Hierarchical and Decentralized Multitasking Control of Discrete Event Systems

Klaus Schmidt, Max H. de Queiroz and José E. R. Cury

Abstract

In this paper, a hierarchical and decentralized approach for composite discrete-event systems (DES) that have to fulfill multiple tasks is elaborated. Colored marking generators that can distinguish classes of tasks are used as the system model, and a colored abstraction procedure as well as sufficient conditions for nonblocking and hierarchically consistent control are developed. It is shown that the computational complexity for supervisor computation is reduced. A flexible manufacturing system example demonstrates the efficiency of the approach.

This technical report provides the proofs that could not be elaborated in [1] due to space limitations.

I. Introduction

The supervisory control theory (SCT) introduced in [2] addresses the control of discrete-event systems (DES) that are modeled by a generator, whose marked states represent the completion of some control objective (task). Given an admissible system behavior represented by a language, a minimally restrictive supervisor can be computed algorithmically. This supervisor is designed to restrict the plant behavior such that it respects the admissible language, and ensures nonblocking behavior with respect to the marked states. While the admissible language can be viewed as a safety specification (ensuring that nothing ”bad” happens), nonblocking can be understood as a liveness specification, which ensures that the supervisor will not prevent the completion of a task (something good can happen). As situations where the liveness of multiple tasks is desired are common, the SCT framework has been extended to incorporate multiple tasks in [3]. Colored marking generators (CMGs) are introduced for the synthesis

K. Schmidt is with the Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg, Germany, klaus.schmidt@rt.eei.uni-erlangen.de

J.E.R. Cury and M.H. de Queiroz are with the Department of Automation and Systems, Federal University of Santa Catarina, Florianópolis SC 88040-900 Brazil, {max,cury}@das.ufsc.br

of a minimally restrictive supervisor that respects the admissible behavior and ensures the liveness of multiple tasks. Modular control in this framework is addressed in [4], where also a composition operation for CMGs is deduced from the synchronous composition operation for generators.

Although local supervisors for modular specifications and composite plants can be synthesized very efficiently using the modular approach, the resulting overall system need not be nonblocking. In the worst case, the nonblocking verification, and the synthesis of a coordinator to resolve possible conflicts can still require the compositon of the overall system model. Since this can again lead to exponential growth of the system state space, the use of hierarchical control ideas for multitasking DES is proposed in this paper.

Several hierarchical control approaches have been developed in recent years [5], [6], [7], [8], [9], [10], [11]. The approaches in [5], [6], [7], [8], [9], [10] employ the *natural projection* for hierarchical abstraction, and adopt the controllability properties of the low-level events for the high-level events. This allows for the application of standard supervisory control algorithms also for the high-level control, while it is possible that minimal restrictivness does not necessarily hold because of the choice of the high-level controllability properties.[1] A more powerful high-level control structure is introduced in [11] in order to guarantee *strong hierarchical consistency*. However, standard supervisory control cannot be applied for the high-level control, and the compositional property of system models is no longer valid.

In this paper, systems that are composed of different subsystems, and whose desired behavior involves multiple system tasks are considered. In order to use the system structure efficiently, multitasking control [3] is extended with hierarchical control ideas [5], [8], [9], [10] that preserve the compositional property. To this end, a multitasking version of the natural projection is defined, and sufficient conditions for nonblocking control are established for the resulting hierarchical and decentralized control architecture for multitasking DES.

The outline of the paper is as follows. Basic definitions are provided in Section II. Section III states the hierarchical control architecture for multitasking control and provides nonblocking control results. The method is illustrated by a detailed example in Section IV, and conslusions are given in Section V.

[1]Additional conditions to circumvent this issue are stated in [8], [9].

II. PRELIMINARIES

A. Multitasking Discrete-Event Systems

For a multitasking discrete-event system (MTDES), a color (label) can be associated to each class of task. Let Σ be the set of all events that can occur in the system and C be the set of all colors. Let Σ^* be the set of all finite strings of elements in Σ, including the empty string ϵ. A language L is a subset of Σ^*. $\overline{L}$ represents the prefix closure of L. Each color $c \in C$ is assigned to a language $L_c \in Pwr(\Sigma^*)$ (power set of Σ^*) that represents the set of all sequences of events in Σ that can complete a task of the respective class. Thus, the *colored behavior* of a MTDES can be modeled by the set $\Lambda_C \in Pwr(Pwr(\Sigma^*) \times C)$ given by $\Lambda_C := \{(L_c, c) | c \in C\}$.

For a colored behavior Λ_C, the language marked by $c \in C$ is defined by $L_c(\Lambda_C) := L$ such that $(L, c) \in \Lambda_C$. The language marked by $B \subseteq C$ is defined by $L_B(\Lambda_C) := \bigcup_{b \in B} L_b(\Lambda_C)$.

For $M_{B_1} \in Pwr(Pwr(\Sigma_1^*) \times B_1)$ and $N_{B_2} \in Pwr(Pwr(\Sigma_2^*) \times B_2)$, $M_{B_1} \subseteq N_{B_2}$ if $B_1 \subseteq B_2$ and $\forall b \in B_1$, $L_b(M_{B_1}) \subseteq L_b(N_{B_2})$.

The *synchronous composition* of M_{B_1} and N_{B_2} is

$$\begin{aligned} M_{B_1} || N_{B_2} := & \{(L_b(M_{B_1}) || L_b(N_{B_2}), b), \forall b \in B_1 \cap B_2\} \\ \cup & \{(L_b(M_{B_1}) || L_{B_2}(N_{B_2}), b), \forall b \in B_1 - B_2\} \\ \cup & \{(L_{B_1}(M_{B_1}) || L_b(N_{B_2}), b), \forall b \in B_2 - B_1\}. \end{aligned}$$

An MTDES can be modeled by a Moore automaton, whose outputs, represented by subsets of colors, define the classes of tasks that are completed after the corresponding strings. Such a *colored marking generator* (CMG), is formally defined by a 6-tuple $G = (Q, \Sigma, C, \delta, \chi, q_0)$, where Q is a set of states; Σ is a set of events; C is a set of colors; $\delta : Q \times \Sigma \to Q$ is a transition function; $\chi : Q \to Pwr(C)$ is a marking function; q_0 is the initial state.

For a CMG G, the *eligible event function* $\Gamma : Q \to Pwr(\Sigma)$ associates each state $q \in Q$ to a subset of Σ with all events that can occur in q. In order to extend δ to a partial function on $Q \times \Sigma^*$, recursively let $\delta(q, \varepsilon) = q$ and $\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$, whenever both $q' = \delta(q, s)$ and $\delta(q', \sigma)$ are defined. The *generated language* $L(G) := \{s \in \Sigma^* | \delta(q_0, s) \text{ is defined}\}$ of G, is the set of all finite event strings that can be reached from the initial state q_0.

The language *marked* by $c \in C$, is given by $L_c(G) := \{s \in L(G) | c \in \chi(\delta(q_0, s))\}$. For the color set B, $\emptyset \subset B \subseteq C$, the language marked by B is defined by $L_B(G) := \{s \in L(G) | B \cap \chi(\delta(q_0, s)) \neq \emptyset\}$. The colored behavior of a CMG G is given by $\Lambda_C(G) := \{(L_c(G), c) | c \in C\}$.

