\texttt{R2-M1}_{P1}, \texttt{R2-M2}_{P2}\}$, $\Sigma^{in}_{R3} = \{\texttt{inR3}_{P1}, \texttt{M2-R3}_{P2}\}$, $\Sigma^{out}_{R3} = \{\texttt{R3-M2}_{P1}, \texttt{outR3}_{P2}\}$, $\Sigma^{in}_{M1} = \{\texttt{R2-M1}_{P1}, \texttt{R1-M1}_{P2}\}$, $\Sigma^{out}_{M1} = \{\texttt{M1-R1}_{P1}, \texttt{M1-R2}_{P2}\}$ and $\Sigma^{in}_{M2} = \{\texttt{R3-M2}_{P1}, \texttt{R2-M2}_{P2}\}$, $\Sigma^{out}_{M2} = \{\texttt{R2-R2}_{P1}, \texttt{M2-R3}_{P2}\}$.

Considering that the robots all have a capacity of one product, their models according to (4) and (5) have 3 states as shown in Fig. 9.

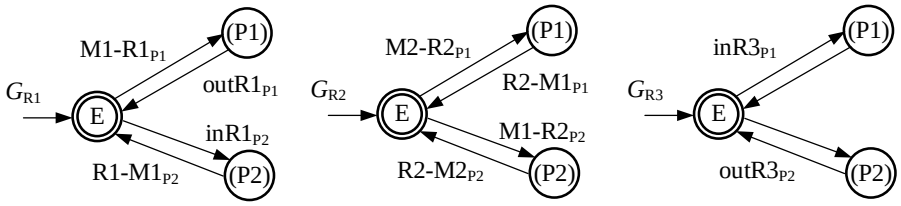

FIGURE 9. Robot models.

Differently, the machines have a capacity of two product and two product types pass M1 and M2. Hence, their models need to remember the product order similar to the model in Fig. 6. The resulting models for M1 and M2 are shown in Fig. 10 and 11, respectively.

The overall model G of the FMS is then given by the synchronous composition of the component models are

$$G = G_{R1} || G_{R2} || G_{R3} || G_{M1} || G_{M2}. \tag{7}$$

Since this model has 2368 states, it is not displayed in the paper.

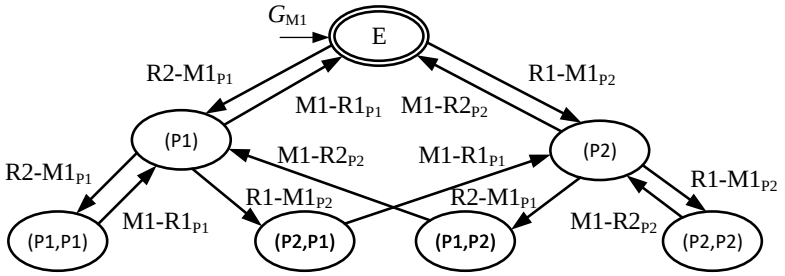

FIGURE 10. Model of machine M1.

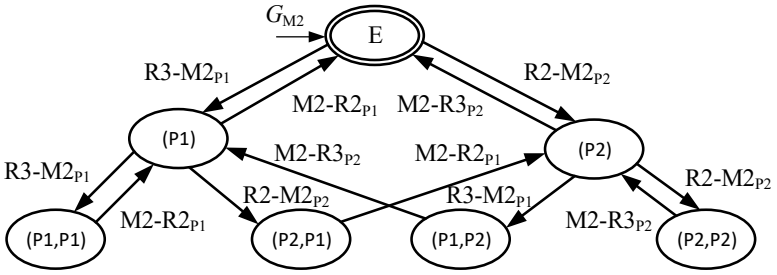

FIGURE 11. Model of machine M2.

5.3. Supervisor Computation for the Example System

The FMS model G already represents all the possible product paths as specified in Fig. 8. However, it turns out that G is a blocking automaton, that is, the uncontrolled FMS will encounter deadlock situations. Two examples of such deadlock situation are illustrated in Fig. 12. The figure shows the production components (robots and machines), whereby, the machines are displayed in the form of a FIFO (first-in-first-out) queue that can hold up to two products and such that the product entering the component first will also leave first. Products of type P1 are represented by blue disks, whereas products of type P2 are shown as green triangles.

Consider the scenario on the left-hand side of Fig. 12. Here, two products of type P1 are present in M1 and one product of type P2 is present in R1. That is, the products in M1 need to be picked by R1, whereas the product in R1 must be placed in M1. This is not possible since both M1 and R1 are fully occupied. Hence, the FMS will deadlock when reaching this scenario. Similarly, the FMS deadlocks in the scenario on the right-hand side of Fig. 12. Here, M2 is fully occupied and the first product of type P2 has to be picked by R3. However, R3 is already occupied by a product of type P1, which has to move to M2.

