Nonblocking Hierarchical Control of Decentralized Discrete Event Systems

Klaus Schmidt, Thomas Moor, and Sebastian Perk

***Abstract*—This contribution investigates the hierarchical control of decentralized discrete event systems (DES) that are synchronized by shared events. A hierarchical control architecture providing hierarchical consistency is introduced. Moreover, it allows for composition of decentralized subsystems on the high-level of the hierarchy and hence reduces the computational complexity of supervisory control synthesis for language inclusion specifications. In this context, a crucial issue is the nonblocking operation of the overall system. Our main theorem identifies sufficient conditions for this desirable property.**

***Index Terms*—Decentralized control, discrete event systems (DES), hierarchical control, large-scale systems, supervisory control.**

I. Introduction

In the past decade a great variety of ideas have been studied to reduce the complexity of synthesis algorithms for the supervisory control of discrete event systems. A key ingredient of promising approaches is to assume or to 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 work addresses this issue for compound plants, that is, systems that are composed of a set of subsystems. In order to reduce the complexity of supervisor synthesis, we propose a hierarchical design that introduces a vertical control structure, while preserving the given horizontal structure of the compound plant. We use the *natural projection* for information aggregation, where we require that the set of events that are shared between at least two subsystems has to be contained in the high-level alphabet. This allows the computation of the hierarchical abstraction by first projecting the subsystems and then composing them to an overall high-level system, i.e., the overall low-level model need not be evaluated explicitly. As we use the same modeling framework for both the low and the high level, the method qualifies for a multilevel hierarchy. We perform supervisor synthesis for the high-level system and derive a low-level realization of the high-level supervisor for each subsystem. It guarantees *hierarchical consistency* of the hierarchical architecture, i.e., the desired high-level behavior is implemented by low-level control. Additionally, we introduce *local nonblocking*, *marked string acceptance* and *marked string controllability* as structural conditions to achieve nonblocking behavior of our hierarchical and decentralized multilevel architecture.

Several other approaches address the problem of reducing the complexity of supervisor synthesis. In *modular control* [1], monolithic supervisors for different specifications are designed and implemented together. Although controllability can be verified easily, checking if the modular supervisors are nonconflicting, that is, if their joint action is nonblocking, is computationally expensive. An improvement of this technique is given in [2], where the plant is considered as a composite system. The modular computation of controllable sublanguages of a specification language is elaborated in [3], where abstractions of the composite plant are used to avoid the composition of the system components.

In contrast to the modular approach, *decentralized control* focuses on the computation of decentralized supervisors that only have partial observations of the plant events as investigated in [4]. Recent work also includes communication between the supervisors. Extensions of the method to compound systems with modular specifications are given in [5].

A different view on the decentralized control of compound discrete event systems is taken in [6], [7]. Purely structural conditions of the subsystems are used. The overall system does not have to be computed, and thus, after verifying the required system properties, supervisor synthesis can be performed with a smaller computational effort than for the monolithic approach. In our work, we extend the idea of identifying structural system properties for both horizontal and vertical system compositions.

The above approaches which focus on the horizontal structure of discrete event systems are supplemented by hierarchical methods that make use of their vertical structure. Hierarchical system models can be constructed "top-down" as in [8], [9], i.e., a high-level model (for example an automaton) of the system is generated first, and the structural components of the high-level model are filled with more detailed information, or "bottom-up". Then, the low-level model is abstracted to higher levels by aggregation of information (see [10]–[19]). The hierarchical design proposed in this paper belongs to the second category.

The development of bottom-up hierarchical control techniques was initiated by the work in [10]. A *reporter map* is employed for hierarchical abstraction, and *output control consistency* is introduced as a structural condition to guarantee *hierarchical consistency*, where marking is not considered. The abstraction of the control mechanism is generalized in [11]

Manuscript received October 03, 2006; revised August 17, 2007 and March 27, 2008. Current version published November 05, 2008. Recommended by Associate Editor J. E. R. Cury.

The authors are with the Chair of Automatic Control, University of Erlangen-Nuremberg, Erlangen 91058, Germany (e-mail: klaus.schmidt@rt.eei.uni-erlangen.de; thomas.moor@rt.eei.uni-erlangen.de; sebastian.perk@rt.eei.uni-erlangen.de).

