NONBLOCKING HIERARCHICAL CONTROL OF DECENTRALIZED DES

Klaus Schmidt, Sebastian Perk and Thomas Moor

Lehrstuhl für Regelungstechnik
Universität Erlangen-Nürnberg
Cauerstraße 7, D–91058 Erlangen, Germany
klaus.schmidt@rt.eei.uni-erlangen.de

Abstract: This work considers a hierarchical control architecture for a class of discrete event systems which can also be applied to decentralized control systems. It is shown that nonblocking supervisory control on the high level of the hierarchy results in nonblocking and hierarchically consistent control on the low level.

Keywords: supervisory control, discrete event systems, decentralized systems, hierarchical abstraction.

1. INTRODUCTION

There has been considerable effort in studying methods to reduce the complexity of synthesis algorithms for the supervisory control of discrete event systems Yoo and Lafortune (2000); Hubbard and Caines (2002); da Cunha et al. (2002); Zhong and Wonham (1990); Lee and Wong (2002); Wong and Wonham (1996); Leduc et al. (2001); Komenda and van Schuppen (2003). Generally, promising approaches assume or impose a particular control architecture, such that computationally expensive product compositions of individual subsystems can be either avoided or at least postponed to a more favorable stage in the design process. Our contribution builds on hierarchical control and introduces two structural properties, marked state acceptance and local nonblocking.

In hierarchical architectures Zhong and Wonham (1990); da Cunha et al. (2002); Hubbard and Caines (2002); Schmidt et al. (2004), controller synthesis is based on a plant abstraction (high-level model), which is supposed to be less complex than the original plant model (low-level model). Technically, abstractions can be defined as language projections. While projections are known to be of exponential computational complexity in the worst case, Wong (1997) identifies application relevant cases with polynomial complexity. An important question is how to derive the plant abstraction, such that a high-level controller can be implemented by available low-level control actions (hierarchical consistency). A characterization of this property is given in Zhong and Wonham (1990).

This paper builds on previous results by Schmidt et al. (2004), using the natural language projection on high-level events as an abstraction. We then recall that this abstraction complies with hierarchical consistency as defined in Zhong and Wonham (1990) and it is shown that nonblocking high-level supervisors yield nonblocking low-level supervisors. Furthermore, this architecture is applied to decentralized DES with the shared events of the subsystems as high-level events. Whenever the projections of the subsystems behave computationally nicely, this change of order promises a substantial computational benefit. This is demonstrated by an example.

The outline of the paper is as follows. Basic notations and definitions of supervisory control theory are recalled in Section 2. Section 3 introduces the notion of marked state acceptance and local nonblocking combined with hierarchical control and proves nonblocking control for the architecture. In Section 4, the architecture is extended to form a decentralized and

hierarchical control architecture. A comprehensive example in Section 5 illustrates our contribution.

2. PRELIMINARIES

We recall basic facts from supervisory control theory. Wonham (2004); Cassandras and Lafortune (1999).

For a finite alphabet Σ, the set of all finite strings over Σ is denoted Σ^*. We write $s_1 s_2 \in \Sigma^*$ for the concatenation of two strings $s_1, s_2 \in \Sigma^*$. We write $s_1 \leq s$ when s_1 is a *prefix* of s, i.e. if there exists a string $s_2 \in \Sigma^*$ with $s = s_1 s_2$. The empty string is denoted $\varepsilon \in \Sigma^*$, i.e. $s\varepsilon = \varepsilon s = s$ for all $s \in \Sigma^*$. A *language* over Σ is a subset $H \subseteq \Sigma^*$. The *prefix closure* of H is defined by $\overline{H} := \{s_1 \in \Sigma^* | \exists s \in H \text{ s.t. } s_1 \leq s\}$. A language H is *prefix closed* if $H = \overline{H}$.

The *natural projection* $p_i : \Sigma^* \to \Sigma_i^*$, $i = 1,2$, for the (not necessarily disjoint) union $\Sigma = \Sigma_1 \cup \Sigma_2$ is defined iteratively: (1) let $p_i(\varepsilon) := \varepsilon$; (2) for $s \in \Sigma^*$, $\sigma \in \Sigma$, let $p_i(s\sigma) := p_i(s)\sigma$ if $\sigma \in \Sigma_i$, or $p_i(s\sigma) := p_i(s)$ otherwise. The set-valued inverse of p_i is denoted $p_i^{-1} : \Sigma_i^* \to 2^{\Sigma^*}$, $p_i^{-1}(t) := \{s \in \Sigma^* | p_i(s) = t\}$. The *synchronous product* $H_1 || H_2 \subseteq \Sigma^*$ of two languages $H_i \subseteq \Sigma_i^*$ is $H_1 || H_2 = p_1^{-1}(H_1) \cap p_2^{-1}(H_2) \subseteq \Sigma^*$.

