Redundant Tasks in Multitasking Control of Discrete Event Systems

Klaus Schmidt * **José E. R. Cury** **

* *Chair of Automatic Control, University of Erlangen-Nuremberg, Germany (e-mail: klaus.schmidt@rt.eei.uni-erlangen.de).*
** *Department of Automation and Systems, Federal University of Santa Catarina, Florianpolis SC 88040-900 Brazil (e-mail: cury@das.ufsc.br)*

Abstract: This paper addresses the control of multitasking DES that allow for dealing with liveness properties in the case where multiple classes of tasks have to be independently completed by the system. Colored marking generators (CMG) have been previously introduced as a model to consider multitasking control. The computational cost of the supervisor synthesis for multitasking DES grows with the number of classes of tasks. In this paper we investigate conditions under which removing tasks of the DES model does not affect the result of supervisory control in the sense that their completion is guaranteed as a consequence of the completion of the other tasks in the DES model. Conditions are derived under which tasks of a class or a set of classes can be removed from the model, and the results are extended to the case of abstracted models in a hierarchical and decentralized control architecture. Those conditions, which can be verified in polynomial time, are stated as properties of strongly connected components of the automata models in different levels of the control hierarchy. The results of the paper are illustrated by a manufacturing system example, showing the potential gains of the approach.

Keywords: Discrete-event systems, hierarchical systems, multitasking supervisory control.

1. INTRODUCTION

The supervisory control theory (SCT) is an expressive framework for the synthesis of controllers for discrete-event systems (DES) (Ramadge and Wonham, 1987). In the SCT, automata models represent the plant and closed-loop desired behaviors, while marked states indicate the completion of system tasks. In this framework a supervisor constrains the behavior of the plant in order to respect the closed-loop specification and to ensure nonblocking, i.e., it always allows the controlled system to reach a marked state. In the SCT, nonblocking can be interpreted as a liveness specification that ensures that the supervisor never prevents the completion of a system task. In (de Queiroz et al., 2005) an approach is introduced to allow for dealing with the case where multiple classes of tasks are identified and (strongly) nonblocking corresponds to the ability of the system to independently complete tasks of all different classes. DES problems comprising multiple classes of tasks often arise in applications, like in manufacturing and communication systems for example (Schmidt et al., 2007; de Queiroz et al., 2005; Fabian and Kumar, 2000; Thistle and Malhame, 1997). Colored marking generators (CMG) are introduced in (de Queiroz et al., 2005) for the synthesis of a minimally restrictive supervisor that respects the specified behavior and ensures the liveness of multiple tasks. Modular control in this framework is addressed in (de Queiroz and Cury, 2005).

In (Schmidt et al., 2007), multitasking control is extended with hierarchical and decentralized control ideas (Hill and Tilbury, 2006; Feng and Wonham, 2008; Schmidt et al., 2008) by combining the computational efficiency of hierarchical abstractions with the ability to specify multiple liveness objectives. To this end, a colored (multitasking) version of both the natural projection and the observer property (Wong and Wonham, 1996) is employed in the hierarchical abstraction process such that the resulting hierarchical control architecture is hierarchically consistent and (strongly) nonblocking.

The computational cost of the supervisor synthesis for multitasking DES grows with the number of classes of system tasks. This is essentially due to the co-accessibility test involved in the synthesis procedure which must be performed with relation to each of the classes of tasks. In some particular cases it may be observed that completion of tasks in a particular class is always guaranteed as a consequence of completion of tasks of other classes. Such *redundant* tasks, that may be introduced either by the modeling process of the DES or as a consequence of the abstraction process in hierarchical control, could be removed from the model to reduce the computational cost. In this paper we derive conditions under which tasks of a class can be identified as redundant tasks, and extend the results to the case where hierarchical and decentralized architectures are to be used. Those conditions, which can be verified in polynomial time, are stated as properties on strongly connected components of the plant models in different levels of the control hierarchy.

The results of the paper are illustrated by a manufacturing system example with effective reductions in the number of classes of tasks. Also, the example shows that the number of states of the abstracted plant models is potentially reduced when removing redundant classes of tasks.

The paper is organized as follows. Section 2 introduces the basic concepts of multitasking supervisory control. Main results on the verification of the stated conditions are presented in Section 3, and extended to hierarchical and decentralized control in Section 4. The detailed example in Section 5 illustrates the approach, and some conclusions are given in Section 6.

2. MULTITASKING DISCRETE EVENT SYSTEMS

2.1 Basic Notation

A color (label) for a multitasking discrete-event system (MTDES) is associated to a class of task. Tasks belong to the same class when they are related to liveness objectives that have the same meaning in the control problem. Let Σ be the set of all system events 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 event sequences in Σ that can complete a task of the respective class. Thus, the *colored behavior* of a MTDES can be modeled by the set $\Lambda_C := \{(L_c, c) | c \in C\} \in Pwr(Pwr(\Sigma^*) \times C)$.