Digital Object Identifier 10.1109/TAC.2008.2006817

by introducing *control structures*. Additionally, *consistency of marking* between hierarchical levels helps guaranteeing nonblocking system behavior. The idea of *control structures* is further elaborated in [12]. On the low level of the hierarchy, the Ramadge/Wonham (RW) framework is used. On the high level, the concept of control structures is combined with a flexible marking function to render conditions such as consistency of marking unnecessary. The work in [14], [20] presents a hierarchical design method for systems with an input/output structure in the behavioral framework, where it is possible to use a high-level specification as the system abstraction. The approaches in [13], [21] perform an aggregation (partition) of the state space to obtain an abstraction of the low-level system. Elaborating on the connectivity of the high-level states, nonblocking supervisors are designed.

A method that relates to the present paper in that it explicitly uses structural information of the discrete event plant is presented in [22]. It is based on interfaces, which indicate how the hierarchical levels can interact, and enables the computation of nonblocking supervisors for large-scale systems.

A control architecture that combines hierarchical and decentralized control for compound systems while imposing nonblocking closed-loop behavior is presented in [15] by the authors. A number of extensions and variations have been discussed in [16]–[19].

Technically, this paper builds on results that were obtained in [18] and is organized as follows. In Section II, the basic notation is presented, and a framework for the hierarchical control of discrete event systems is introduced. Based on this control framework, the main issues to be addressed in this paper are outlined. In Section III, we elaborate our method for hierarchical abstraction and then focus on the low-level implementation of a high-level supervisor in Section IV. We also present sufficient conditions for nonblocking and hierarchically consistent supervisory control. The applicability of the approach to large-scale composite systems is demonstrated with a laboratory case study in Section V.

II. Preliminaries

The work is based on the supervisory control theory for discrete event systems [23], [24]. At first, the notation and basic concepts are briefly summarized.

Notation

For a finite alphabet Σ, the set of all finite strings over Σ is denoted Σ^* (Kleene-closure). We write $s_1 s_2 \in \Sigma^*$ for the concatenation of two strings s_1, $s_2 \in \Sigma^*$ and $s_1 \leq s$ 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 $\epsilon \in \Sigma^*$, i.e., $s\epsilon = \epsilon s = s$ for all $s \in \Sigma^*$. A *language* over Σ is a subset $L \subseteq \Sigma^*$. The *prefix closure* of L is defined by $\overline{L} := \{s_1 \in \Sigma^* | \exists s \in L \text{ s.t. } s_1 \leq s\}$. A language L is *prefix closed* if $L = \overline{L}$. For any string $s \in L$, $\Sigma_L(s) := \{\sigma | s\sigma \in \overline{L}\}$ is the set of eligible events after s. The concatenation and the Kleene-closure are naturally extended to languages $L_1, L_2 \in \Sigma^*$ by defining $L_1 L_2 := \{s_1 s_2 \in \Sigma^* | s_1 \in L_1 \wedge s_2 \in L_2\}$ and $L^* := \{\epsilon\} \cup L \cup LL \cup \cdots$.

A relevant subclass of the languages over Σ is the class of *regular* languages, which represents the class of languages that is constructed by regular expressions over Σ [25].

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 $p_i(\epsilon) := \epsilon$; 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* $L_1 || L_2 \subseteq \Sigma^*$ of two languages $L_i \subseteq \Sigma_i^*$ is $L_1 || L_2 = p_1^{-1}(L_1) \cap p_2^{-1}(L_2) \subseteq \Sigma^*$. It can be shown that the natural projection, the inverse projection and the synchronous product of regular languages are again regular languages.

Similar to the notation in [26], we model a discrete event system as a tuple $H = (L, L_{\mathrm{m}})$ of two regular languages $L, L_{\mathrm{m}} \subseteq \Sigma^*$, where L is prefix-closed and $L_{\mathrm{m}} \subseteq L$. The language L_{m} describes the desirable strings of the system (for example strings which indicate the termination of a task). We refer to these strings as *marked strings*. L includes all strings which can be generated by the system (in particular prefixes of strings in L_{m}). As both L and L_{m} are required to be regular, any system can be represented as a finite automaton [18].[1] For convenience, we introduce the canonical projection operators $\mathcal{L}$ and $\mathcal{L}_{\mathrm{m}}$ such that $\mathcal{L}(H) := L, \mathcal{L}_{\mathrm{m}}(H) := L_{\mathrm{m}}$. The natural projection $p : \Sigma^* \to \Sigma_0^*$ and the synchronous composition $L_1 || L_2$ are extended to systems with $p(H) := (p(\mathcal{L}(H)), p(\mathcal{L}_{\mathrm{m}}(H)))$ and $H_1 || H_2 := (\mathcal{L}(H_1) || \mathcal{L}(H_2), \mathcal{L}_{\mathrm{m}}(H_1) || \mathcal{L}_{\mathrm{m}}(H_2))$, respectively. Note that $p(H)$ and $H_1 || H_2$ are again systems.