A *finite automaton* is a tuple $G = (X, \Sigma, \delta, x_0, X_m)$, 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* $X_m \subseteq X$. We write $\delta(x,\sigma)!$ if δ is defined at (x,σ). In order to extend δ to a partial function on $X \times \Sigma^*$, recursively let $\delta(x,\varepsilon) := x$ and $\delta(x, s\sigma) := \delta(\delta(x,s),\sigma)$, whenever both $x' = \delta(x,s)$ and $\delta(x',\sigma)!$. $L(G) := \{s \in \Sigma^* : \delta(x_0,s)!\}$ and $L_m(G) := \{s \in L(G) : \delta(x_0,s) \in X_m\}$ are the *closed* and *marked language* generated by the finite automaton G, respectively. For a formal definition of the synchronous composition of two automata G_1 and G_2 we refer to e.g. Cassandras and Lafortune (1999) and note that $L(G_1 || G_2) = L(G_1) || L(G_2)$.

In a supervisory control context, we write $\Sigma = \Sigma_c \cup \Sigma_u$, $\Sigma_c \cap \Sigma_u = \emptyset$, to distinguish *controllable* (Σ_c) and *uncontrollable* (Σ_u) events. A *control pattern* is a set γ, $\Sigma_u \subseteq \gamma \subseteq \Sigma$, and the set of all control patterns is denoted $\Gamma \subseteq 2^{\Sigma}$. A *supervisor* is a map $S\colon L(G) \to \Gamma$, where $S(s)$ represents the set of enabled events after the occurrence of string s; i.e. a supervisor can disable controllable events only. The language $L(S/G)$ generated by G under supervision S is iteratively defined by (1) $\varepsilon \in L(S/G)$ and (2) $s\sigma \in L(S/G)$ iff $s \in L(S/G), \sigma \in S(s)$ and $s\sigma \in L(G)$. Thus, $L(S/G)$ represents the behavior of the *closed-loop system*. To take into account the marking of G, let $L_m(S/G) := L(S/G) \cap L_m(G)$. The closed-loop system is *nonblocking* if $\overline{L_m(S/G)} = L(S/G)$, i.e. if each string in $L(S/G)$ is the prefix of a marked string in $L_m(S/G)$.

A language H is said to be controllable w.r.t. $L(G)$ if there exists a supervisor S such that $\overline{H} = L(S/G)$. The set of all languages that are controllable w.r.t. $L(G)$ is denoted $\mathcal{C}(L(G))$ and can be characterized by $\mathcal{C}(L(G)) = \{H \subseteq L(G) | \exists S \text{ s.t. } \overline{H} = L(S/G)\}$. Furthermore, the set $\mathcal{C}(L(G))$ is closed under arbitrary union. Hence, for every *specification* language E there uniquely exists a *supremal controllable sublanguage* of E w.r.t. $L(G)$, which is formally defined as $\kappa_{L(G)}(E) := \cup\{K \in \mathcal{C}(L(G)) | K \subseteq E\}$. A supervisor S that leads to a closed-loop behavior $\kappa_{L(G)}(E)$ is said to be *maximal permissive*. A maximal permissive supervisor can be realized on the basis of a generator of $\kappa_{L(G)}(E)$. The latter can be computed from G and a generator of E. The computational complexity is of order $O(N^2M^2)$, where N and M are the number of states in G and the generator of E, respectively.

A language E is *L_m-closed* if $\overline{E} \cap L_m = E$ and the set of $L_m(G)$-closed languages is denoted $\mathcal{F}_{L_m(G)}$. For specifications $E \in \mathcal{F}_{L_m(G)}$, the plant $L(G)$ is nonblocking under maximal permissive supervision.

3. HIERARCHICAL CONTROL SYSTEMS

For hierarchical control, the event-based scheme in Zhong and Wonham (1990) (compare Figure 2) is used. The detailed plant model G and the supervisor S^{lo} form a low-level closed-loop system, indicated by Con^{lo} (control action) and Inf^{lo} (feedback information). Similarly, the high-level closed loop consists of an abstract plant model G^{hi} and the supervisor S^{hi}. The two levels are interconnected via Com^{hilo} and Inf^{lohi}. The former allows S^{hi} to impose high-level control on S^{lo}, the latter drives the abstract plant G^{hi} in accordance to the detailed model. From the perspective of the high-level supervisor, the forward path sequence Com^{hilo}, Con^{lo} is usually designated "command and control", while the feedback path sequence Inf^{lohi}, Inf^{hi} is identified with "report and advise".