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 $L_B(\Lambda_C) := \bigcup_{b\in B} L_b(\Lambda_C)$. The *synchronous composition* of $M_{B_1} \in Pwr(Pwr(\Sigma_1^*) \times B_1)$ and $N_{B_2} \in Pwr(Pwr(\Sigma_2^*) \times B_2)$ 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 define the classes of tasks that are completed after the corresponding strings. Formally, we define the *colored marking generator* (CMG) $G = (Q, \Sigma, C, \delta, \chi, q_0)$, 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* of G is $L(G) := \{s \in \Sigma^* | \delta(q_0, s) \text{ is defined}\}$, and the language *marked* by $c \in C$, is given by $L_c(G) := \{s \in L(G) | c \in \chi(\delta(q_0, s))\}$. 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 (de Queiroz et al., 2005). 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 $\forall b \in B$, $\overline{L_b(\Lambda_C)} = \overline{L_C(\Lambda_C)}$.

2.2 Multitasking Supervisory Control

Let a MTDES be modeled by a colored marking generator $G = (Q, \Sigma, C, \delta, \chi, q_0)$ whose alphabet is partitioned into controllable events Σ_c and uncontrollable events Σ_u. We assume w.l.o.g. that a *colored specification* $A_D \subseteq Pwr(\Sigma^*) \times D$ is constructed from a safety specification $K = \overline{K} \subseteq L(G)$ and liveness conditions defined by the set of classes of tasks C and a set of new classes E s.t. $E \cap C = \emptyset$ and $D = C \dot{\cup} E$ as follows.

$$A_D = \{(L_c, c)| \ c \in D \text{ s.t. } L_c = K \cap L_c(G) \text{ for } c \in C \text{ and } L_c \subseteq K \text{ for } c \in E\}. \quad (1)$$

A *coloring supervisor* $S : L(G) \to Pwr(\Sigma) \times Pwr(E)$ associates to each string of the plant a set of enabled events and a set of colors (of E) marking the string as a completed task of the classes represented by these colors. 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 $\forall s \in L(G)$, $\Sigma_u \cap \Gamma(\delta(q_0, s)) \subseteq \mathcal{R}(S(s))$, and strongly nonblocking w.r.t D if $\forall d \in D$, $\overline{L_d(S/G)} = L(S/G)$.

Theorem 1. ((de Queiroz et al., 2005)). 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: $\overline{L_D(A_D)}\Sigma_u \cap L(G) \subseteq \overline{L_D(A_D)}$;
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 (de Queiroz et al., 2005), it is also proved that the supremal controllable and strongly nonblocking colored behavior contained in A_D, named SupCSNB(A_D, G, D), exists and can be computed with complexity polynomial in the number of states of the model.

3. REMOVING REDUNDANT COLORS

The algorithmic computation of SupCSNB(A_D, G, D) as defined above relies on an iterative computation of nonblocking subbehaviors of $\Lambda_C(G)||A_D$ for all colors $d \in D$. Hence, each additional color contributes to the computational cost of the supervisor synthesis. The goal of this section is to identify and remove colors that are not relevant for the supervisor synthesis in order to reduce this computational cost. The idea is first illustrated by an example in Section 3.1, and then formalized in Section 3.2.

3.1 Motivation and Problem Formulation

We consider two neighboring components of the production cell in Fig. 1; the conveyor belt C1 and the machine M. The task of C1 is to transport parts to the machine M, which processes each part before it can depart. C1 is modeled by the CMG $G_{\mathrm{C1}}^{(1)}$ in Fig. 2, where C1 stops if `c1stp` occurs, and the events `c1-0`, `c0-1` and `c1-2`, `c2-1` describe the exchange of parts with the neighboring conveyor belts C0 and C2, respectively. The machine (CMG $G_{\mathrm{M}}^{(1)}$) can start processing (`ms`) and finishes processing with the uncontrollable event `mf`. In addition, one color is introduced for each component. It is desired that C1 can always become empty (C1e) and that the machine cannot be prevented from processing (Mp). Note that transitions with controllable events are labeled with a tick and that the set of colors is displayed next to the respective state in all plant models. It is specified in $M^{(1)}_{\mathrm{C1-M}}$ that every part entering C1 has to stop at M and can only leave C1 after processing is finished. Fig. 2 displays the CMG $R^{(1)}_{\mathrm{C1-M}}$ that represents the closed-loop behavior $SupCSNB(G, A_D, D)$ with the plant $G := G^{(1)}_{\mathrm{C1}}||G^{(1)}_{\mathrm{M}}$, the specification behavior $A_D := L(M^{(1)}_{\mathrm{C1-M}})||\Lambda_C(G)$ according to (1) and the color set $D = C = \{\mathrm{C1e}, \mathrm{Mp}\}$.