In order to avoid such deadlock situations, we design a maximally permissive and nonblocking supervisor as described in Section 2.2. The supervisor automaton S for this case has 72 states and restricts the behavior of G in order to avoid deadlocks. Since the supervisor S is too large it cannot be displayed in this paper.

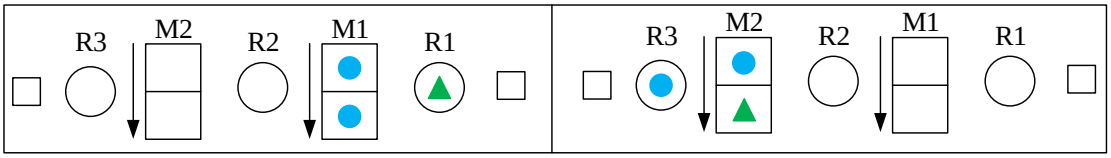

FIGURE 12. Model of machine M2.

We next illustrate the operation of the computed supervisor S and demonstrate the practicability of the proposed modeling technique. To this end, we compare the designed supervisor with a supervisor that is computed for a plant model without keeping track of the product order. In the modified plant, the robot models in Fig. 9 remain the same, whereas the machine models are replaced by the automata $\hat{G}_{\mathrm{M1}}$ and $\hat{G}_{\mathrm{M2}}$ in Fig. 13. Here, both products can leave the respective production component in the state $(\mathrm{P1},\mathrm{P2})$ independent of the arrival order of the products. In that case, the modified plant $\hat{G} = G_{\mathrm{R1}}||G_{\mathrm{R2}}||G_{\mathrm{R3}}||G_{\mathrm{M1}}||G_{\mathrm{M2}}$ has 24 956 states and the corresponding maximally permissive and nonblocking supervisor $\hat{S}$ has 564 states.

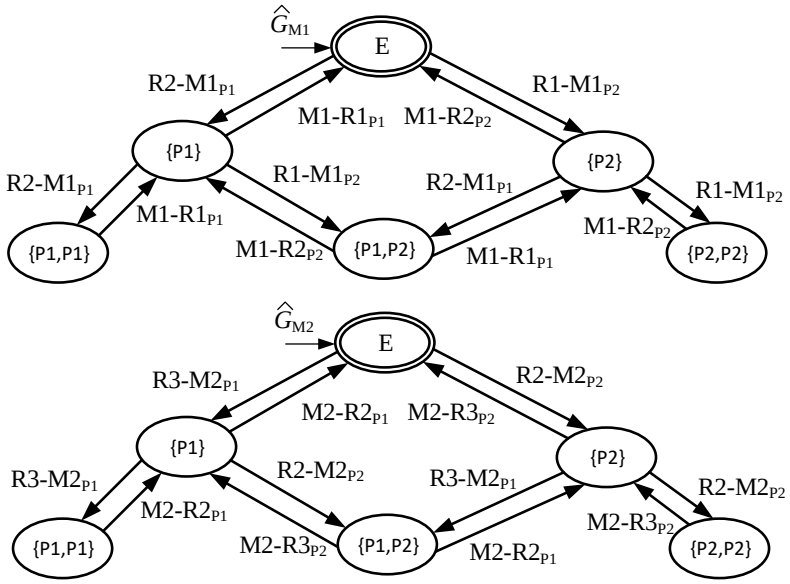

FIGURE 13. Model of the machines M1 and M2 without preserving the product order.

We next compare the operation of the supervisor S for the order-preserving model G and the supervisor $\hat{S}$ for the modified model $\hat{G}$ by following an example product path. Here, the left-hand side of Fig. 14 shows the operation of S and the right-hand side shows the operation of $\hat{S}$. In both cases, we consider two products of type P1 and P2. The product of type P2 step by step approaches M1 and enters this machine first. The product of type P1 also moves towards M1 and enters this machine after the other product. In Fig. 14, this state of the FMS is reached at the step that is shaded in gray. After this step, both supervisors may exhibit a different operation. Since