3.1 Basic Definitions

Definition 3.1. (Hierarchical Abstraction). Let $G = (X, \Sigma, \delta, x_0, X_m)$ be a DES and $\Sigma^{hi} \subseteq \Sigma$ a set of high-level events. A *reporter map* [1] is a map $\theta : \Sigma^* \to (\Sigma^{hi})^*$ such that (1) $\theta(\varepsilon) = \varepsilon$ and (2) either $\theta(s\sigma) = \theta(s)$ or $\theta(s\sigma) = \theta(s)\sigma^{hi}$, where $\sigma \in \Sigma$, $\sigma^{hi} \in \Sigma^{hi}$. The high-level language is defined by $L^{hi} := \theta(L(G))$. The high-level marking is chosen s.t. $L_m^{hi} \subseteq L^{hi}$, where L_m^{hi} is required to be regular. The canonical recognizer of L_m^{hi} is denoted G^{hi}, and hence, $L(G^{hi}) = L^{hi}$, $L_m(G^{hi}) = L_m^{hi}$. Finally, high-level controllable and uncontrollable events are denoted Σ_c^{hi} and Σ_u^{hi}, respectively, where $\Sigma^{hi} = \Sigma_c^{hi} \cup \Sigma_u^{hi}$, $\Sigma_c^{hi} \cap \Sigma_u^{hi} = \emptyset$. (G, p^{hi}, G^{hi}) is called a hierarchical abstraction.

For our further discussion, we need to define the interconnection of low- and high-level supervisors with the plant.

[1] In the sequel, we focus our attention on the reporter map $\theta = p^{hi}$, where p^{hi} is the natural language projection into $(\Sigma^{hi})^*$.

Definition 3.2. (Hierarchical Control System). Referring to the notation in Definition 3.1, a *hierarchical control system (HCS)* consists of G, G^{hi}, S^{hi} and S^{lo}, where the *high-level supervisor* S^{hi} and the *low-level supervisor* S^{lo} fulfill the following conditions: S^{hi}: $L^{hi} \rightarrow \Gamma^{hi}$ with the high-level control patterns $\Gamma^{hi} := \{\gamma | \Sigma_u^{hi} \subseteq \gamma \subseteq \Sigma^{hi}\}$; and $S^{lo} : L(G) \rightarrow \Gamma$ with $\theta(L(S^{lo}/G)) \subseteq L(S^{hi}/G^{hi})$.

Given a high-level specification $E^{hi} \in \mathcal{F}_{L_m^{hi}}$, we can synthesize S^{hi} such that $L(S^{hi}/G^{hi}) = \kappa_{L^{hi}}(E^{hi})$ with a nonblocking high-level closed-loop. At this stage, the remaining task is to implement high-level control actions for the low-level plant by means of S^{lo}.

Definition 3.3. (Hierarchical Control Problem). Given G, G^{hi}, S^{hi}, find a low-level supervisor S^{lo} as in Definition 3.2 such that the low-level controlled language of the HCS, $L(S^{lo}/G)$, is nonblocking.

For the rest of this paper, we assume that $\Sigma^{hi} \subseteq \Sigma$ and the reporter map $\theta = p^{hi}$ with $p^{hi} : \Sigma^* \rightarrow (\Sigma^{hi})^*$ is chosen. The following lemma is taken from Schmidt et al. (2004) and states that the HCS is hierarchical consistent for a special implementation of the low-level supervisor.

Definition 3.4. (Consistent Implementation). Given a hierarchical abstraction (G, p^{hi}, G^{hi}) and a supervisor S^{hi}, the consistent implementation S^{lo} of S^{hi} is $S^{lo}(s) := S^{hi}(s^{hi}) \cup (\Sigma - \Sigma^{hi})$, for $s \in L(G)$, and $s^{hi} := p^{hi}(s)$. $(G, p^{hi}, G^{hi}, S^{hi}, S^{lo})$ is called a HCS with a consistent implementation.

Lemma 3.1. (Hierarchical Consistency (HC)). If a consistent implementation is chosen then he hierarchical control system in Definition 3.2 is *hierarchically consistent*, i.e. $p^{hi}(L(S^{lo}/G)) = L(S^{hi}/G^{hi})$.

3.2 Local Nonblocking and Marked State Consistency

As a key-tool for achieving nonblocking behavior, the following properties of DES, *local nonblocking and marked state acceptance*, are introduced.