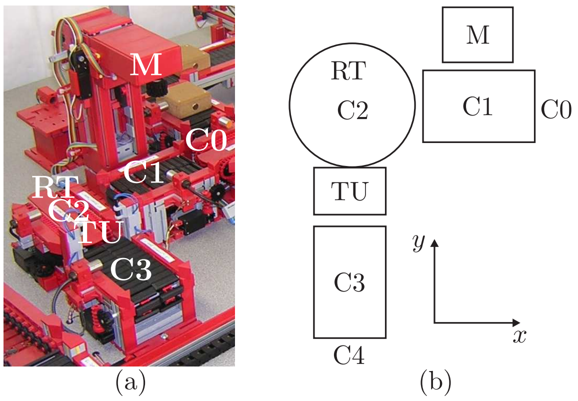


Fig. 1. Production cell: (a) Picture; (b) Overview.

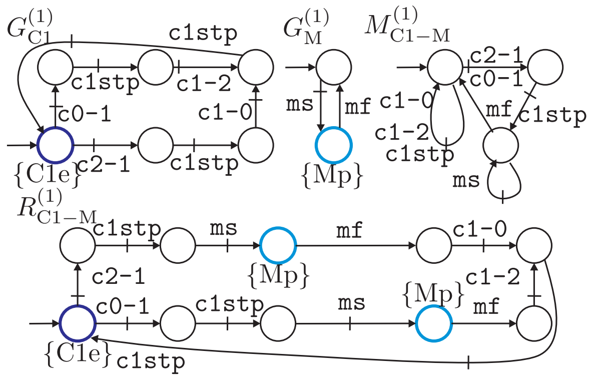


Fig. 2. Conveyor belt C1 and machine M.

A closer inspection of $R^{(1)}_{\mathrm{C1-M}}$ reveals that, although the colors C1e and Mp were introduced independently in their respective component models, there is a direct dependency after the supervisor synthesis. Suppose an additional SNB supervisor $\tilde{S}$ shall be designed for a new specification $\tilde{A}_D$ and the plant $R^{(1)}_{\mathrm{C1-M}}$ with the color set $D = \{\mathrm{C1e}, \mathrm{Mp}\}$. It is now sufficient to synthesize a nonblocking supervisor w.r.t. the color set $\tilde{D} = \{\mathrm{Mp}\}$ since this already implies SNB w.r.t. D due to the plant structure. In particular, $\tilde{S}$ makes sure that a state with color Mp is always reachable in $\tilde{S}/R^{(1)}_{\mathrm{C1-M}}$. Observing that $\tilde{S}$ can never disable the uncontrollable event `mf`, this implies that it is also always possible that an unmarked state is reached in $\tilde{S}/R^{(1)}_{\mathrm{C1-M}}$. Then, it holds that on each path back to a state with color Mp, a state with the color C1e is passed. It can hence be concluded that the color C1e is not relevant for any further supervisor synthesis and can be removed from $R^{(1)}_{\mathrm{C1-M}}$.

3.2 Condition for Color Removal

Based on the motivating example, the goal of this section is to identify colors that are not relevant for the supervisor synthesis. In order to reduce the computational effort for the supervisor computation, we then propose to remove such colors. Formally, we want to solve Problem 1.

Problem 1. Let $G = (Q, \Sigma, C, \delta, \chi, q_0)$ be a CMG, let Σ_{u} be a set of uncontrollable events and assume that $c \in C$ is a color. We want to determine verifiable conditions such that for all specifications A_D according to (1)

$$\Lambda_D(\tilde{S}/G) = \Lambda_D(SupCSNB(G, A_D, D)),$$

where $\tilde{S} : L(G) \to 2^{\Sigma} \times 2^{D-C}$ is a coloring supervisors for the reduced specification $A_{\tilde{D}} = \{(L_d(A_d), d)|d \in \tilde{D}\}$ over the color set $\tilde{D} = D - \{c\}$, i.e.,

$$\Lambda_{\tilde{D}}(\tilde{S}/G) = SupCSNB(G, A_{\tilde{D}}, \tilde{D}).$$ □

We first adapt the definition of a *strongly connected component* (SCC) in (Hopcroft and Ullman, 1975) to CMGs.

Definition 1. (SCC). Let $G = (Q, \Sigma, C, \delta, \chi, q_0)$ be a CMG. A subgraph of G with the states $\mathcal{G} \subseteq Q$ is called a strongly connected component (SCC) of G if for all state pairs $q, q' \in \mathcal{G}$, there is $u, u' \in \Sigma^*$ s.t. $\delta(q, u) = q'$ and $\delta(q', u') = q$ and for all $\mathcal{G}' \supset \mathcal{G}$, $\mathcal{G}'$ is not a SCC of G. □

Theorem 2 states sufficient conditions to solve Problem 1 by exploiting structural information about the plant G (a proof can be found in (Schmidt and Cury, 2009)).