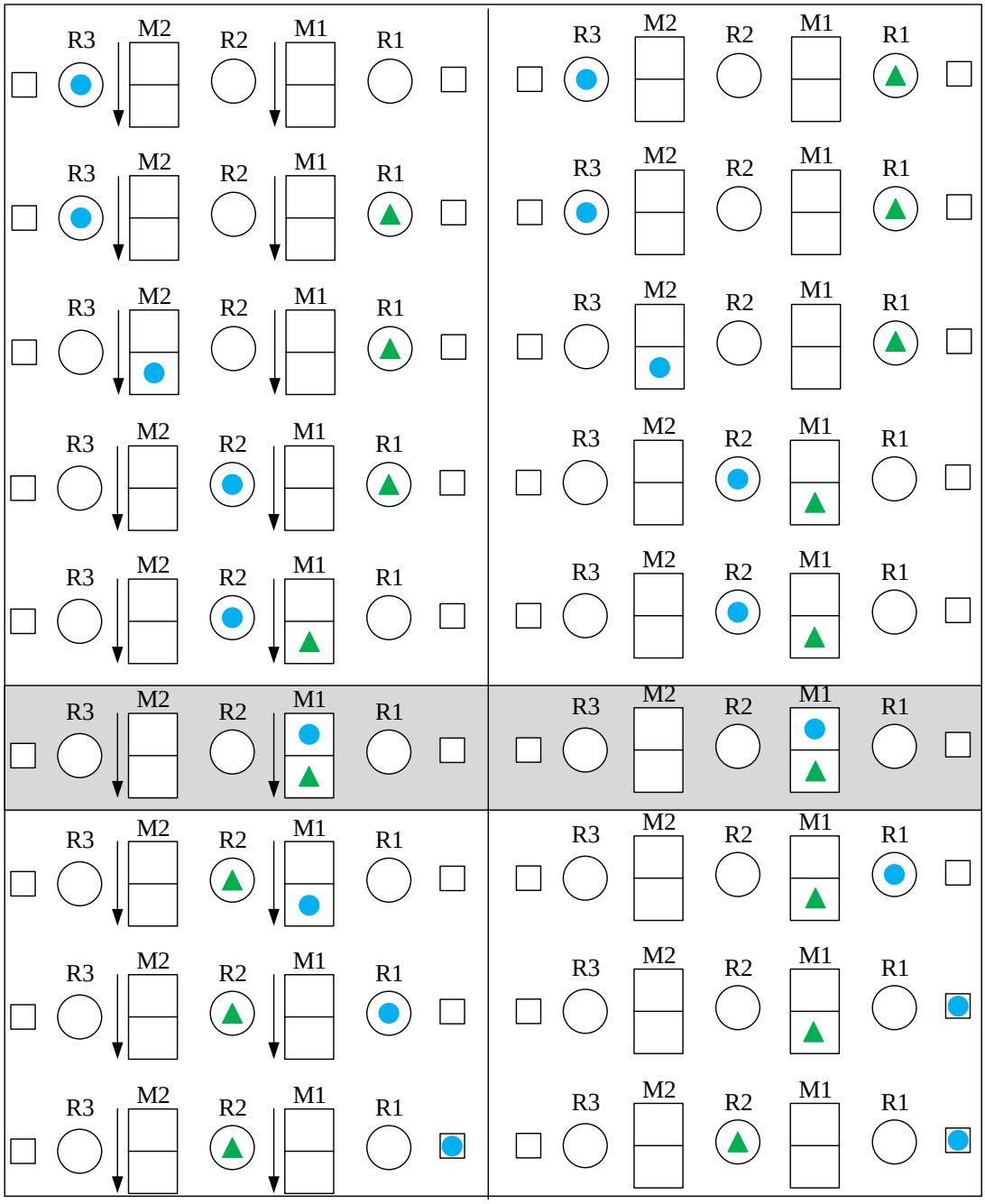

FIGURE 14. Example product path: Order-preserving case (left) and case with arbitrary product order (right).

S is order-preserving, it must be the case that the product of type P2 which entered M1 first must also leave M1 first. That is, this product next moves to R2. After that, the product of type P1 can move to R1 and leave the FMS. Differently, the product of type P1 can move to R1 first and leave the FMS if $\hat{S}$ is used since $\hat{S}$ does not keep track of the product order. Re-visiting the discussion in Section 3.2, we further note that the realization of the supervisor $\hat{S}$ needs additional information

compared to the realization of S. In the described situation with two products in M1, $\hat{S}$ needs to know which product leaves M1 first. This is only possible by installing a sensor for identifying products leaving M1. No such sensor is required when using the proposed order-preserving supervisor S.

6. Conclusions

The subject of this paper is the development of a new discrete event system (DES) model for the supervisory control of flexible manufacturing systems (FMS). This model considers that FMS consist of various production components such as machines and robots that exchange product among each other. In addition, the model accounts for the fact that FMS are able to manufacture different product types that potentially share various production components such as machines and robots. Different from the existing literature, the proposed model also addresses the case where a production component can hold multiple products and then processes these products sequentially. Specifically, if the capacity of a production component is greater than one, any product that enters the production component first is processed first and also leaves the production component first. Accordingly, the proposed model is order-preserving and keeps track of the different product types entering and leaving such production component. After presenting an algorithmic procedure for constructing such order-preserving model depending on the processed product types and the capacity of a production component, the paper demonstrates the practicability of the proposed method by an FMS application example, including a comparison to existing models that allow products to enter and leave a production component in an arbitrary order.