Definition 3.5. (Locally Nonblocking DES). Let (G, p^{hi}, G^{hi}) be a hierarchical abstraction. $s^{hi} \in L^{hi}$ is locally nonblocking if for all $s \in L(G)$ with $p^{hi}(s) = s^{hi}$ and $\forall \sigma \in \Sigma^{hi}(s^{hi})$, $\exists u_\sigma \in (\Sigma - \Sigma^{hi})^*$ s.t. $su_\sigma\sigma \in L(G)$. (G, p^{hi}, G^{hi}) is locally nonblocking if s^{hi} is locally nonblocking.

This means, that every string has a local successor string ending with any of the high-level events which are possible after the corresponding high-level string. The local nonblocking condition is equivalent to the *observer* property in Wonham (2004).

Marked state acceptance is a condition relating high-level and low-level marking.

Definition 3.6. (Marked State Acceptance). Let (G, p^{hi}, G^{hi}) be a hierarchical abstraction and define

$$L_{s^{hi},ex} := \{s \in L(G) | p^{hi}(s) = s^{hi} \wedge \exists \sigma^{hi} \in \Sigma^{hi} \text{ s.t. } s\sigma^{hi} \in L(G)\} \subseteq \Sigma^*.$$

as the set of exit strings of $s^{hi} \in L^{hi}$.
The string $s_m^{hi} \in L_m^{hi}$ is marked state accepting [2] if for all $s_{ex} \in L_{s_m^{hi},ex}$

$$\exists s' \leq s_{ex} \text{ with } p^{hi}(s') = s_m^{hi} \text{ and } s' \in L_m.$$

(G, p^{hi}, G^{hi}) is marked string accepting if s_m^{hi} is marked string accepting for all $s_m^{hi} \in L_m^{hi}$.

This means every exit string corresponding to a marked high-level string has a marked predecessor string in the low level and each low-level marked state can be extended to an exit string if the high-level string has a successor event.

Figure 1 illustrates the above definitions. $\Sigma^{hi} = \{\alpha, \beta\}$ denotes the set of high-level events and a is a low-level event. The entry strings and the exit strings for $\alpha \in L^{hi}$ are $L_{\alpha,e} = \{a\alpha\}$ and $L_{\alpha,ex} = \{a\alpha aa, a\alpha aaa\}$, respectively. The automaton is not locally nonblocking as $\alpha \notin L_{a\alpha aaa,\alpha} = \{\beta\}$ but $\alpha \in \Sigma^{hi}(\alpha)$ for $\alpha \in L^{hi}$. Also marked state acceptance is violated as $a\alpha aa$ does not have a marked predecessor string.

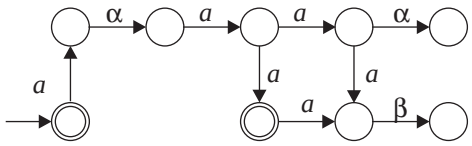

Fig. 1. Illustration of the definitions

Definition 3.7. (Live Regular Language). A regular language $L \in \Sigma^*$ is called live if $\forall s \in L, \exists \sigma \in \Sigma$ s.t. $s\sigma \in L$.

Liveness establishes that after any string in the given language, there is a feasible successor event.

Combining the consistent implementation with marked state acceptance and the locally nonblocking condition, the following theorem states that the proposed control architecture is nonblocking and hierarchically consistent. [3]

Theorem 3.1. (Live Nonblocking Control). Let $(G, p^{hi}, G^{hi}, S^{hi}, S^{lo})$ be a hierarchical control system with a consistent implementation. Also let the projected system (G, p^{hi}, G^{hi}) be marked state accepting and locally nonblocking. If the high-level supervised language $L(S^{hi}/G^{hi})$ is live, then S^{lo} solves the hierarchical control problem in Definition 3.3 and the HCS is hierarchically consistent.

---

[2] Note that $s^{hi} \in L^{hi} - L_m^{hi} \Rightarrow (p^{hi})^{-1}(s^{hi}) \cap L_m = \emptyset$.

[3] Proofs of theorems and lemmas in this paper are given in Schmidt et al. (2005).

4. HIERARCHICAL CONTROL FOR DECENTRALIZED SYSTEMS

In the sequel the hierarchical architecture is applied to decentralized control systems.

Definition 4.1. (Decentralized Control System). A *decentralized control system* (DCS) consists of subsystems, modeled by finite state automata $G_i, i = 1,\ldots,n$ over the respective alphabets Σ_i. The overall system is defined as $G := ||_{i=1}^{n} G_i$ over the alphabet $\Sigma := \bigcup_{i=1}^{n} \Sigma_i$. The controllable and uncontrollable events are $\Sigma_{i,c} := \Sigma_i \cap \Sigma_c$ and $\Sigma_{i,u} := \Sigma_i \cap \Sigma_u$, respectively, where $\Sigma_c \cup \Sigma_u = \Sigma$ and $\Sigma_c \cap \Sigma_u = \emptyset$. For brevity and convenience, let $L := L(G)$, $L_m := L_m(G)$, $L_i := L(G_i)$, and $L_{i,m} := L_m(G_i)$.