A formal definition of the *synchronous composition* $G_1||G_2$ of two CMGs G_1 and G_2 is given in [3]. Note that $L(G_1||G_2) = L(G_1)||L(G_2)$ and $\Lambda_C(G_1||G_2) = \Lambda_C(G_1)||\Lambda_C(G_2)$.

Given a nonempty subset of colors B, a CMG G is *strongly nonblocking* w.r.t. B, if $\forall b \in B$, $L(G) = \overline{L_b(G)}$, that is, if any generated string can be completed (not necessarily in the same way) to a task of all the classes represented by colors of B. A colored behavior $\Lambda_C \in Pwr(Pwr(\Sigma^*) \times C)$ is strongly nonblocking w.r.t. $B \subseteq C$ when $\forall b \in B$, $\overline{L_b(\Lambda_C)} = \overline{L_C(\Lambda_C)}$. Furthermore, it is shown in [3] that the *maximal strongly nonblocking behavior* $SupSNB(\Lambda_C, B)$ contained in Λ_C for a color set $B \subseteq C$ exists.

B. Multitasking Supervisory Control

Let a MTDES be modeled by a colored marking generator $G = (Q, \Sigma, C, \delta, \chi, q_0)$, with eligible event function Γ, whose alphabet is partitioned into controllable events $\Sigma_{\rm c}$ and uncontrollable events $\Sigma_{\rm u}$. Let D be a set of important tasks for which liveness (strong nonblocking) is required. Let the specification be given by a colored behavior $A_D \in Pwr(Pwr(\Sigma^*) \times D)$ such that $\forall d \in D \cap C$, $\emptyset \subset L_d(A_D) \subseteq L_d(G)$, and $\forall d \in E := D - C$, $\emptyset \subset L_d(A_D) \subseteq L(G)$.

A *coloring supervisor* $S : L(G) \to Pwr(\Sigma) \times Pwr(E)$ is a mapping that associates to each sequence of events of the plant a set of enabled events and a set of new colors (of E) representing completed tasks.

For $S(s) = (\gamma, \mu)$, let $\mathcal{R}(S(s)) = \gamma$ and $\mathcal{I}(S(s)) = \mu$. The events that can occur in S/G after the occurrence of a string $s \in L(G)$ are given by $\mathcal{R}(S(s)) \cap \Gamma(\delta(q_0, s))$. A string $s \in L(S/G)$ is marked by a color $c \in C$ if $s \in L_c(G)$ or by a color $e \in E$ if $e \in \mathcal{I}(S(s))$. A coloring supervisor S is *admissible* if $\forall s \in L(G)$, $\Sigma_{\rm u} \cap \Gamma(\delta(q_0, s)) \subseteq \mathcal{R}(S(s))$.

A supervisor S is strongly nonblocking w.r.t D if $\forall d \in D$, $\overline{L_d(S/G)} = L(S/G)$.

Theorem 1 ([3]): Necessary and sufficient conditions for the existence of an admissible coloring supervisor S strongly nonblocking w.r.t. D such that $\Lambda_D(S/G) = A_D$ and $L(S/G) = \overline{L_D(A_D)}$ are:

1) controllability: $\overline{L_D(A_D)}\Sigma_{\rm u} \cap L(G) \subseteq \overline{L_D(A_D)}$;
2) D-closure: $\forall d \in (D \cap C)$, $L_d(A_D) = \overline{L_d(A_D)} \cap L_d(G)$;
3) strong nonblocking of A_D w.r.t. D.

In [3], it is also proven that the supremal controllable and strongly nonblocking colored behavior contained in A_D, named $\mathrm{SupCSNB}(A_D, G, D)$, can be computed with complexity polynomial in the number of states of the model.

Remark 1: Note that the standard Ramadge/Wonham supervisory control theory as introduced in [2] can be described as a special case of the multitasking supervisory control by allowing for CMGs G

with only one color, and by requiring an empty color set E of the supervisor S. In that case, G is a finite automaton with the marked language $L_{\mathrm{m}}(G)$, the supervisor is a map $S : L(G) \rightarrow Pwr(\Sigma)$, and the supremal controllable and nonblocking sublanguage $SupCNB(L_{\mathrm{m}}(G), K)$ can be computed for $K \subseteq L_{\mathrm{m}}(G)$.

C. Hierarchical and Decentralized Control

In the next section, hierarchical and decentralized multitasking control is introduced. As the hierarchical control ideas are based on results formulated in the Ramadge/Wonham supervisory control framework, the original hierarchical and decentralized control approach as introduced in [5], and extended in [7], [8], [9], [10], is described first.

As a system model, *composite* DES represented by finite automata G_i, $i = 1, \ldots, n$, over the corresponding alphabets $\Sigma_i = \Sigma_{i,\mathrm{u}} \dot{\cup} \Sigma_{i,\mathrm{c}}$ are used. Here, $\Sigma_{i,\mathrm{u}}$ and $\Sigma_{i,\mathrm{c}}$ denote the uncontrollable and the controllable events, respectively. It is assumed that each subsystem shares the events $\Sigma_{i,\mathrm{s}} := \bigcup_{k=1,k\neq i}^{n} (\Sigma_i \cap \Sigma_k)$ with other subsystems. The global set of *shared events* is thus given by $\Sigma_{\mathrm{s}} = \bigcup_{i=1}^{n} \Sigma_{i,\mathrm{s}}$.

The overall system model is $G := \|_{i=1}^{n} G_i$ over the alphabet $\Sigma := \bigcup_{i=1}^{n} \Sigma_i$. Moreover, it is required that the components that share an event agree on the control status of this event, i.e. $\forall i, k,\ i \neq k,\ \Sigma_{i,\mathrm{u}} \cap \Sigma_{k,\mathrm{c}} = \emptyset$. Under this hypothesis, it holds that $\Sigma_{\mathrm{u}} = \bigcup_{i=1}^{n} \Sigma_{i,\mathrm{u}}$ and $\Sigma_{\mathrm{c}} = \bigcup_i \Sigma_{i,\mathrm{c}}$.

The suggested hierarchical and decentralized control approach as introduced in [5], respects the composite system structure in both the abstraction process and the low-level supervisor implementation (see Figure 1).

For hierarchical abstraction, an alphabet $\Sigma_0 \subseteq \Sigma$ is chosen that contains the shared events, i.e. $\Sigma_{\mathrm{s}} \subseteq \Sigma_0$. Using the natural projection $p_0 : \Sigma^* \rightarrow \Sigma_0^*$, the high-level plant is defined as a finite automaton G_0 over Σ_0 such that $L(G_0) = p_0(L(G))$ and $L_{\mathrm{m}}(G_0) = p_0(L_{\mathrm{m}}(G))$. The choice of $\Sigma_{\mathrm{s}} \subseteq \Sigma_0$ facilitates the computation of G_0. High-level subsystems $G_{i,0}$ can be defined using the alphabets $\Sigma_{i,0} := \Sigma_i \cap \Sigma_0$ and the natural projections $p_{\Sigma_i \rightarrow \Sigma_{i,0}} : \Sigma_i^* \rightarrow \Sigma_{i,0}^*$, $i = 1, \ldots, n$ such that $L(G_{i,0}) = p_{\Sigma_i \rightarrow \Sigma_{i,0}}(L(G_i))$ and $L_{\mathrm{m}}(G_{i,0}) = p_{\Sigma_i \rightarrow \Sigma_{i,0}}(L_{\mathrm{m}}(G_i))$. Then, the high-level plant can be computed as follows.