We define a subset relation for systems H_1 and H_2 with the same alphabet Σ, i.e., $\mathcal{L}(H_i), \mathcal{L}_{\mathrm{m}}(H_i) \subseteq \Sigma^*$. H_2 is a subset of H_1 if the languages of H_2 are subsets of the respective languages of H_1, i.e., $H_2 \subseteq H_1$ iff $\mathcal{L}(H_2) \subseteq \mathcal{L}(H_1)$ and $\mathcal{L}_{\mathrm{m}}(H_2) \subseteq \mathcal{L}_{\mathrm{m}}(H_1)$.

In the supervisory control context, we write $\Sigma = \Sigma_{\mathrm{c}} \dot{\cup} \Sigma_{\mathrm{uc}}$, where Σ_{uc} is called the set of *uncontrollable events* and Σ_{c} is the set of *controllable events*. Considering that H is a system with the alphabet Σ partitioned as above, we denote H as a *control system* whenever the sets of controllable and uncontrollable events are clear from the context. A *control pattern* is a set γ, $\Sigma_{\mathrm{uc}} \subseteq \gamma \subseteq \Sigma$, and the set of all control patterns is denoted $\Gamma \subseteq 2^{\Sigma}$. A *supervisor* for a control system H is a map $S: \mathcal{L}(H) \to \Gamma$, where $S(s)$ represents the set of enabled events after the occurrence of the string $s \in \mathcal{L}(H)$; i.e., a supervisor can disable controllable events only. A system S/H denotes a control system H under supervision by S. The language $\mathcal{L}(S/H)$ generated by S/H is iteratively defined by (1) $\epsilon \in \mathcal{L}(S/H)$ and (2) $s\sigma \in \mathcal{L}(S/H)$ iff $s \in \mathcal{L}(S/H), \sigma \in S(s)$ and $s\sigma \in \mathcal{L}(H)$. Thus, $\mathcal{L}(S/H)$ represents the behavior of the *closed-loop system*. To take into account the marked strings of H, let $\mathcal{L}_{\mathrm{m}}(S/H) := \mathcal{L}(S/H) \cap \mathcal{L}_{\mathrm{m}}(H)$. The closed-loop system is *nonblocking* if $\overline{\mathcal{L}_{\mathrm{m}}(S/H)} = \mathcal{L}(S/H)$, i.e., if each string in $\mathcal{L}(S/H)$ is the prefix of a marked string in $\mathcal{L}_{\mathrm{m}}(S/H)$. Note that S/H is again a control system ([18]) with $\Sigma = \Sigma_{\mathrm{c}} \dot{\cup} \Sigma_{\mathrm{uc}}$.

A language $K \subseteq \Sigma^*$ is said to be controllable w.r.t. a system H and a set of uncontrollable events $\Sigma_{\mathrm{uc}} \subseteq \Sigma$ if $\overline{K}\Sigma_{\mathrm{uc}} \cap \mathcal{L}(H) \subseteq \overline{K}$. The set of all sublanguages that are controllable

[1] In this work, the choice of the tuple (L, L_{m}) as a system model facilitates the theoretical considerations. Nevertheless, all systems can be represented by finite automata. Thus, all computations can be carried out on finite sets.

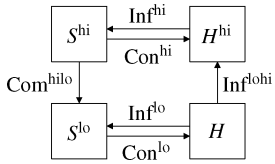


Fig. 1. Hierarchical control architecture.

w.r.t. H and Σ_{uc} is denoted $\mathcal{C}(H)$. It is characterized by $\mathcal{C}(H) = \{K \subseteq \mathcal{L}(H) | \ \overline{K}\Sigma_{\mathrm{uc}} \cap \mathcal{L}(H) \subseteq \overline{K}\}$. Furthermore, the set $\mathcal{C}(H)$ is closed under arbitrary union (this result is analogous to what is presented in [24]). Hence, for every *specification* language E, there uniquely exists a *supremal controllable sublanguage* of E w.r.t. H and Σ_{uc}, which is formally defined as $\kappa_{H,\Sigma_{\mathrm{uc}}}(E) := \cup\{K \in \mathcal{C}(H) | \ K \subseteq E\}$. A supervisor S that leads to a closed-loop behavior $\mathcal{L}_{\mathrm{m}}(S/H) = \kappa_{H,\Sigma_{\mathrm{uc}}}(E)$ is said to be *maximally permissive*.