Theorem 2. (Main Theorem). Write $\tilde{C} = C - \{c\}$. Problem 1 is solved if the following condition is satisfied. There is no SCC with the states $\mathcal{G} \subseteq Q$ in G s.t.

(i) $\bigcup_{q\in\mathcal{G}} \chi(q) = \tilde{C}$
(ii) $\nexists \sigma \in \Sigma_{\mathrm{u}}$, $q \in \mathcal{G}$ s.t. $\delta(q, \sigma) \notin \mathcal{G}$.

That is, c can be removed from the color set C of G and the specification $A_{\tilde{D}}$ can be used instead of A_D if (i) and (ii) hold. Note that Theorem 2 also applies to the motivating example in Section 3.1. Both SCCs of $R^{(1)}_{\mathrm{C1-M}}$ with the color Mp (states labeled with Mp) have an uncontrollable transition with `mf` leaving the respective SCC.

3.3 Algorithmic Verification

The following algorithm allows to check the condition in Theorem 2 by finding a violating SCC if such SCC exists.

Algorithm 1. (Check Removal of Color c).
Given: CMG $G = (Q, \Sigma, C, \delta, \chi, q_0)$, color c.

```
1. $\tilde{G} = (\tilde{Q}, \tilde{\Sigma}, \tilde{C}, \tilde{\delta}, \tilde{\chi}, \tilde{q}_0) = G$
2. delete all states with $c$ from $\tilde{G}$; remove $c$ from $\tilde{C}$:
     $\forall q \in \tilde{Q} : c \in \tilde{\chi}(q) \Rightarrow \tilde{Q} := \tilde{Q} - \{q\}$;   $\tilde{C} := \tilde{C} - \{c\}$
3. find all SCCs in $\tilde{G}$ that contain states with all colors
   in $\tilde{C}$. Denote these SCCs as $\mathcal{G}_1, \ldots, \mathcal{G}_m$.
4. remove all states that are not in $\bigcup_{i=1}^{m} \mathcal{G}_i$ from $\tilde{G}$:
     $\forall q \in \tilde{Q} : q \notin \bigcup_{i=1}^{m} \mathcal{G}_i \Rightarrow \tilde{Q} := \tilde{Q} - \{q\}$
5. delete all states in $\mathcal{G}_i$, $i = 1, \ldots, m$ that have uncon-
   trollable transitions in the original automaton $G$ that
   lead outside $\mathcal{G}_i$, i.e., $\forall i \in \{1, \ldots, m\}$:
     $\forall q \in \mathcal{G}_i, \forall \sigma \in \Sigma_{\mathrm{u}} : \delta(q, \sigma) \notin \mathcal{G}_i \Rightarrow \tilde{Q} := \tilde{Q} - \{q\}$
```

```
6. **if** states were deleted in step 5. **and** $\tilde{Q}$ is not empty
       **go to** step 3.
7. **if** $\tilde{Q}$ is empty
       **return true**
   **else**
       **return false**  □
```

The algorithm iteratively removes states from the plant CMG G if they violate (i) (3. and 4.) or if they violate (ii) (5.) in Theorem 2. The algorithm terminates in at most $|Q|$ steps, where $|Q|$ is the state count G. Furthermore, the computation of the SCCs in 3. can be performed by Tarjan's algorithm in (Hopcroft and Ullman, 1975) with a complexity of $O(\max\{|Q|, |\delta|\})$, where $|\delta|$ denotes the number of transitions of G. Together, Algorithm 1 exhibits a computational complexity of $O(|Q| \cdot \max\{|Q|, |\delta|\})$.

We apply Algorithm 1 to $G = R^{(1)}_{\mathrm{C1-M}}$ in Fig. 2 and $c = \mathrm{C1e}$. In 2., the initial state is removed. Two SCCs that consist of the states with the color MP remain after 3. and 4. Since the uncontrollable event `mf` leads outside both SCCs, $\tilde{Q}$ is empty after 5. Hence, the algorithm returns **true**, which is consistent with the previous discussion.

It is readily observed that an iterative application of the above procedure enables the removal of an arbitrary number of colors as long as Algorithm 1 returns **true**.

4. MULTITASKING HIERARCHICAL AND DECENTRALIZED CONTROL

In the previous section, it is pointed out that the removal of redundant colors leads to computational savings in the supervisor synthesis for MTDES. In this section, we combine the idea of removing colors with hierarchical and decentralized control for MTDES (Schmidt et al., 2007).