Lemma 1 (High Level Plant [5]): Assume the notation from above with $\Sigma_{\mathrm{s}} \subseteq \Sigma_0$. Then

$$\begin{aligned} L(G_0) &= p_0(\|_{i=1}^{n} L(G_i)) = \|_{i=1}^{n} L(G_{i,0}) \\ L_{\mathrm{m}}(G_0) &= p_0(\|_{i=1}^{n} L_{\mathrm{m}}(G_i)) = \|_{i=1}^{n} L_{\mathrm{m}}(G_{i,0}). \end{aligned}$$

Adopting the controllability properties from the low level, i.e. $\Sigma_{\mathrm{u},0} := \Sigma_{\mathrm{u}} \cap \Sigma_0$ and $\Sigma_{\mathrm{c},0} := \Sigma_{\mathrm{c}} \cap \Sigma_0$, the high-level plant is again given as a finite automaton in the Ramadge/Wonham framework. Supervisory

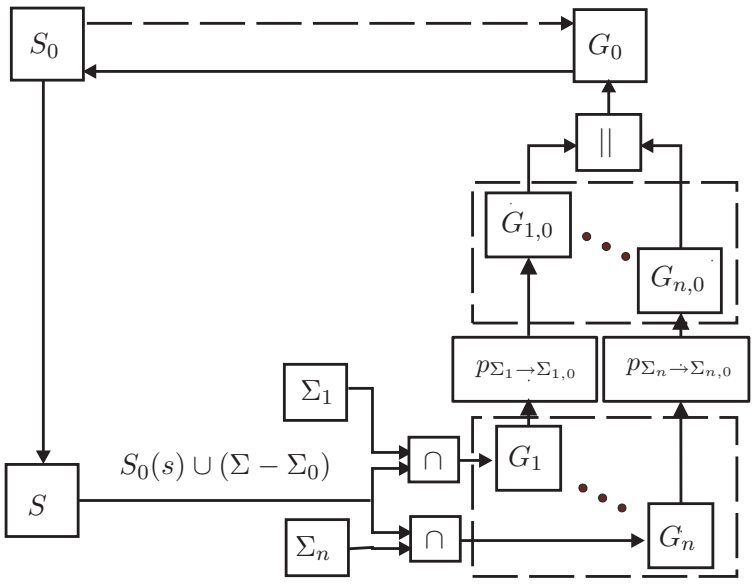

Fig. 1. Hierarchical and decentralized control architecture

control for a specification $K \subseteq L_{\mathrm{m}}(G_0)$ can be applied to determine a nonblocking high-level supervisor $S_0 : L(G_0) \to Pwr(\Sigma_0)$ such that $L_{\mathrm{m}}(S_0/G_0) = SupCNB(L_{\mathrm{m}}(G_0), K)$.

The control action of the high-level supervisor S_0 is then implemented by the low-level supervisor $S : \Sigma^* \to Pwr(\Sigma)$ such that for any $s \in L(G)$, $S(s) = S_0(p_0(s)) \cup (\Sigma - \Sigma_0)$. Consequently, each low-level subsystem G_i observes the control action $S(s) \cap \Sigma_i$ as depicted in Figure 1.

Using this control architecture, *hierarchical consistency* is already guaranteed, i.e. it holds that $p_0(L(S/G)) = L(S_0/G_0)$ [5]. To ensure nonblocking control, additional conditions are required. In this paper, the *observer condition* as introduced in [12] is employed as a sufficient condition for nonblocking control according to [8], [9], [10].

Definition 1 (Observer): Let $L' \subseteq L \subseteq \Sigma^*$ be languages and let $p_0 : \Sigma^* \to \Sigma_0^*$ be the natural projection for $\Sigma_0 \subseteq \Sigma$. p_0 is an L'-observer (w.r.t. L) iff for all $s \in \overline{L}$ and $t \in \Sigma_0^*$

$$p_0(s)t \in p_0(L') \Rightarrow \exists u \in \Sigma^* \text{ s.t. } su \in L' \wedge p_0(su) = p_0(s)t.$$

The described control architecture where $p_{\Sigma_i\to\Sigma_{i,0}}$ is an $L_{\mathrm{m}}(G_i)$-observer for $i = 1, \ldots, n$ is nonblocking.

Theorem 2 (Nonblocking Control): Let G_i and $G_{i,0}$, $i = 1, \ldots, n$, and S_0 and S be given as above. If $p_{\Sigma_i\to\Sigma_{i,0}}$ is an $L_{\mathrm{m}}(G_i)$-observer (w.r.t. $L(G_i)$) for $i = 1, \ldots, n$, then

(i) p_0 is an $L_{\rm m}(G)$-observer (w.r.t. $L(G)$) [13],

(ii) the closed loop is nonblocking: $\overline{L_{\rm m}(S/G)} = L(S/G)$.

The approach is computationally efficient as the composition of the overall low-level plant is avoided by composing only the smaller high-level subsystems. Note that the high-level system has indeed a smaller state space than the low-level plant, as the observer property implies that the minimal generator for $p_0(L(G))$ has maximally as many states as the minimal generator for $L(G)$, [14]. Additionally, [15] provides a method to determine the high-level alphabets $\Sigma_{i,0}$ such that each $p_{\Sigma_i \to \Sigma_{i,0}}$ is an $L_{\rm m}(G_i)$-observer. Also observe that the high-level closed-loop is again a finite automaton that can be used as the low-level model for further hierarchical abstraction in a multi-level hierarchy.

III. Multitasking Hierarchical and Decentralized Control

The computational efficiency of hierarchical and decentralized control, and the ability to specify multiple control objectives is now combined in a hierarchical and decentralized control architecture for multitasking supervisory control. Analogous to Section II-C, it is assumed that the low-level plant is given as a set G_i, $i = 1, \ldots, n$ of colored marking generators with the respective color set C_i, and the overall plant is $G = ||_{i=1}^n G_i$ with the color set $C := \bigcup_{i=1}^n C_i$.

First, the natural projection p_0 is extended to colored behaviors Λ_C.

Definition 2 (Colored Natural Projection): Let $\Lambda_C \in Pwr(Pwr(\Sigma^*) \times C)$ be a colored behavior, and assume $\Sigma_0 \subseteq \Sigma$ with the natural projection $p_0 : \Sigma^* \to \Sigma_0^*$. The *colored natural projection* $m_0 : Pwr(Pwr(\Sigma^*) \times C) \to Pwr(Pwr(\Sigma_0^*) \times C)$ is defined such that

$$L_c(m_0(\Lambda_C)) = p_0(L_c(\Lambda_C)), \text{ for all } c \in C.$$

Accordingly, the colored natural projections $m_{\Sigma_i \to \Sigma_{i,0}}$ are defined, and the high-level subsystems evaluate to $G_{i,0}$, $i = 1, \ldots, n$, where $L(G_{i,0}) = p_{\Sigma_i \to \Sigma_{i,0}}(L(G_i))$ and $\Lambda_C(G_{i,0}) = m_{\Sigma_i \to \Sigma_{i,0}}(\Lambda_C(G_i))$.