The language $E \in \Sigma^*$ is *L-closed* for a language $L \in \Sigma^*$, if $\overline{E} \cap L = E$. Nonblocking maximally permissive supervision is guaranteed for a control system H over Σ, if and only if the specification E is $\mathcal{L}_{\mathrm{m}}(H)$-closed and controllable w.r.t. H and $\Sigma_{\mathrm{uc}} \subseteq \Sigma$. A maximally permissive supervisor can be realized by an automata representation of $\kappa_{H,\Sigma_{\mathrm{uc}}}(E)$. The latter can be computed from an automata representation of H and E with a computational complexity of order $O(N^2M^2)$, where N, M is the respective number of states of the automata for H and E.

A. Hierarchical Control

Although the complexity of supervisor synthesis is polynomial in time, it is not always practical to compute a supervisor as the number of states of a discrete event system which is composed of several components grows exponentially with its number of components. *Hierarchical control* addresses this state space explosion problem by introducing a vertical system structure, see Fig. 1 ([10]).

On the low level, a control system H describes the detailed behavior of the given system. The idea of hierarchical control is to compute a supposedly smaller high-level model H^{hi}, and to perform the supervisor synthesis using H^{hi}. Thus, the high-level closed-loop consists of the abstracted plant model H^{hi} and the supervisor S^{hi} with the feedback information $\mathrm{Inf}^{\mathrm{hi}}$ and the control action $\mathrm{Con}^{\mathrm{hi}}$. The abstraction process is indicated by the information channel $\mathrm{Inf}^{\mathrm{lohi}}$. As the control action of the high-level supervisor S^{hi} on H^{hi} is just virtual, it must be translated to the control action of a low-level supervisor S^{lo}, which then directly controls the low-level system H. This translation is symbolized by the command channel $\mathrm{Com}^{\mathrm{hilo}}$. Together, H and S^{lo} form a low-level closed-loop system, indicated by $\mathrm{Con}^{\mathrm{lo}}$ (control action from the supervisor) and $\mathrm{Inf}^{\mathrm{lo}}$ (feedback information from the control system).

III. Hierarchical Abstraction

In real-world applications, large-scale discrete event systems are composed of smaller subsystems that interact with each other. A natural way to describe such a system is to model each individual component by a control system. The overall system behavior is then given by the synchronous composition of the subsystems. Based on this low-level model of a compound system, we point out issues that have to be addressed in the hierarchical abstraction process, and we develop a hierarchical abstraction method that conforms to the above issues. We also show that the high-level model can be computed efficiently.

A. Low-Level System Model

Definition 3.1 formalizes our model for compound systems.

Definition 3.1 (Decentralized Control System): A *decentralized control system* (DCS) is a family $\{H_i\}_{i\leq n}$ of n control systems H_i with the respective alphabets Σ_i, $i = 1, \ldots, n$. The controllable and uncontrollable events of a component H_i are denoted $\Sigma_{i,\mathrm{c}}$ and $\Sigma_{i,\mathrm{uc}}$, respectively. We assume that all components agree on the controllability status of the shared events, i.e., $\Sigma_{k,\mathrm{uc}} \cap \Sigma_{j,\mathrm{c}} = \emptyset$ for all $j, k = 1, \ldots, n$ with $k \neq j$. The overall control system is $H := ||_{i=1}^{n} H_i$ with the alphabet $\Sigma := \bigcup_{i=1}^{n} \Sigma_i$, the controllable events $\Sigma_{\mathrm{c}} := \bigcup_{i=1}^{n} \Sigma_{i,\mathrm{c}}$ and the uncontrollable events $\Sigma_{\mathrm{uc}} := \bigcup_{i=1}^{n} \Sigma_{i,\mathrm{uc}}$.

We model the low-level system H in Fig. 1 as a decentralized control system.