4.1 Control Approach

It is assumed that the original (low-level) plant is given as a set $G_i = (Q_i, \Sigma_i, C_i, \delta_i, \chi_i, q_{0,i})$, $i = 1, \ldots, n$ of CMGs, and the overall plant is $G = ||_{i=1}^n G_i$ with the color set $C := \bigcup_{i=1}^n C_i$. The hierarchical abstraction of G is based on the *colored natural projection.*

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

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

The *high-level plant* G_0 is then computed using abstractions of the plant components G_i, $i = 1, \ldots n$ on a superset of their *shared events* $\Sigma_{i,\cap} := \bigcup_{k=1,k\neq i}^n (\Sigma_i \cap \Sigma_k)$.

Definition 3. (High-level Plant). Let G and G_i, $i = 1, \ldots,$ n, p_0 and m_0 be defined as above. Assume that $\Sigma_{i,0} \subseteq \Sigma_i$ are given such that $\Sigma_{i,\cap} \subseteq \Sigma_{i,0}$ and introduce the natural projections $p_{\Sigma_i \to \Sigma_{i,0}}$ and the colored natural projections $m_{\Sigma_i \to \Sigma_{i,0}}$. Then, with $L(G_{i,0}) := p_{\Sigma_i \to \Sigma_{i,0}}(L(G_i))$ and $\Lambda_C(G_{i,0}) := m_{\Sigma_i \to \Sigma_{i,0}}(\Lambda_C(G_{i,0}))$, the high-level plant G_0 is defined by

$$G_0 = ||_{i=1}^n G_{i,0}, \qquad \square$$

The abstraction process is illustrated on the right-hand side of Fig. 3. Given a coloring behavior $A_{D,0} \in Pwr(Pwr(\Sigma_0^*) \times D)$ as a high-level specification, the coloring *high-level supervisor* $S_0 : L(G_0) \to Pwr(\Sigma_0) \times Pwr(E)$ with $E = D - C$ is computed such that S_0 realizes $SupCSNB(A_{D,0}, G_0, D)$. The control action of the corresponding *low-level supervisor* $S : L(G) \to Pwr(\Sigma) \times Pwr(E)$ is then defined for each $s \in L(G)$ as

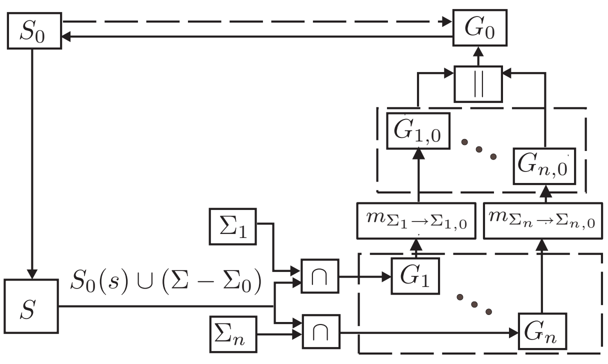
$$S(s) := \big(S_0(p_0(s)) \cup (\Sigma - \Sigma_0), \mathcal{I}(S_0(p_0(s)))\big), \qquad (2)$$

such that $L(S/G) = L(S_0/G_0)||L(G)$ and $\Lambda_C(S/G) = \Lambda_C(S_0/G_0)||\Lambda_C(G)$.

Hence, the control action after a string $s \in L(G)$ is

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

as depicted on the left-hand side of Fig. 3.



S_0
G_0
$||$
$G_{1,0}$
$G_{n,0}$
$m_{\Sigma_1 \to \Sigma_{1,0}}$
$m_{\Sigma_n \to \Sigma_{n,0}}$
Σ_1
$S_0(s) \cup (\Sigma - \Sigma_0)$
$\cap$
G_1
S
Σ_n
$\cap$
G_n

Fig. 3. Hierarchical and decentralized control architecture

In order to guarantee that the low-level closed loop S/G is SNB, we employ the *colored observer* condition.

Definition 4. (Colored Observer (Schmidt et al., 2007)). 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 $\Sigma_0 \subseteq \Sigma$ and p_0, m_0 be defined as above. 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), i.e, for each $s \in \overline{L}$, $t \in \Sigma_0^*$, and $c \in C$

$$p_0(s)t \in L_c(m_0(\Lambda_C)) \Rightarrow \exists u \in \Sigma^* \text{ s.t. } su \in L_c(\Lambda_C) \text{ and } p_0(su) = p_0(s)t$$

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. (Schmidt et al. (2007)). 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 (2). 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 SNB, i.e., for all $c \in C$

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

4.2 Removal of Redundant Colors

We now combine hierarchical control in the framework presented in the previous section with the idea of removing redundant colors. To this end, we first recall the *mutual controllability* from (Lee and Wong, 2002).