Note that each two subsystems G_i, G_j of a DCS are synchronized by shared events if $\Sigma_i \cap \Sigma_j \neq \emptyset$.

The definition of the combined hierarchical and decentralized architecture is given as follows (Figure 2).

Definition 4.2. A Hierarchical and Decentralized Control System (HDCS) consists of the following entities

- A detailed plant model is a decentralized control system G as in Definition 4.1.
- Locally nonblocking low-level controllers are denoted $S_i\colon L_i \to \Gamma_i$, where Γ_i are the respective control patterns. Low-level closed-loop languages are denoted $L_i^c := L(S_i/G_i)$, $L_{i,m}^c := L_i^c \cap L_{i,m}$, $L^c := ||_{i=1}^{n} L_i^c$, $L_m^c := ||_{i=1}^{n} L_{i,m}^c = L^c \cap L_m$. Also let G^c be a generator such that $L^c = L(G^c)$, $L_m^c = L_m(G^c)$.
- (G^c, p^{hi}, G^{hi}) is a hierarchical abstraction and the reporter map: $\theta := p^{hi}$ is used where $p^{hi} : \Sigma^* \to (\Sigma^{hi})^*$ denotes the natural projection with $\bigcup_{i,j,i\neq j}^{n} (\Sigma_i \cap \Sigma_j) \subseteq \Sigma^{hi} \subseteq \Sigma$ and the high level marking is chosen as [4] $L_m^{hi} := p^{hi}(L_m^c)$. High-level controllable events are defined as $\Sigma_c^{hi} := \Sigma_c \cap \Sigma^{hi}$ and $\Sigma_u^{hi} := \Sigma_u \cap \Sigma^{hi}$.
- The high-level supervisor is denoted $S^{hi}\colon L^{hi} \to \Gamma^{hi}$ with the high-level closed-loop language $L(S^{hi}/G^{hi})$ and a valid low-level supervisor $S^{lo}\colon L^c \to \Gamma$ must fulfill $p^{hi}(L(S^{lo}/G^c)) \subseteq L(S^{hi}/G^{hi})$.
- a decentralized implementation of S^{lo} consists of supervisors S_i^{lo} s.t. $L(S_i^{lo}/G_i^c) = p_i(L(S^{lo}/G^c))$.

Lemma 4.1 is taken from Schmidt et al. (2004). It enables the computation of hierarchical abstractions by only using hierarchically abstracted subsystems.

Lemma 4.1. (High Level Plant). Assume the control architecture of Definition 4.2 and let $L_i^{hi} = p^{hi}(L_i^c)$ and $L_{i,m}^{hi} = p^{hi}(L_{i,m}^c)$. Then the high level closed and marked languages are $L^{hi} = p^{hi}(||_{i=1}^{n} L_i^c) = ||_{i=1}^{n} L_i^{hi}$ and $L_m^{hi} = p^{hi}(||_{i=1}^{n} L_{i,m}^c) = ||_{i=1}^{n} L_{i,m}^{hi}$, respectively.

[4] By construction L_m^{hi} is regular.

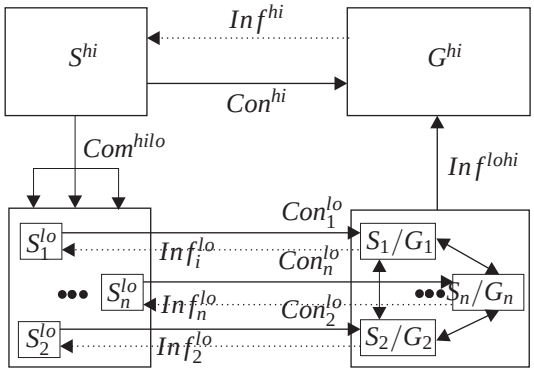

Fig. 2. Control Scheme for HDCS

Mutual controllability ensures that the decentralized subsystems agree on the control action to be executed. It was established in Lee and Wong (2002).