B. Hierarchical Abstraction of Decentralized Systems

Given a DCS on the low level, we now turn to the development of our hierarchical abstraction method. Particularly, the following two issues will be addressed.

i) Decentralized control systems possess an inherent structure due to the interaction of their components. This structural information gets lost when the system is composed to a monolithic DES. Considering this argument and the fact that it is not practical to compose large-scale DCS because of the state space explosion, we want to retain the decentralized system structure in the abstraction process.

ii) It is possible that although all individual components of a DCS are nonblocking, the overall system can be blocking. This property is captured by the *nonconflicting* condition [1]. A DCS $\{H_i\}_{i\leq n}$ with nonblocking subsystems H_i is nonconflicting if it holds that

$$||_{i=1}^{n} \mathcal{L}(H_i) = ||_{i=1}^{n} \overline{\mathcal{L}_{\mathrm{m}}(H_i)} = \overline{||_{i=1}^{n} \mathcal{L}_{\mathrm{m}}(H_i)}.$$

Conflicts in DCSs are always caused by disagreement of different components on the occurrence of shared events in their synchronized behavior. This observation suggests to retain the information about the synchronized occurrence of shared events in the abstraction step.

In DCSs, the shared behavior is represented by the occurrence of shared events $\Sigma_{\mathrm{s}} := \bigcup_{i,j=1,i\neq j}^{n} (\Sigma_i \cap \Sigma_j)$. Therefore, we use the natural projection $p^{\mathrm{hi}} : \Sigma^* \to (\Sigma^{\mathrm{hi}})^*$ from the system alphabet Σ to a high-level alphabet Σ^{hi} for hierarchical abstraction, where Σ^{hi} is chosen such that $\Sigma_{\mathrm{s}} \subseteq \Sigma^{\mathrm{hi}} \subseteq \Sigma$, that is, all events shared between at least two subsystems H_i and H_j are contained in Σ^{hi}. This results in the high-level control system $H^{\mathrm{hi}} = p^{\mathrm{hi}}(H)$ in Fig. 1. However, this way of computing H^{hi} involves the composition of the overall low-level system H which has to be avoided according to item (i) in the above discussion. The following proposition provides a solution to this problem.

Proposition 3.1 (High Level Plant [15], [18]): Let $\{H_i\}_{i\leq n}$ be a DCS with Σ^{hi} s.t. $\Sigma_{\mathrm{s}} \subseteq \Sigma^{\mathrm{hi}} \subseteq \Sigma$, and let $p^{\mathrm{hi}} : \Sigma^* \to$

$(\Sigma^{\mathrm{hi}})^*$ be the natural projection. Also define the decentralized high-level alphabets as $\Sigma_i^{\mathrm{hi}} := \Sigma^{\mathrm{hi}} \cap \Sigma_i$ with the corresponding decentralized natural projections $p_i^{\mathrm{dec}} : \Sigma_i^* \to (\Sigma_i^{\mathrm{hi}})^*$ for $i = 1, \ldots, n$. Then,

$$H^{\mathrm{hi}} = p^{\mathrm{hi}}(H) = p^{\mathrm{hi}}(\|_{i=1}^n H_i) = \|_{i=1}^n p_i^{\mathrm{dec}}(H_i).$$

The proof of Proposition 3.1 is based on the extension of a result in [24] as pointed out in Lemma 3.1 and Corollary 3.1. Related proofs can be found in [15], [18].

Lemma 3.1 ([24]): Let Σ_1 and Σ_2 be alphabets and let $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$. Assume $\Sigma_0 \subseteq \Sigma_1 \cup \Sigma_2$ and $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma_0$ with the natural projections $p_0 : (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_0^*$, $p_1' : \Sigma_1^* \to (\Sigma_0 \cap \Sigma_1)^*$ and $p_2' : \Sigma_2^* \to (\Sigma_0 \cap \Sigma_2)^*$. Then $p_0(L_1 \| L_2) = p_1'(L_1) \| p_2'(L_2)$.

Corollary 3.1 follows by iterative application of Lemma 3.1.

Corollary 3.1 ([18]): 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 $(\Sigma_i \cap \Sigma_j) \subseteq \Sigma_0$ for all $i, j = 1, \ldots, n$ with $i \neq j$, and define 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

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

Now Corollary 3.1 can be applied to the languages $\mathcal{L}(H_i)$ and $\mathcal{L}_{\mathrm{m}}(H_i)$, $i = 1, \ldots, n$ in Proposition 3.1.