Definition 5. (Mutual Controllability). The CMGs G_i and G_j are denoted mutually controllable if

$$L(G_i)(\Sigma_{j,\mathrm{u}} \cap \Sigma_i) \cap p_{\Sigma_i \cup \Sigma_j \to \Sigma_i}(p^{-1}_{\Sigma_i \cup \Sigma_j \to \Sigma_j}(L(G_j)) \subseteq L(G_i)$$

$$L(G_j)(\Sigma_{i,\mathrm{u}} \cap \Sigma_j) \cap p_{\Sigma_j \cup \Sigma_i \to \Sigma_j}(p^{-1}_{\Sigma_j \cup \Sigma_i \to \Sigma_i}(L(G_i)) \subseteq L(G_j)$$

Mutual controllability ensures that after any execution of a composed system, the occurrence of a shared uncontrollable event is either feasible in every subsystem which shares it, or it is not feasible in any subsystem.

The following theorem is proved in (Schmidt and Cury, 2009). It relates the redundancy of a color c to the redundancy of the color in the components where c appears.

Theorem 4. Let $G = ||_{i=1}^{n}$ be a plant with the components $G_i, i = 1, \dots, n$, and let G_0 be the high-level plant. Assume that G_i, G_j are mutually controllable for all $i \neq j$. Also define the set $C_{i,\cap} := \bigcup_{l=1, l\neq i}^{n} (C_i \cap C_l)$ of shared colors among components, and assume that $c \in C_k - C_{k,\cap}$ for some $k \in \{1, \dots, n\}$, $\tilde{C} := C - \{c\}$, and S_0 is a supervisor such that S_0/G_0 is SNB for $\tilde{C}$. If G_k satisfies Theorem 2 for $C_k - \{c\}$, then S evaluated with (2) is SNB w.r.t. C.

We now propose the following procedure for the combination of hierarchical abstraction and color removal.

1. Remove all redundant colors $c \notin C_{i,\cap}$ from each G_i and denote the remaining colors by $\tilde{C}_i \subseteq C_i$
2. Determine colored observers $m_{\Sigma_i \to \Sigma_{i,0}}$, $i = 1, \dots, n$
3. Synthesize the supervisor S according to (2) for a given high-level specification $A_{D,0}$.

5. APPLICATION EXAMPLE

In this section, we apply hierarchical multitasking control to the production cell (PC) in Fig. 1, and illustrate the benefits of removing redundant colors. All computations are performed using the "multitasking" plugin of the `libFAUDES` software library for DES (libFAUDES, 2008).

5.1 General Setup

In addition to the components C1 and M described in Section 3.1, the PC consists of the conveyor belts C2 and C3, the rotary table RT, and a test unit TU. CMG models for all components have been determined based on physical plant events (sensors and actuators). However, the description in this paper starts with plant models on the hierarchical level (1) in order to provide a compact representation. The state counts of the closed-loop CMGs $R_i^{(0)}$, $i \in \mathcal{C} := \{\mathrm{C1, C2, C3, M, RT, TU}\}$ on the lowest level (0) are displayed in Fig. 5 (next to the respective CMG).

5.2 Models on Level 1

We model the plant components using the same convention for event names as in Section 3.1 (see also Fig. 4).

Conveyor belt C2 ($G_{\mathrm{C2}}^{(1)}$): C2 allows to transport parts from C1 to C3 or from C3 to C2 and back to C3 or to C1. The color C2e requires C2 to always become empty again.

Rotary table RT ($G_{\mathrm{RT}}^{(1)}$): RT initially points in the x-direction. It can turn to the y-direction (`RTy`) and back to the x-direction (`RTx`). RT must always be able to stop (`RTstp`) in one of its two positions (color RTs).

Conveyor belt C3 ($G_{\mathrm{C3}}^{(1)}$): C3 accepts parts from C2 and C4, and then delivers them either to C2 or C4. The color C3e indicates that C3 should always become empty again.

Test unit TU ($G_{\mathrm{TU}}^{(1)}$): TU is located between C2 and C3. It checks parts that travel from C2 to C3 or vice versa, and decides if they are acceptable (`acc`) or have to be rejected (`rej`). TU keeps track of parts until they leave towards C1 (`c2-1`) or C4 (`c3-4`). By coloring, we ensure that parts can always be either accepted (A) or rejected (R).