Using the colored natural projection in Definition 2 with $\Sigma_{\rm s} \subseteq \Sigma_0$, the high-level plant G_0 such that $L(G_0) = p_0(L(G))$ and $\Lambda_C(G_0) = m_0(\Lambda_C(G))$, can again be computed by composing the high-level subsystems.

Lemma 2: Assume the notation from above with $\Sigma_{\rm s} \subseteq \Sigma_0$. Then

$$\begin{aligned} L(G_0) &= p_0(||_{i=1}^n L(G_i)) = ||_{i=1}^n L(G_{i,0}) \\ \Lambda_C(G_0) &= m_0(||_{i=1}^n \Lambda_C(G_i)) = ||_{i=1}^n \Lambda_C(G_{i,0}) \end{aligned}$$

Proof: Lemma 4 in the appendix implies that $L(G_0) = p_0(||_{i=1}^n G_i) = ||_{i=1}^n p_{\Sigma_i \to \Sigma_{i,0}}(L(G_i)) = ||_{i=1}^n L(G_{i,0})$.

Furthermore, it holds for all $c \in C$ that $L_c(\Lambda_C(G_0)) = L_c(m_0((||_{i=1}^n \Lambda_C(G_i)))) = p_0(L_c(||_{i=1}^n \Lambda_C(G_i))) = p_0(||_{i,c\in C_i} L_c(\Lambda_C(G_i))) = ||_{i,c\in C_i} p_{\Sigma_i \to \Sigma_{i,0}}(L_c(\Lambda_C(G_i))) = ||_{i,c\in C_i} L_c(\Lambda_C(G_{i,0})) = L_c(||_{i=1}^n \Lambda_C(G_{i,0}))$. Thus, $\Lambda_C(G_0) = ||_{i=1}^n \Lambda_C(G_{i,0})$. ■

According to Lemma 2, the high-level colored marking generator G_0 can be computed as shown in Figure 1. Given a coloring behavior $A_D \in Pwr(Pwr(\Sigma_0^*) \times D)$ as a high-level specification, a coloring supervisor $S_0 : L(G_0) \to Pwr(\Sigma_0) \times Pwr(E)$ with $E = D - C$ can be computed such that S_0 realizes $SupCSNB(A_D, G_0, D)$. Different from the low-level implementation in Section II-C, the set of new colors E introduced by S_0 has to be considered. The control action of the low-level supervisor $S : L(G) \to Pwr(\Sigma) \times Pwr(E)$ is thus defined for each $s \in L(G)$ as

$$S(s) := (S_0(p_0(s)) \cup (\Sigma - \Sigma_0), \mathcal{I}(S_0(p_0(s)))). \tag{1}$$

The control action after a string $s \in L(G)$ observed by each subsystem is then

$$(\mathcal{R}(S(s)) \cap \Sigma_i, \mathcal{I}(S(s)) \cap (C_i \cup E)).$$

As hierarchical consistency does not depend on coloring, and the supervisor implementation without coloring $\mathcal{R}(S)$ is equivalent to the standard implementation in Section II-C, hierarchical consistency also follows for the supervisors S and S_0 in this section, i.e. $p_0(L(S/G)) = L(S_0/G_0)$.

To address strongly nonblocking control for the proposed control architecture, an analogous condition to the observer condition is required. The extension of the $L_{\mathrm{m}}(G)$-observer to the colored case replaces the marked language $L_{\mathrm{m}}(G)$ with the colored behavior $\Lambda_C(G)$, and the natural projection p_0 by the colored natural projection m_0.

Definition 3 (Colored Observer): Let $L \subseteq \Sigma^*$ be a language and let $\Lambda_C \in Pwr(Pwr(\Sigma^*) \times C)$ be a coloring behavior with $L_C(\Lambda_C) \subseteq L$. Also let $p_0 : \Sigma^* \to \Sigma_0^*$ be the natural projection, and $m_0 : Pwr(Pwr(\Sigma^*) \times C) \to Pwr(Pwr(\Sigma_0^*) \times C)$ be the colored natural projection for $\Sigma_0 \subseteq \Sigma$. m_0 is a Λ_C-observer (w.r.t. L) iff for each $c \in C$, p_0 is an $L_c(\Lambda_C)$-observer (w.r.t. L).

Requiring that $m_{\Sigma_i \to \Sigma_{i,0}}$ is a $\Lambda_C(G_i)$-observer for $i = 1, \ldots, n$ is sufficient for strongly nonblocking control.

Theorem 3: Assume that G_i, $G_{i,0}$, and $m_{\Sigma_i \to \Sigma_{i,0}}$, $i = 1, \ldots, n$ are defined as above. Also let S_0 be a strongly nonblocking coloring high-level supervisor with a low-level supervisor S as in Equation (1). If $m_{\Sigma_i \to \Sigma_{i,0}}$ is a $\Lambda_C(G_i)$-observer (w.r.t. $L(G_i)$) for all $i = 1, \ldots, n$, then the overall closed loop is strongly nonblocking, i.e., for all $c \in C$

$$\overline{L_c(S/G)} = L(S/G).$$

To prove Theorem 3, first a result similar to Theorem 2 (i) is derived.

Lemma 3: Let the CMGs G_i be given as above and assume that the colored natural projections $m_{\Sigma_i \to \Sigma_{i,0}}$, $i = 1, \ldots, n$ are $\Lambda_C(G_i)$-observers. Then $m_0 : \Sigma^* \to \Sigma_0^*$ is a $\Lambda_C(G)$-observer.

Proof: According to Definition 3, $p_{\Sigma_i \to \Sigma_{i,0}}$ is an $L_c(\Lambda_C(G_i))$-observer for all i such that $c \in C_i$. Applying Theorem 2 (i) and the definition of the synchronous compositon, p_0 is an $L_c(\Lambda_C(G))$-observer. As c was chosen arbitrarily, p_0 is an $L_c(\Lambda_C(G))$-observer for each $c \in C$. With Definition 3, m_0 is a $\Lambda_C(G)$-observer. ■

Using this result, Theorem 3 can be proven.

Proof: It has to be shown that for all $s \in L(S/G)$, it holds that $s \in \overline{L_c(S/G)}$ for each color $c \in C$.

Assume $s \in L(S/G)$ (note that such s exists, as $\varepsilon \in L(S/G)$ due to the definiton of S). Then for each $c \in C$,

$$p_0(s) \in L(S_0/G_0) \wedge \exists t \in \Sigma_0^* \text{ s.t } p_0(s)t \in L_c(S_0/G_0),$$

as S_0 is strongly nonblocking. Using the fact that p_0 is an $L_c(G)$-observer according to Lemma 3, and the supervisor implementation with $\mathcal{R}(S(s)) = \mathcal{R}(S_0(p_0(s))) \cup (\Sigma - \Sigma_0)$, Lemma 5 in the appendix implies that

$$\exists u \in \Sigma^* \text{ s.t. } su \in L_c(S/G) \wedge p_0(su) = p_0(s)t.$$

Thus, $s \in \overline{L_c(S/G)}$ for all $c \in C$, which proves that $\overline{L_c(S/G)} = L(S/G)$ for all $c \in C$. ■

Note that the polynomial time algorithm to determine the alphabets for the hierarchical abstraction in [15] can be extended to the case with colored marking. This also implies that again, the high-level CMG G_0 never has a larger state space than the low-level CMG G.