Lemma 4.2. (Mutual Controllability). For a HDCS as in Definition 4.2, let $L_i^{hi} := p^{hi}(L_i)$. If L_i^{hi} and L_j^{hi} are mutually controllable for $i, j = 1,\ldots,n$, i.e.

$$L_j^{hi}(\Sigma_u^{hi} \cap \Sigma_i \cap \Sigma_j) \cap p_j\big((p_i^{hi})^{-1}(L_i^{hi})\big) \subseteq L_j^{hi},$$

then for K^{hi} controllable with respect to L^{hi}, it holds that $p_i(K^{hi})\|(\Sigma_i - \Sigma^{hi}) \cap L_i^c$ is controllable w.r.t. L_i^c.

For the final result we assume that the low-level controlled subsystems G_i^c are marked state accepting and locally nonblocking with respect to Σ^{hi}. The main theorem gives a possible choice of the low-level supervisor such that the proposed control architecture is hierarchically consistent and nonblocking.

Theorem 4.1. (Main Result). Let the hierarchical control architecture for decentralized DES be defined as in Definition 4.2 and let all hierarchical abstractions (G_i^c, G_i^{hi}), $i = 1,\ldots,n$ be marked state accepting and locally nonblocking w.r.t. Σ^{hi} and let the high-level languages L_i^{hi} be mutually controllable. Also chose a standard supervisor implementation for S^{lo} and define the corresponding decentralized supervisors S_i^{lo}

$$S_i^{lo}(p_i(s)) := p_i(S^{lo}(s)),\ i = 1,\ldots,n.$$

If all languages $p_i(L(S^{hi}/G^{hi}))$ are circular [5], then the HDCS is hierarchically consistent and the low-level control is nonblocking and the S_i^{lo} constitute a decentralized implementation of S^{lo}.

For determining the low-level supervisors S_i^{lo}, it is not necessary to compute S^{lo}. Lemma 4.3 shows a decentralized computation of S_i^{lo}.

Lemma 4.3. Let $S_i^{lo}, i = 1,\ldots,n$ be defined as in Theorem 4.1. Then $\forall i, L(S_i^{lo}/G_i^c) = \kappa_{L_i^c}(E_i)$ with $E_i = p_i\big(L(S^{hi}/G^{hi})\big)\|(\Sigma_i - \Sigma^{hi})$.

[5] In this paper, circularity is required. There are other sufficient conditions which give the same result

Remark: The proposed architecture readily extends to a multi-level hierarchy. For notational convenience the two-level hierarchy was chosen in this paper.

5. EXAMPLE

Our method shall be demonstrated by an example. The control system is part of a larger production plant and its purpose is to distribute workpieces (see Figure 3).

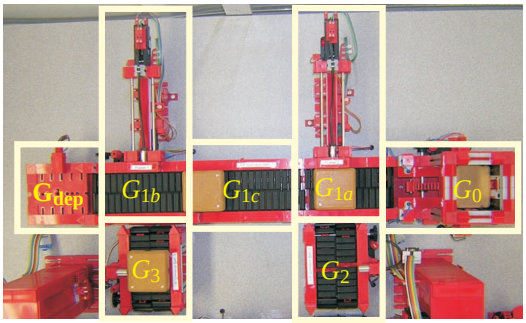

Fig. 3. Distribution system

5.1 Synthesis on Level 0

For sake of clarity, we will give a detailed description of the controller synthesis on the lowest level (level 0) for the subsystem G_{1a}.

Subsystem $G_{1a}^{(0)}$ represents the section of the long conveyor belt between the stack feeder (G_0) and the sensor detecting arrival of workpieces at the right pusher. The discrete event model of $G_{1a}^{(0)}$ describes the motion on the conveyor belt as well as the motion of the pusher on the lowest level transporting workpieces to G_2. It takes into account sensor readings and actuator commands and has 83 states. There are more efficient ways of modeling the plant but the focus of this work is to generate supervisors which can be implemented directly on a PLC. The specification $E_{1a}^{(0)}$, stating that either the conveyor belt or the pusher is allowed to move, yields a supervisor $S_{1a}^{(0)}$ s.t. $L(S_{1a}^{(0)}/G_{1a}^{(0)}) = \kappa_{L(G_{1a}^{(0)})}(E_{1a}^{(0)})$. Figure 4 shows the canonical recognizer $G_{1a}^{(0),c}$ for $L(S_{1a}^{(0)}/G_{1a}^{(0)})$ and Table 5.1 provides event definitions.

a	workpiece (wp) from G_0 to G_1	b	G_1 starts moving
c	wp from G_1 to G_2	d	G_1 stops
e	wp arrives at G_{pu2}	f	wp leaves G_{pu2}
g	wp from G_1 to G_3	i	G_{pu2} extends
h	wp from G_1 to G_{dep}	m	G_{pu2} retracts
n	G_{pu2} leaves extended position	l	G_{pu2} stops
j	G_{pu2} leaves rest position	k	G_{pu2} is extended
o	G_{pu2} arrives at rest position	q	wp leaves G_{dep}
p	G_{pu2} ready for next transport	u	G_3 is empty
s	wp arrives at G_{dep}	t	push wp to G_3
r	wp arrives at G_{pu1}	v	G_2 is empty