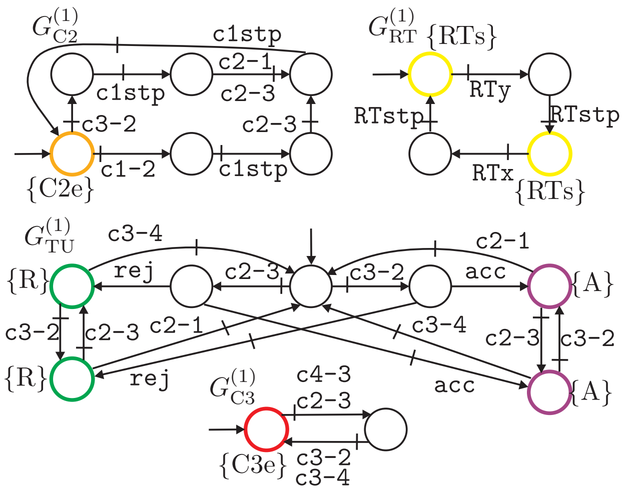
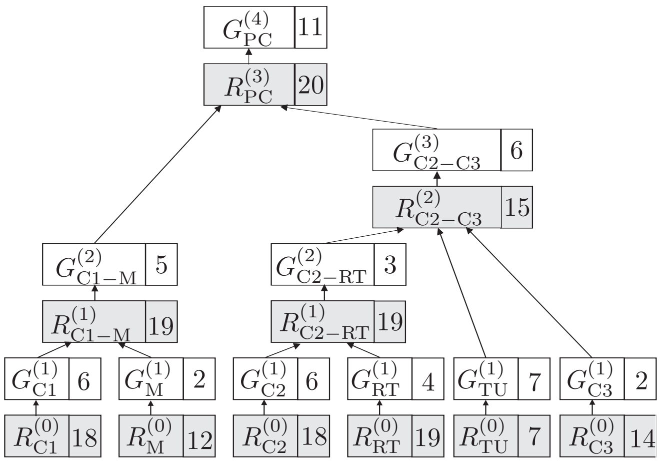


Fig. 4. Level 1 models of the production cell.

5.3 Hierarchical Supervisor Synthesis

We now perform hierarchical supervisor synthesis according to Section 4.2. The hierarchical architecture is presented in Fig. 5, where gray and white boxes denote closed-loop CMGs and abstracted plant models, respectively.



$G_{\mathrm{PC}}^{(4)}$ 11
$R_{\mathrm{PC}}^{(3)}$ 20
$G_{\mathrm{C2-C3}}^{(3)}$ 6
$R_{\mathrm{C2-C3}}^{(2)}$ 15
$G_{\mathrm{C1-M}}^{(2)}$ 5
$G_{\mathrm{C2-RT}}^{(2)}$ 3
$R_{\mathrm{C1-M}}^{(1)}$ 19
$R_{\mathrm{C2-RT}}^{(1)}$ 19
$G_{\mathrm{C1}}^{(1)}$ 6
$G_{\mathrm{M}}^{(1)}$ 2
$G_{\mathrm{C2}}^{(1)}$ 6
$G_{\mathrm{RT}}^{(1)}$ 4
$G_{\mathrm{TU}}^{(1)}$ 7
$G_{\mathrm{C3}}^{(1)}$ 2
$R_{\mathrm{C1}}^{(0)}$ 18
$R_{\mathrm{M}}^{(0)}$ 12
$R_{\mathrm{C2}}^{(0)}$ 18
$R_{\mathrm{RT}}^{(0)}$ 19
$R_{\mathrm{TU}}^{(0)}$ 7
$R_{\mathrm{C3}}^{(0)}$ 14

Fig. 5. Hierarchical architecture for the production cell

C2 and RT: A supervisor is designed for the conveyor belt C2 that is mounted on RT. The specifications $M_{\mathrm{C2-RT},1}^{(1)}$ and $M_{\mathrm{C2-RT},2}^{(1)}$ in Fig. 6 require that only one of the components is allowed to move, and RT has to turn according to delivery performed by C1, respectively. The resulting supervisor $R_{\mathrm{C2-RT}}^{(1)}$ has 19 states. Its abstraction $G_{\mathrm{C2-RT}}^{(2)}$ is shown in Fig. 7.

C2, RT, C3 and TU: The specification $M_{\mathrm{C2-C3}}^{(2)}$ addresses the combined behavior of C2, TU and C3 on level 2 of the hierarchy. It states that accepted parts have to leave PC via C4, while rejected parts have to pass M and leave towards C0. The supervisor $R^{(2)}_{\mathrm{C2-C3}}$ has 15 states and contains the redundant colors RTs and C2e. The abstraction $G^{(3)}_{\mathrm{C2-C3}}$ is shown in Fig. 7.

Production Cell: Finally, the overall PC $R^{(3)}_{\mathrm{PC}}$ is synthesized on level 3 of the hierarchy without an additional safety specification. Again, one color (Mp) can be removed such that the abstraction $G^{(4)}_{\mathrm{PC}}$ in Fig. 7 only contains two of the originally seven colors. It can for example be used as a model of the PC in a larger manufacturing system. The overall closed-loop system represented by

$$S/G = (\|_{i\in\mathcal{C}} R_i^{(0)})||R^{(1)}_{\mathrm{C1-M}}||R^{(1)}_{\mathrm{C2-RT}}||R^{(2)}_{\mathrm{C2-C3}}||R^{(3)}_{\mathrm{PC}} \quad (3)$$

is SNB and fulfills the given specifications.