IV. Flexible Manufacturing System Example

The hypothetical Flexible Manufacturing System (FMS) in Figure 2 as introduced in [3] is studied. It generates two types of products from raw blocks and raw pegs: a block with a conical pin (Product A) and a block with a cylindrical painted pin (Product B). The FMS consists of eight devices: three conveyors C1, C2 and C3, a Mill (M), a Lathe (L), a Robot (R), a Painting Device (PD), and an Assembly Machine (AM). The devices are connected through buffers Bi, $i = 1, \ldots, 8$, with capacity for one part. The arrows in Figure 2 indicate the flow of unfinished parts through the FMS. Raw blocks enter C1 and reach B1. Raw pegs enter C2 and arrive in B2. The Robot picks a raw block from B1 and places it into B3 or moves raw pegs from B2 to B4. The Mill starts processing a block from B3 and returns a geometrically shaped part with a hole on top. The Lathe can make two types of pins with the peg from B4: a conical

pin or a cylindrical pin. Then the Robot moves a finished block from B3 to B5, moves a conical pin from B4 to B6 or moves a cylindrical pin from B4 to B7. C3 transports the pin from B7 to B8, where it is painted, and takes it back to B7. Finally, the AM creates a Product A (Product B) by assembling a block from B5 and a conical pin from B6 (cylindrical pin from B7).

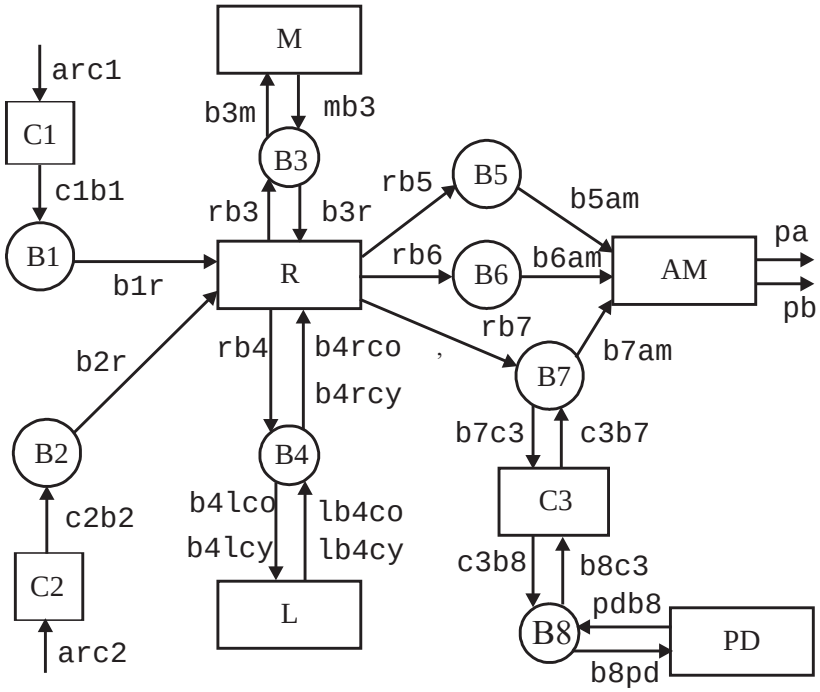

Fig. 2. Flexible Manufacturing System (FMS)

The open-loop behavior of the FMS is modeled by the set of eight asynchronous CMGs in Figure 3, where the controllable events are marked with ticks. The manufacture of one Product A and of one Product B is respectively indicated by the tasks a and b in the model for the AM.

Each restriction can be modularly expressed by a generic specification, which is a colored behavior defined on a particular subset of events from the global alphabet of the composite plant. The generic specifications $M_{\mathrm{B}i}$, $i = 1, \ldots, 8$ for avoiding overflow and underflow in the buffers Bi, $i = 1, \ldots, 8$, respectively, are generated by the CMGs in Figure 4. The tasks e_1 and e_2 in M_{B1} and M_{B2} specify that the buffers B1 and B2 always have to be able to reach the empty state. M_{B3} and M_{B4} state that simultaneous operation of the Lathe and the Mill is always possible (states with color o), and both buffers can always become empty (colors e_3 and e_4). Finally, the task e indicates that the buffers B7 and B8

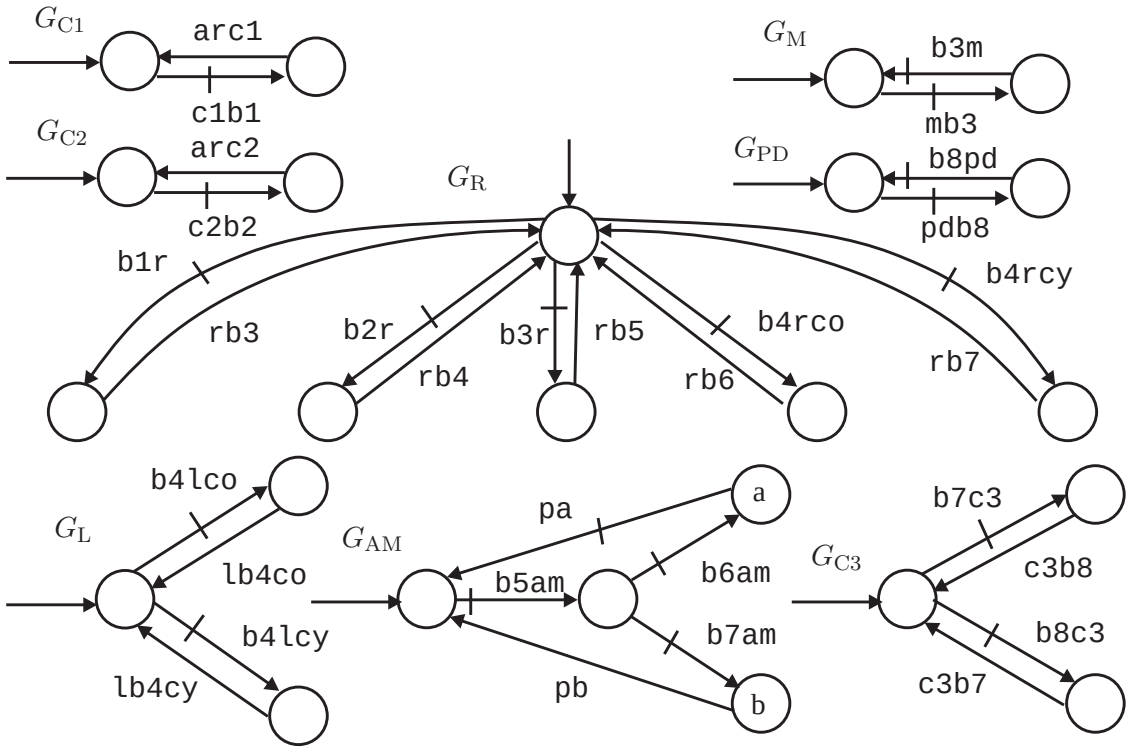

Fig. 3. Flexible Manufacturing System (FMS)

can reach their empty state simultaneously, while no pin is either in PD or in C3. The synchronous composition $M := ||_{i=1}^{8} M_{\mathrm{B}i}$ of all generic specifications has the color set $E = \{\mathrm{e}_1, \mathrm{e}_2, \mathrm{e}_3, \mathrm{e}_4, \mathrm{o}, \mathrm{e}\}$. The global specification is then obtained from $A_D = \Lambda_C(M)||\Lambda_C(G)$. Therefore, in order to respect all the specifications defined in Section II, the controlled system must respect A_D, and be strongly nonblocking with respect to the color set $D = \{\mathrm{a}, \mathrm{b}, \mathrm{e}_1, \mathrm{e}_2, \mathrm{e}_3, \mathrm{e}_4, \mathrm{o}, \mathrm{e}\}$.