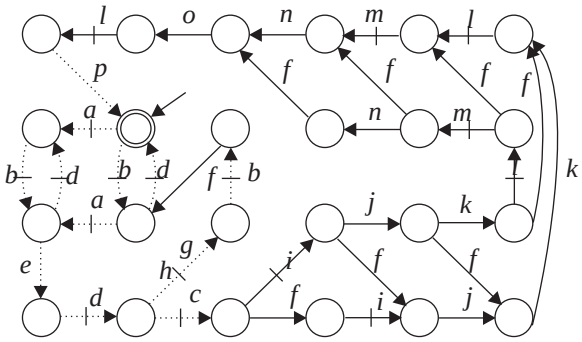

Fig. 4. Low-level plant $G_{1a}^{(0),c}$

Now the level 1 abstraction can be computed with the relevant high-level events $\Sigma_{1a}^{(1)} = \{a,b,c,d,e,g,h,p\}$. Projecting $L(G^{(0),c})$ on $\Sigma_{1a}^{(1)}$ yields $L(G_{1a}^{(1)})$ (see Figure 5). It can easily be verified that $G_{1a}^{(0),c}$ is marked state accepting and locally nonblocking wrt. $\Sigma_{1a}^{(1)}$.

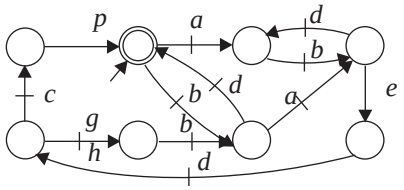

Fig. 5. Abstracted plant $G_{1a}^{(1)}$

The abstractions $G_{1b}^{(1)}$, $G_{1c}^{(1)}$ and $G_{dep}^{(1)}$ are obtained by applying analogous computations.

5.2 Synthesis on Level 1

Applying Lemma 4.1 to the abstracted subsystems, the level 1 plant is $G_1^{(1)} = G_{1a}^{(1)} \| G_{1b}^{(1)} \| G_{1c}^{(1)} \| G_{dep}^{(1)}$. it has 271 states, and it can be verified that the decentralized subplants are mutually controllable. According to Theorem 4.1, a supervisory controller can be synthesized on level 1 and translated to level 0. The specifications are (see Figure 6):

- the long conveyor belt must stop ($\{d\}$) when an event from $\{e,r,s\}$ occurs and only then ($E_1^{(1)}$).
- no more workpiece shall be transported to the depot if it is full ($E_2^{(1)}$).
- workpieces shall be delivered in the order G_3 - G_3 - G_2 ($E_3^{(1)}$).

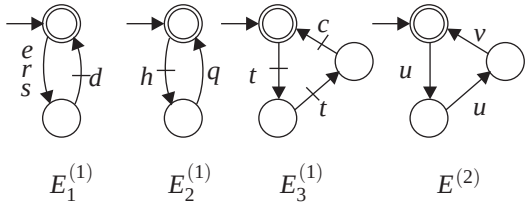

Fig. 6. Specifications

$G_{1a}^{(0)}$	$G_{1b}^{(0)}$	$G_{1c}^{(0)}$	$G_{dep}^{(0)}$	$G_{sf}^{(0)}$	$G_2^{(0)}$	$G_3^{(0)}$
83	63	13	5	8	4	4
$E_{1a}^{(0)}$	$E_{1b}^{(0)}$	$E_{1c}^{(0)}$	$E_{dep}^{(0)}$	$E_{sf}^{(0)}$	$\|\|_{i=1}^{3} E_i^{(1)}$	$E^{(2)}$
2	2	7	-	9	12	3

Table 1. Plant and Specifications

The level 1 supervisor is $S_1^{(1)}$ with $L(S_1^{(1)}/G_1^{(1)}) = \kappa_{L(G_1^{(1)})}\big(\bigcup_{i=1}^{3} E_i^{(1)}\big) \cap \mathcal{F}_{L_m(G_1^{(1)})}$ and the level 1 supervised plant has 65 states. It can also be verified that the projected level 1 controlled languages $p_i(L(S_1^{(1)}/G_1^{(1)}))$ are circular for $i \in \{1a, 1b, 1c, dep\}$.