Proof: With Corollary 3.1, it holds that $p^{\mathrm{hi}}(\mathcal{L}(H)) = p_1^{\mathrm{dec}}(\mathcal{L}(H_1)) \| \cdots \| p_n^{\mathrm{dec}}(\mathcal{L}(H_n))$ and $p^{\mathrm{hi}}(\mathcal{L}_{\mathrm{m}}(H)) = p_1^{\mathrm{dec}}(\mathcal{L}_{\mathrm{m}}(H_1)) \| \cdots \| p_n^{\mathrm{dec}}(\mathcal{L}_{\mathrm{m}}(H_n))$. This means that $p^{\mathrm{hi}}(H) = p_1^{\mathrm{dec}}(H_1) \| \cdots \| p_n^{\mathrm{dec}}(H_n)$. ∎

Proposition 3.1 reduces the computational complexity of the projection operation for decentralized systems. It is no longer necessary to compute the overall low-level control system, and then project it to the high level, but it is possible to project the decentralized subsystems to the high level first, and then compose the projected high-level systems to form the overall high-level control system.

The described hierarchical abstraction method for DCS is formalized in the following definition.

Definition 3.2 (Projected Decentralized Control System): Let $\{H_i\}_{i \leq n}$ be a DCS with Σ^{hi} s.t. $\Sigma_{\mathrm{s}} \subseteq \Sigma^{\mathrm{hi}} \subseteq \Sigma$, and let $p^{\mathrm{hi}} : \Sigma^* \to (\Sigma^{\mathrm{hi}})^*$ be the natural projection. Also define the decentralized high-level alphabets as $\Sigma_i^{\mathrm{hi}} := \Sigma^{\mathrm{hi}} \cap \Sigma_i$ with the corresponding decentralized natural projections $p_i^{\mathrm{dec}} : \Sigma_i^* \to (\Sigma_i^{\mathrm{hi}})^*$ for $i = 1, \ldots, n$. A *projected decentralized control system* (PDCS) is a triple $(\{H_i\}_{i \leq n}, p^{\mathrm{hi}}, \{H_i^{\mathrm{hi}}\}_{i \leq n})$, where $H_i^{\mathrm{hi}} := p_i^{\mathrm{dec}}(H_i)$, $i = 1, \ldots, n$. The overall high-level system is $H^{\mathrm{hi}} = \|_{i=1}^n H_i^{\mathrm{hi}}$, and the high-level controllable and uncontrollable events are defined as $\Sigma_{\mathrm{c}}^{\mathrm{hi}} := \Sigma_{\mathrm{c}} \cap \Sigma^{\mathrm{hi}}$ and $\Sigma_{\mathrm{uc}}^{\mathrm{hi}} := \Sigma_{\mathrm{uc}} \cap \Sigma^{\mathrm{hi}}$, respectively. We denote the triple $(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$ a *projected control system* (PCS).

Fig. 2(a) illustrates the concept of the PDCS with the DCS $\{H_i\}_{i \leq n}$ on the low level, the projections p_i^{dec}, $i = 1, \ldots, n$ and the high-level system $H^{\mathrm{hi}} = \|_{i=1}^n H_i^{\mathrm{hi}}$. It can be shown that $\{H_i^{\mathrm{hi}}\}_{i \leq n}$ is again a DCS.

The hierarchical abstraction for decentralized control systems is explained in the following example.

Example 3.1: Consider $\mathcal{L}_{\mathrm{m}}(H_1) = \epsilon + \alpha\mathtt{a}(\epsilon + \mathtt{c}\beta)$, $\mathcal{L}(H_1) = \overline{\alpha(\mathtt{bc}(\mathtt{bc})^*\beta + \mathtt{a}(\mathtt{b} + \mathtt{c}\beta))}$, $\mathcal{L}_{\mathrm{m}}(H_2) = \alpha\mathtt{d}(\epsilon + \beta + \gamma\mathtt{e})$ and

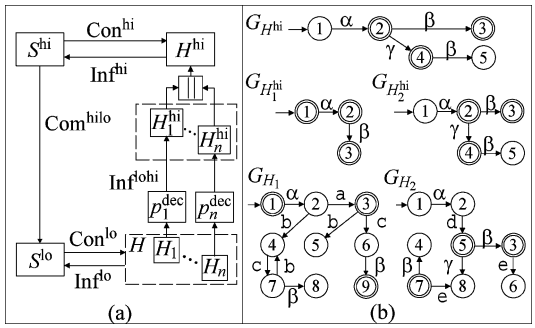


Fig. 2. (a) PDCS construction; (b) Example of a PDCS.