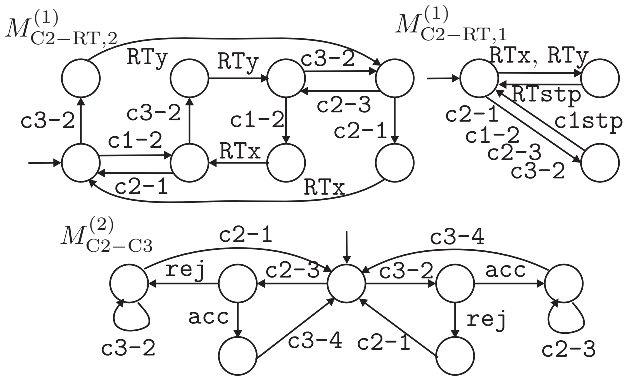


Fig. 6. Safety specifications for the production cell.

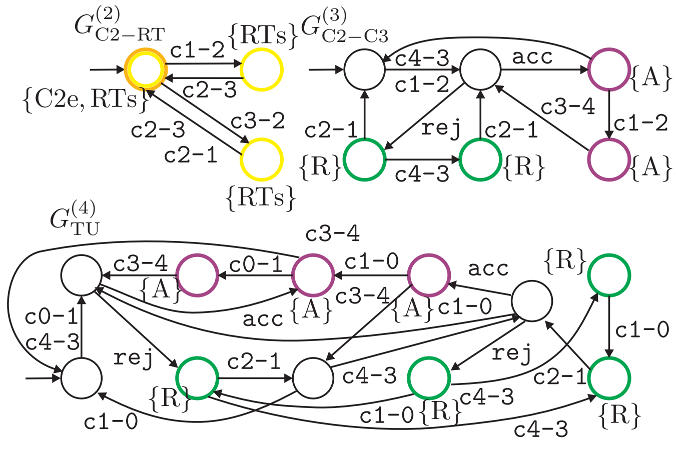


Fig. 7. Hierarchical supervisors for the production cell.

5.4 Performance Comparison

In comparison, a completely monolithic supervisor synthesis results in an overall plant G with 1 133 484 states, a composed specification with 9 298 states, and a monolithic supervisor with 17 355 states. In contrast, the hierarchical synthesis in Section 5.3 comprises a sum of 161 states, since S/G in (3) need not be composed.

If hierarchical control without removing colors is used, not only computations for all 7 colors have to be carried out, but also the resulting high-level models $G^{(3)}_{\mathrm{C2-C3}}$ (11 states) and $G^{(4)}_{\mathrm{PC}}$ (31 states) are larger. This is due to the fact that the colored observer condition has to be fulfilled for all colors.

6. CONCLUSIONS

The results in this paper show how identifying and removing redundant tasks in multitasking control of DES may lead to considerable savings in the computational effort of synthesizing supervisors for this class of systems. The illustration of the established conditions in the example of a manufacturing cell puts in evidence the gains we can have in hierarchical and decentralized control architectures, not only by the removal of colors in the CMG models of different levels in the system hierarchy, but also by the reduction in the size of the abstracted models as a consequence of eliminating tasks. Further research currently being carried out on this subject includes applying the results in a larger example and deriving algorithmic computations of maximal sets of redundant classes of tasks.

REFERENCES

de Queiroz, M.H. and Cury, J.E.R. (2005). Modular multitasking supervisory control of composite discrete event systems. In *IFAC World Congress*.

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

Fabian, M. and Kumar, R. (2000). Mutually nonblocking supervisory control of des. *Automatica*, 36, 1863–1869.

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

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

Hopcroft, J.E. and Ullman, J.D. (1975). *The design and analysis of computer algorithms*. Addison-Wesley.

Lee, S.H. and Wong, K. (2002). Structural decentralised control of concurrent DES. *European Journal of Control*, 35, 1125–1134.

libFAUDES (2008). Friedrich-Alexander University Discrete Event Systems library. URL `http://www.rt.eei.uni-erlangen.de/FGdes/faudes/index.php`.

Ramadge, P.J. and Wonham, W.M. (1987). Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25, 206–230.

Schmidt, K. and Cury, J. (2009). Redundant tasks in multitasking control of discrete event systems. *Technical Report, University of Erlangen-Nuremberg and Federal University of Santa Catarina*.

Schmidt, K., de Queiroz, M., and Cury, J. (2007). Hierarchical and decentralized multitasking control of discrete event systems. *Decision and Control, 2007 46th IEEE Conference on*, 5936–5941.

Schmidt, K., Moor, T., and Perk, S. (2008). Nonblocking hierarchical control of decentralized discrete event systems. *Automatic Control, IEEE Transactions on*, 53(10), 2252–2265.

Thistle, J.G.R. and Malhame, P. (1997). Multiple marked languages and noninterference in modular supervisory control. In *35th Annual Allerton Conference*.

Wong, K.C. and Wonham, W.M. (1996). Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 6(3), 219–314.