5.3 Exemplary Level 0 Supervisor Implementation

The decentralized implementation of supervisor $S_1^{(1)}$ on level 0 shall be demonstrated using the subsystem $G_{1a}^{(0),c}$. According to Lemma 4.3, a supervisor S_{1a}^{1-0} s.t. $L(S_{1a}^{1-0}/G_{1a}^{(0),c}) = \kappa_{L(G_{1a}^{(0),c})}(E_{1a}^{1-0})$ with $E_{1a}^{1-0} = p_{1a}^{(0)}\big(L(S_1^{(1)}/G_1^{(1)})\|(\Sigma_{1a} - \Sigma_{1a}^{(1)})^*\big)$ with $p_{1a}^{(0)} : (\Sigma^{(0)})^* \to (\Sigma_{1a}^{(0)})^*$ and the automaton $S_{1a}^{1-0}/G_{1a}^{(0),c}$ has 80 states. Analogous computations for the other subplants yield local supervisors acting in parallel and guaranteeing the specified system behavior.

5.4 Implementation Results

For classifying the computational effort of synthesis and implementation, the number of states of the supervisor implementations of the monolithic synthesis and of the proposed method are compared.

Canonical recognizers of the plant and the overall specification have $10 \cdot 10^6$ and $8 \cdot 10^3$ reachable states, respectively (see Table 5.4 for the composition of plant and specifications). The monolithic supervisor synthesis yields a supervised plant with $4.7 \cdot 10^6$ reachable states. While there exist more efficient models for this system our model pragmatically refers to real world sensors and actuators. This is a prerequisite for automatic generation of PLC code.

S_0^{1-0}/G_0^0	S_{1a}^{1-0}/G_{1a}^0	S_{1b}^{1-0}/G_{1b}^0	S_{1c}^{1-0}/G_{1c}^0
12	80	87	7
S_{dep}^{1-0}/G_{dep}^0	S_2^{1-0}/G_2^0	S_3^{1-0}/G_3^0	
5	11	11	

Table 2. Decentralized supervisors

For computing the controller for the whole system, one more level of the hierarchy is added to our approach, consisting of an abstraction of $G_1^{(1),c)}$ (9 states), $G_2^{(2)}$ (2 states) and $G_3^{(2)}$ (2 states). Implementing the specification $E^{(2)}$ which states that the workpieces shall leave the plant in the order G_3 - G_3 - G_2 the controlled plant on level 2 has 18 states and the level 2 supervisor action can be translated to level 0 as shown before, yielding 7 decentralized supervisors (see Table 5.4). It is essential to note that the computation of the overall plant is not necessary and that the supervisors have a small number of states, whereas a monolithic implementation is not advisable.

6. CONCLUSIONS

In this paper, the decentralized structure of a discrete event system has been used to reduce the complexity of the supervisor design procedure as well as the complexity of the supervisor implementation. To this end, a hierarchical abstraction which respects the decentralized structure of the system was introduced and a hierarchical supervisor design procedure was elaborated. Furthermore nonblocking behavior and hierarchical consistency of the supervised plant was proven. The results were illustrated by a real world example. Algorithms for the proposed method have been implemented and ongoing work provides a solution to automatic PLC code generation.

REFERENCES

C.G Cassandras and S. Lafortune. Introduction to discrete event systems. *Kluwer*, 1999.

A.E.C. da Cunha, J.E.R. Cury, and B.H. Krogh. An assume guarantee reasoning for hierarchical coordination of discrete event systems. *WODES*, 2002.

P. Hubbard and P.E. Caines. Dynamical consistency in hierarchical supervisory control. *IEEE TAC*, 2002.

J. Komenda and J. H. van Schuppen. Decentralized control with coalgebra. *ECC*, 2003.

R.J. Leduc, W.M. Wonham, and M. Lawford. Hierarchical interface-based supervisory control: Parallal case. *Allerton Conf. on Comm., Contr. and Comp.*, 2001.

S-H. Lee and K.C. Wong. Stuctural decentralised control of concurrent DES. *EJC*, 2002.

K. Schmidt, J. Reger, and T. Moor. Hierarchical control of structural decentralized DES. *WODES*, 2004.

K. Schmidt, S.Perk, and T. Moor. Noblocking hierarchical control of decentralized DES. *Technical Report, Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg*, 2005.

K. Wong. On the complexity of projections of discrete-event systems., 1997. URL `citeseer.nj.nec.com/wong98complexity.html`.

K.C. Wong and W.M. Wonham. Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems*, 1996.

W.M Wonham. Notes on control of discrete event systems. *Department of Electrical & Computer Engineering, University of Toronto*, 2004.

T. Yoo and S. Lafortune. A generalized framework for decentralized supervisory control of discrete event systems. *WODES*, 2000.

H. Zhong and W.M. Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE TAC*, 1990.