$\mathcal{L}(H_2) = \overline{\alpha\mathtt{d}(\beta\mathtt{e} + \gamma\mathtt{e}\beta)}$. A graphical representation of the respective control systems in the form of finite automata is given in Fig. 2(b). Here, the automaton G_H corresponding to a control system H is defined such that its *closed language* is $\mathcal{L}(G_H) = \mathcal{L}(H)$ and its *marked language* is $\mathcal{L}_{\mathrm{m}}(G_H) = \mathcal{L}_{\mathrm{m}}(H)$.

The shared events of the two subsystems are $\Sigma_{\mathrm{s}} = \Sigma_1 \cap \Sigma_2 = \{\alpha, \beta\}$. We define the high-level alphabet as $\Sigma^{\mathrm{hi}} := \{\alpha, \beta, \gamma\} \supseteq \Sigma_{\mathrm{s}}$. The high-level alphabets for the two components are then $\Sigma_1^{\mathrm{hi}} = \{\alpha, \beta\}$ and $\Sigma_2^{\mathrm{hi}} = \{\alpha, \beta, \gamma\}$ with the corresponding projections $p_i^{\mathrm{dec}} : \Sigma_i^* \to (\Sigma_i^{\mathrm{hi}})^*$ for $i = 1, 2$. According to Proposition 3.1, the overall high-level control system can be computed by composing the decentralized high-level systems. It holds that $\mathcal{L}(H_1^{\mathrm{hi}}) = p_1^{\mathrm{hi}}(\mathcal{L}(H_1)) = \overline{\alpha\beta} = \underline{\mathcal{L}_{\mathrm{m}}(H_1^{\mathrm{hi}})} = p_1^{\mathrm{hi}}(\mathcal{L}_{\mathrm{m}}(H_1))$. Using the same method, $\mathcal{L}(H_2^{\mathrm{hi}}) = \overline{\alpha(\epsilon + \gamma)\beta}$ and $\mathcal{L}_{\mathrm{m}}(H_2^{\mathrm{hi}}) = \alpha(\epsilon + \beta + \gamma)$. Composing the decentralized high-level systems results in $H^{\mathrm{hi}} = H_1^{\mathrm{hi}} \| H_2^{\mathrm{hi}}$ with $\mathcal{L}(H^{\mathrm{hi}}) = \overline{\alpha(\epsilon + \gamma)\beta}$ and $\mathcal{L}_{\mathrm{m}}(H^{\mathrm{hi}}) = \alpha(\epsilon + \beta + \gamma)$.

The computational effort of the abstraction method will be discussed. The first step involves the computation of the decentralized high-level control systems $H_i^{\mathrm{hi}} = p_i^{\mathrm{dec}}(H_i)$, $i = 1, \ldots, n$ using the natural projection p_i^{dec}. The worst case complexity of the natural projection is exponential in the number of states of an automaton for the control system H_i. However, we show in the next section that the natural projection of the control systems suitable for our approach can be evaluated in polynomial time and the representation of H_i^{hi} has fewer states than the representation of H_i. Hence, while the computation of $H^{\mathrm{hi}} = \|_{i=1}^n H_i^{\mathrm{hi}}$ still has exponential complexity in the number n of components, it involves much smaller components than the computation of the detailed low-level model $H = \|_{i=1}^n H_i$.

IV. Nonblocking Hierarchical Control

In the previous section, we elaborated a method to efficiently compute the high-level model of a decentralized control system which captures the shared behavior of the decentralized subsystems. This model can now be used to implement an overall specification $E^{\mathrm{hi}} \subseteq (\Sigma^{\mathrm{hi}})^*$ by a high-level supervisor performing monolithic supervisor synthesis. The resulting nonblocking high-level supervisor S^{hi} such that $\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}}) = \kappa_{H^{\mathrm{hi}}, \Sigma_{\mathrm{uc}}^{\mathrm{hi}}}(E^{\mathrm{hi}})$ is depicted in Fig. 2(a).