The synchronous composition of the eight CMG leads to a CMG $G = (Q, \Sigma, C, \delta, \chi, q_0)$ with 3456 states and color set $C = \{a, b\}$, and it has been shown in [3] that the optimal colored behavior $SupCSNB(A_D, G, D)$ can be guaranteed by a monolithic supervisor with 45504 states.

In order to reduce the computational effort as well as the size of the supervisor, the hierarchical and decentralized approach in Section III is applied. First, the part of the plant that corresponds to the buffer specifications $M_{\mathrm{B}i}$, $i = 5, \ldots, 8$ is considered. Local supervisors are computed using monolithic multitasking supervisory control: for example, G_5 represents the resulting closed-loop behavior from $SupCSNB(\Lambda_C(G_{\mathrm{AM}}||G_{\mathrm{R}}||M_{\mathrm{B5}}), G_{\mathrm{AM}}||G_{\mathrm{R}}, \{\})$ for the subsystem $G_{\mathrm{AM}}||G_{\mathrm{R}}$ and the specification $\Lambda_C(G_{\mathrm{AM}}||G_{\mathrm{R}}||M_{\mathrm{B5}})$. The remaining subsystems with their respective specifications are summarized in

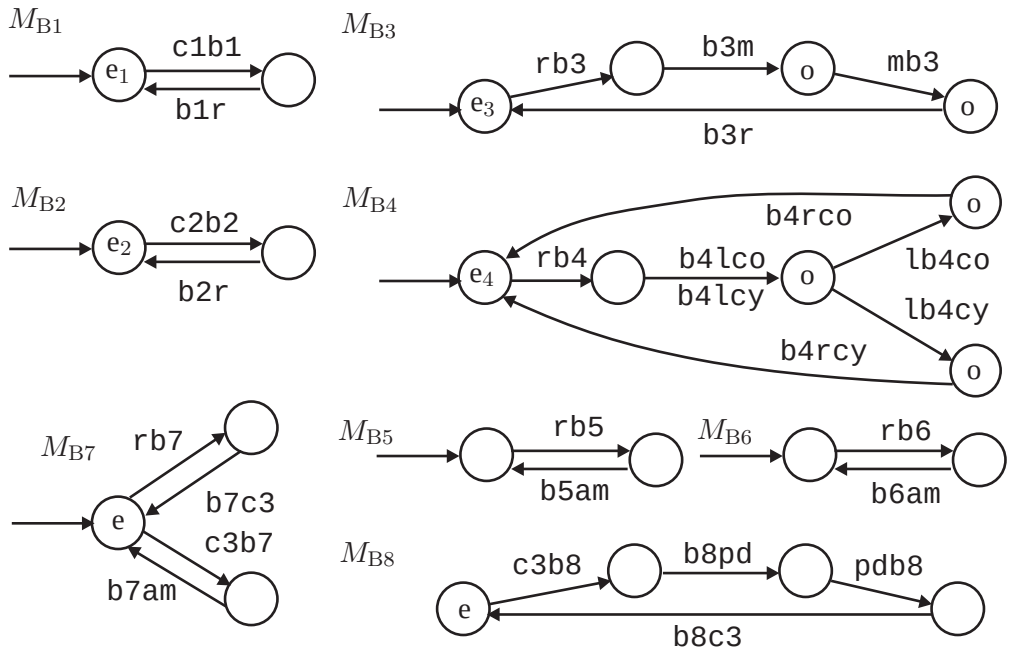

Fig. 4. Buffer specifications for the FMS

Table I.

TABLE I

SUBSYSTEMS G_i, $i = 5, \ldots, 8$ OF THE FMS

closed loop	subsystem	specification
G_5	$G_{\mathrm{AM}}\|\|G_{\mathrm{R}}$	$\Lambda_C(G_{\mathrm{AM}}\|\|G_{\mathrm{R}}\|\|M_{\mathrm{B5}})$
G_6	$G_{\mathrm{AM}}\|\|G_{\mathrm{R}}$	$\Lambda_C(G_{\mathrm{AM}}\|\|G_{\mathrm{R}}\|\|M_{\mathrm{B6}})$
G_7	$G_{\mathrm{C3}}\|\|G_{\mathrm{AM}}\|\|G_{\mathrm{R}}$	$\Lambda_C(G_{\mathrm{C3}}\|\|G_{\mathrm{AM}}\|\|G_{\mathrm{R}}\|\|M_{\mathrm{B7}})$
G_8	$G_{\mathrm{C3}}\|\|G_{\mathrm{PD}}$	$\Lambda_C(G_{\mathrm{C3}}\|\|G_{\mathrm{PD}}\|\|M_{\mathrm{B8}})$

The subsystems G_i, $i = 5, \ldots, 8$ fulfill the respective local buffer specifications (the automata representations of the corresponding reduced supervisors R_i, $i = 5, \ldots, 8$ are shown in Figure 7). However, it is not guaranteed that the joint behavior of these subsystems is strongly nonblocking. The possibly blocking behavior is now resolved using the hierarchical approach in Section III, where the closed-loop subsystems G_i, $i = 5, \ldots, 8$ serve as the low-level models. The computation of the high-level plant G_0 involves the high-level alphabet $\Sigma_0 = \Sigma_{\mathrm{R}} \cup \Sigma_{\mathrm{C3}} \cup \Sigma_{\mathrm{AM}}$ and the color set $C_0 = \{\mathrm{e}, \mathrm{a}, \mathrm{b}\}$. Note that Σ_0

contains the shared events between the different subsystems according to Lemma 2, and that it can be verified that the colored natural projection on Σ_0 fulfills the colored observer property in Definition 3.

In the next step, a strongly nonblocking supervisor S_0 is synthesized for $SupSNB(\Lambda_C(G_0), C_0)$. The automata representation R_0 of the corresponding reduced supervisor is shown in Figure 7. With the result in Theorem 3, the low-level implementation of S_0 guarantees strongly nonblocking system behavior. The hierarchical control architecture including the intermediate subsystem abstractions $G_{i,0}$, $i = 5, \ldots, 8$ is depicted in Figure 5, where each CMG is shown with its respective state count. Note that the supervisor S_0 just acts on the subsystems G_i, $i = 5, \ldots, 8$.