In future work, we will apply the proposed model in the context of modular and hierarchical control of large-scale FMS.

Acknowledgements

It is worth mentioning that the author Anas Nooruldeen is supported by a full scholarship for his PhD study by the Scientific and Technological Research Council of Turkey (TÜBİTAK) and would like to express his thanks and appreciation for this support.

References

[1] M. H. de Queiroz, J. E. R. Cury, W. M. Wonham, Multitasking Supervisory Control of Discrete-Event Systems, *Discrete Event Dynamic Systems*, **15**(4), (2005), 375—395.

[2] W. Chao, Y. Gan, W. M. Wonham, Z. Wang, Nonblocking Supervisory Control of Flexible Manufacturing Systems Based on State Tree Structures, *Formal Methods in Manufacturing Systems*, (2013).

[3] P. N. Pena, T. A. Costa, R. S. Silva, R. H. C. Takahashi, Control of Flexible Manufacturing Systems under model uncertainty using Supervisory Control Theory and evolutionary computation schedule synthesis, *Information Sciences*, **329**, (2016), 491–502.

[4] T. Sprock, C. Bock, L. F. McGinnis, Survey and classification of operational control problems in discrete event logistics systems (DELS), *International Journal of Production Research*, **57**(15-16), (2019), 5215–5238.

[5] R. Rl-Khalil, Z. Darwish, Flexible manufacturing systems performance in US automotive manufacturing plants: a case study, *Production Planning & Control*, **30**(1), (2019), 48–59.

[6] K. Schmidt, T. Moor and S. Perk, Nonblocking Hierarchical Control of Decentralized Discrete Event Systems, *IEEE Transactions on Automatic Control*, **53**(10), (2008), 2252–2265.

[7] L. Feng and W. M. Wonham, Supervisory Control Architecture for Discrete-Event Systems, *IEEE Transactions on Automatic Control*, **53**(6), (2008), 1449–1461.

[8] J. E. R. Cury, M. H. de Queiroz, G. Bouzon, M. Teixeira, Supervisory control of discrete event systems with distinguishers, *Automatica*, **56**, (2015), 93-104.

[9] C. G. Cassandras, S. Lafortune, Introduction to discrete event systems, Second edition, Springer, (2008).

[10] Z. Li, M. Zhou and N. Wu, A Survey and Comparison of Petri Net-Based Deadlock Prevention Policies for Flexible Manufacturing Systems, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, **2**(2), (2008), 173–188.

[11] M. Zhao, M. Uzam, Y. Hou, Near-optimal supervisory control of flexible manufacturing systems using divide-and-conquer iterative method, *Advances in Mechanical Engineering*, **8**(3), (2016), 1–17.

[12] Y. F. Hou, K. Barkaoui, Deadlock analysis and control based on Petri nets: A siphon approach review, *Advances in Mechanical Engineering*, **9**(5), (2017).

[13] Y. Li, L. Yin, Y. Chen, Z. Yu, N. Wu, Optimal Petri net supervisor synthesis for forbidden state problems using marking mask, *Information Sciences*, **505**, (2019), 183–197.

[14] K. Schmidt, C. Breindl, Maximally Permissive Hierarchical Control of Decentralized Discrete Event Systems, *IEEE Transactions on Automatic Control*, **56**(4), (2011), 723–737.

[15] K. Cai, W. M. Wonham, Supervisor Localization: A Top-Down Approach to Distributed Control of Discrete-Event Systems, *IEEE Transactions on Automatic Control*, **55**(3), (2010), 605–618.

[16] R. A. Williams, B. Benhabib, K. C. Smith, A hybrid supervisory control system for flexible manufacturing work-cells, *IEEE International Conference on Robotics and Automation*, **3**, (1994), 2551-2556.

[17] A. Nooruldeen, K. W. Schmidt, State Attraction Under Language Specification for the Reconfiguration of Discrete Event Systems, *IEEE Transactions on Automatic Control*, **60**(6), (2015), 1630–1634.

[18] K. W. Schmidt, Reconfigurability of behavioural specifications for manufacturing systems, *International Journal of Control*, **90**(12), (2017), 2605–2617.

[19] W. M. Wonham, Supervisory control of discrete-event systems, Lecture Notes, Department of Electrical and Computer Engineering, University of Toronto, (2010).

[20] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, H.V. Brussel, Reconfigurable Manufacturing Systems, *CIRP Annals - Manufacturing Technology*, **48**(2), (1999), 527-540.

[21] M.G. Mehrabi, A.G. Ulsoy, Y. Koren , Reconfigurable manufacturing systems: Key to future manufacturing, *Journal of Intelligent Manufacturing*, **11**(4), (2000), 403-419.