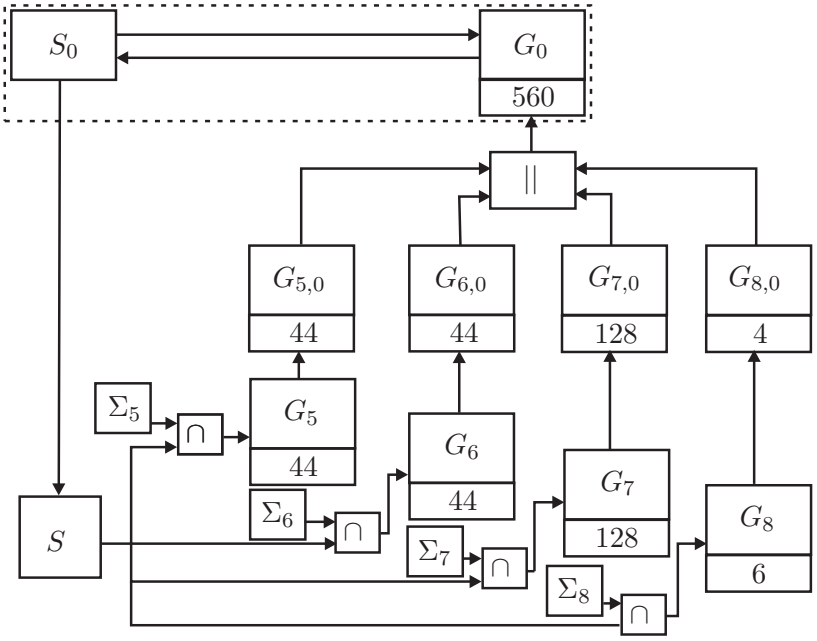

Fig. 5. Hierarchical Structure for the subsystems G_i, $i = 5, \ldots, 8$

The resulting closed-loop CMG S_0/G_0 can now be used for further controller synthesis to address possible conflict of this subsystem with the part of the FMS that has not been considered so far. To this end, $G_9 := S_0/G_0$ is defined, and the subsystems G_i, $i = 1, \ldots, 4$ are computed as closed-loop CMGs for the respective buffer specifications $M_{\mathrm{B}i}$, $i = 1, \ldots, 4$ as shown in Table II. Thus, the low-level model consists of G_i, $i = 1, \ldots, 4, 9$ (see Figure 6). Hierarchical abstraction is performed with the alphabet $\hat{\Sigma}_0 = \Sigma_{\mathrm{R}} \cup \{$ `c1b1`, `c2b2`, `b3m`, `b4lcy`, `b4lco`, `b5am`, `b6am`, `a`, `b`, `c3b7`$\}$ which is chosen such that the shared events of the subsystems (events of the robot Σ_{R}) are contained in $\hat{\Sigma}_0$, and such that the colored natural projection $m_{\Sigma_i \to \Sigma_{i,0}}$ is a $\Lambda_C(G_i)$-observer for $i = 1, \ldots, 4, 9$. It could be verified that the

high-level model $\hat{G}_0$ in Figure 6 is already strongly nonblocking, which implies that the overall closed-loop flexible manufacturing system is strongly nonblocking according to Theorem 3 without introducing an additional supervisor for $\hat{G}_0$.

TABLE II

SUBSYSTEMS G_i, $i = 1, \ldots, 4$ OF THE FMS

closed loop	subsystem	specification
G_1	$G_{\mathrm{C1}}\|\|G_{\mathrm{R}}$	$\Lambda_C(G_{\mathrm{C1}}\|\|G_{\mathrm{R}}\|\|M_{\mathrm{B1}})$
G_2	$G_{\mathrm{C2}}\|\|G_{\mathrm{R}}$	$\Lambda_C(G_{\mathrm{C2}}\|\|G_{\mathrm{R}}\|\|M_{\mathrm{B2}})$
G_3	$G_{\mathrm{M}}\|\|G_{\mathrm{R}}$	$\Lambda_C(G_{\mathrm{M}}\|\|G_{\mathrm{R}}\|\|M_{\mathrm{B3}})$
G_4	$G_{\mathrm{L}}\|\|G_{\mathrm{R}}$	$\Lambda_C(G_{\mathrm{L}}\|\|G_{\mathrm{R}}\|\|M_{\mathrm{B4}})$

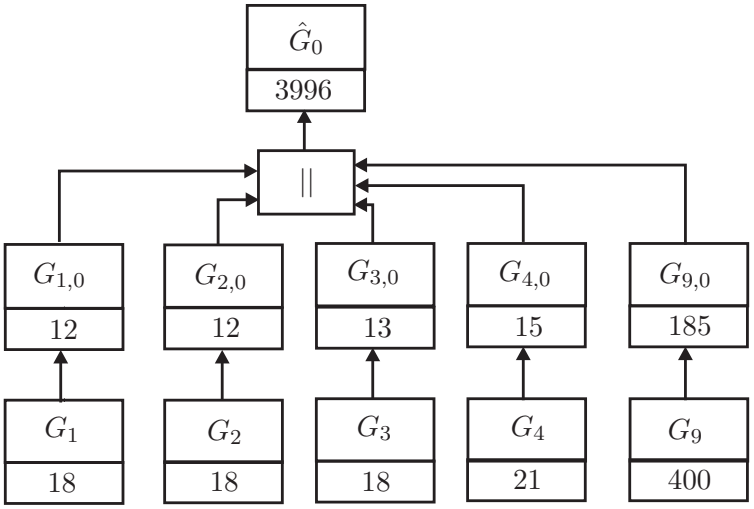

Fig. 6. Hierarchical Structure for the subsystems G_i, $i = 1, \ldots, 4, 9$

Together, 9 supervisors with a maximal state count of 5 were computed. The CMG models involved in the computation are not larger than 3996 states (see Figure 6) compared to 45504 states and 70272 states in [3], [4]. Note that it can be verified that the supervisors in this paper are equivalent to the supervisors in the previous work, and that the supervisor S_0 that had to be deduced from the problem formulation in [4] could be computed systematically in this paper. It is also interesting to note that the high-level model in Figure 6 can be further abstracted on the alphabet $\Sigma_0' = \{$`b4lcy`, `b4lco`, `b4rco`, `b4rcy`, `rb4`, `b3r`, `c1b1`,

c2b2, b1r, b2r, rb3, b7am, b6am, a, b, c3b7$\}$ to a CMG G_0' with 2352 states. This CMG could serve as a model of the FMS that is surrounded by other components in a larger manufacturing system.

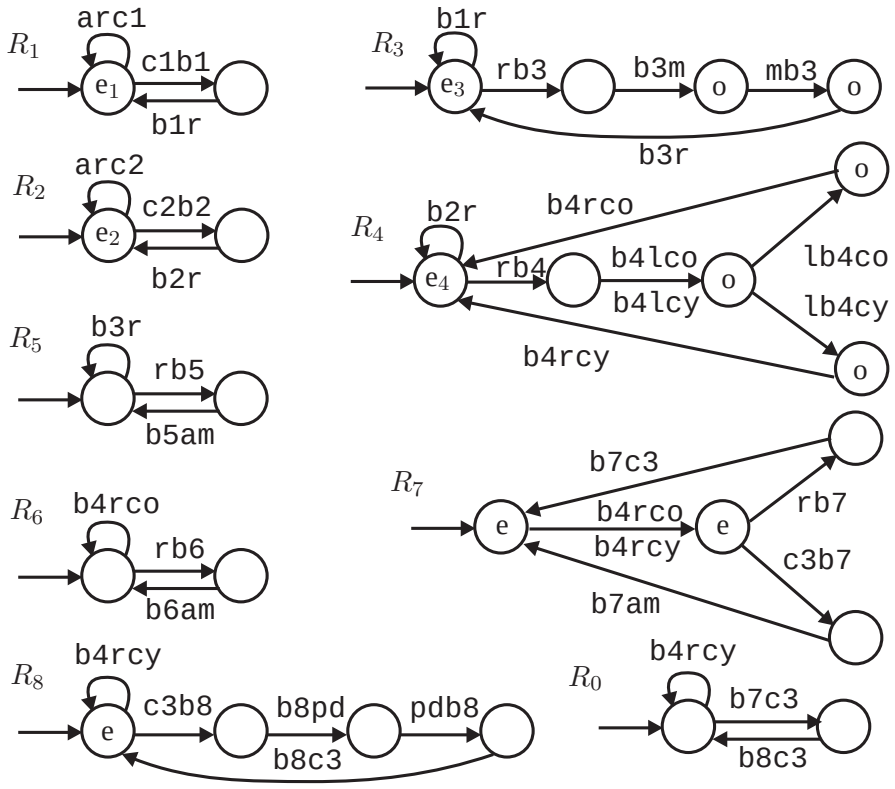

Fig. 7. Reduced Supervisors for the FMS

V. Conclusions

In this paper, a hierarchical control approach for DES has been combined with multitasking supervisory control in order to reduce the computational complexity of supervisor synthesis. A multitasking version of the natural projection as well as the observer property are employed in the abstraction process such that the resulting hierarchical control architecture is hierarchically consistent and strongly nonblocking. The result of the supervisor computation is a set of decentralized supervisors that reside on a small state space, and the efficiency of the approach was illustrated by a flexible manufacturing example. Note that although the supervisor for the example system is equivalent to a monolithic supervisor, maximal permissiveness is not guaranteed by our approach. This issue will be addressed in future work. Furthermore, the use

of more general hierarchical models in terms of the control structure and the actions of the low-level supervisor will be considered, and the application of different methods, e.g. local modular control, for the high-level supervisor synthesis will be investigated.

APPENDIX

The following two lemmas are needed for the proof of the main result of this paper in Theorem 3. They are originally stated in [5] and [8], respectively.

Lemma 4 ([5]): Let $L_1 \subseteq \Sigma_1^*, \ldots, L_n \subseteq \Sigma_n^*$ be languages over the alphabets $\Sigma_1, \ldots, \Sigma_n$. Assume that $\Sigma_0 \subseteq (\Sigma_1 \cup \cdots \cup \Sigma_n)$ and $\bigcup_{i,j,i\neq j}^{n} (\Sigma_i \cap \Sigma_j) \subseteq \Sigma_0$ with the natural projections $p_0 : (\Sigma_1 \cup \cdots \cup \Sigma_n)^* \to \Sigma_0^*$ and $p_i' : \Sigma_i^* \to (\Sigma_i \cap \Sigma_0)^*$, $i = 1, \ldots, n$. Then

$$p_0(L_1 || \cdots || L_n) = p_1'(L_1) || \cdots || p_n'(L_n).$$

Lemma 5 ([8]): Let G be a finite automaton over Σ, and let $\Sigma_0 \subseteq \Sigma$ with the natural projection $p_0 : \Sigma^* \to \Sigma_0^*$. Assume that G_0 is a finite automaton over Σ_0 s.t. $L(G_0) = p_0(L(G))$ and $L_{\mathrm{m}}(G_0) = p_0(L_{\mathrm{m}}(G))$ with a supervisor $S_0 : L(G_0) \to Pwr(\Sigma_0)$. If $S : L(G) \to Pwr(\Sigma)$ is a low-level supervisor implementation of S_0 according to Section II-C and p_0 is an $L_{\mathrm{m}}(G)$-observer, then it holds for all $s \in L(G)$ that

$$\begin{aligned} & p_0(s)t \in L_{\mathrm{m}}(S_0/G_0) \\ & \quad \Rightarrow \exists u \in \Sigma^* \text{ s.t. } su \in L_{\mathrm{m}}(S/G) \wedge p_0(su) = p_0(s)t. \end{aligned}$$

REFERENCES

[1] K. Schmidt, M. H. de Queiroz, and J. E. R. Cury, "Hierarchical and decentralized multitasking control of discrete event systems," *IEEE Conference on Decision and Control*, 2007.

[2] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal of Control and Optimization*, vol. 25, pp. 206–230, 1987.

[3] M. H. de Queiroz, J. E. R. Cury, and W. M. Wonham, "Multitasking supervisory control of discrete-event systems," *Journal on Discrete Event Dynamic Systems: Theory and Applications*, vol. 15, pp. 375–395, 2005.

[4] M. H. de Queiroz and J. E. R. Cury, "Modular multi-tasking supervisory control of composite discrete event systems," in *IFAC World Congress*, 2005.

[5] K. Schmidt, J. Reger, and T. Moor, "Hierarchical control of structural decentralized DES," *Workshop on Discrete Event Systems*, 2004.

[6] R. J. Leduc, M. Lawford, and W. M. Wonham, "Hierarchical interface-based supervisory control-part II: Parallel case," *IEEE Transactions on Automatic Control*, vol. 50, no. 9, pp. 1336–1348, 2005.

[7] K. Schmidt, S. Perk, and T. Moor, "Nonblocking hierarchical control of decentralized systems," *IFAC World Congress, Prague*, 2005.

[8] K. Schmidt, H. Marchand, and B. Gaudin, "Modular and decentralized supervisory control of concurrent discrete event systems using reduced system models," *Workshop on Discrete Event Systems*, 2006.

[9] L. Feng and W. M. Wonham, "Computationally efficient supervisor design: Abstraction and modularity," *Workshop on Discrete Event Systems (WODES), Ann Arbor, USA*, 2006.

[10] R. Hill and D. Tilbury, "Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction," *Workshop on Discrete Event Systems (WODES), Ann Arbor, USA*, 2006.

[11] A. E. C. da Cunha and J. E. R. Cury, "Hierarchical supervisory control based on discrete event systems with flexible marking," *to be pulished in IEEE Transactions on Automatic Control*, 2007.

[12] K. C. Wong and W. M. Wonham, "Hierarchical control of discrete-event systems," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 6, no. 3, pp. 219–314, 1996.

[13] P. N. Pena, J. E. R. Cury, and S. Lafortune, "Testing modularity of local supervisors: An approach based on abstractions," in *Workshop on Discrete Event Systems*, 2006.

[14] K. Wong and W. M. Wonham, "On the computation of observers in discrete-event systems," *Discrete Event Dynamic Systems*, vol. 14, no. 1, pp. 55–107, 2004.

[15] F. Lei and W. M. Wonham, "On the computation of natural observers in discrete-event systems," in *Conference on Decision and Control*, 2006.