In the next step, the virtual high-level control by S^{hi} has to be translated to a low-level supervisor S^{lo}, where S^{lo} is required to be nonblocking and *hierarchically consistent*, i.e., compliant with the behavior imposed by the high-level control. An important implication of the latter condition is that the

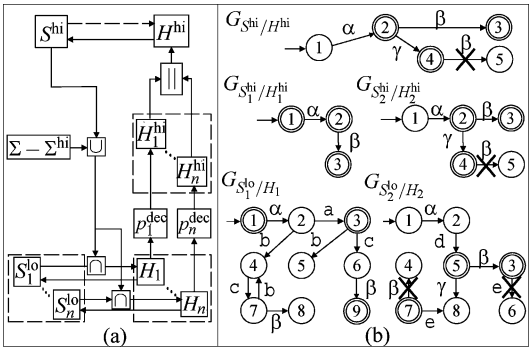


Fig. 3. (a) Hierarchical and decentralized architecture; (b) Example of the decentralized supervisor implementation.

low-level closed-loop behavior is nonempty whenever the high-level closed-loop behavior is nonempty. The hierarchical control problem is formalized in Definition 4.1.

Definition 4.1 (Hierarchical Control Problem): Given a PDCS $(\{H_i\}_{i\leq n}, p^{\mathrm{hi}}, \{H_i^{\mathrm{hi}}\}_{i\leq n})$ with its high-level abstraction $H^{\mathrm{hi}} = \|_{i=1}^{n} H_i^{\mathrm{hi}}$ and a nonblocking high-level supervisor S^{hi}, compute a nonblocking and hierarchically consistent low-level supervisor S^{lo}, i.e.,

$$p^{\mathrm{hi}}\left(\mathcal{L}\left(S^{\mathrm{lo}}/H\right)\right) = \mathcal{L}\left(S^{\mathrm{hi}}/H^{\mathrm{hi}}\right) \quad \text{(hierarchical consistency)}$$
$$\overline{\mathcal{L}_{\mathrm{m}}\left(S^{\mathrm{lo}}/H\right)} = \mathcal{L}\left(S^{\mathrm{lo}}/H\right) \quad \text{(nonblocking control)}.$$

A. Hierarchical and Decentralized Control Architecture

Analogous to the hierarchical abstraction procedure, it is not feasible to perform computations on the overall low-level system for the low-level supervisor implementation. Consequently, we also want to retain the decentralized system structure in the low-level supervisor. To this end, we refine the architecture of Fig. 2(a) and suggest the joint operation of the high-level supervisor S^{hi} and n decentralized low-level supervisors S_i^{lo}, $i = 1, \ldots n$ as illustrated in Fig. 3(a).

In this work, the low-level supervisors S_i^{lo} are computed based on specifications $K_i \subseteq \Sigma_i^*$ that incorporate the behavior of the high-level closed loop $S^{\mathrm{hi}}/H^{\mathrm{hi}}$ as seen from the respective subsystem H_i and that exhibit further desirable properties as formulated in Definition 4.2. An explicit derivation of such specifications is given in Section IV-D.

Definition 4.2 (Decentralized Specification): Let $(\{H_i\}_{i\leq n}, p^{\mathrm{hi}}, \{H_i^{\mathrm{hi}}\}_{i\leq n})$ be a PDCS with its high-level abstraction $H^{\mathrm{hi}} = \|_{i=1}^{n} H_i^{\mathrm{hi}}$, and let $p_i^{\mathrm{hi}} : (\Sigma^{\mathrm{hi}})^* \to (\Sigma_i^{\mathrm{hi}})^*$, $i = 1, \ldots, n$ be natural projections. Given a nonblocking high-level supervisor S^{hi}, the decentralized low-level specification $K_i \subseteq \Sigma_i^*$ is defined such that $p_i^{\mathrm{dec}}(K_i) \supseteq \left(p_i^{\mathrm{hi}}(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))\right)$ and K_i is controllable w.r.t H_i and $\Sigma_{i,\mathrm{uc}}$.

Combining the overall high-level supervisor S^{hi} and the decentralized low-level supervisors S_i^{lo}, we define the overall low-level supervisor $S^{\mathrm{lo}} : \mathcal{L}(H) \to \Gamma$ such that $S^{\mathrm{lo}}/H = S^{\mathrm{hi}}/H^{\mathrm{hi}} \| (\|_{i=1}^{n} S_i^{\mathrm{lo}}/H_i)$. The hierarchical and decentralized architecture is summarized as a *hierarchical and decentralized closed-loop system* in the following definition.

Definition 4.3 (HDCLS [18]): A Hierarchical and Decentralized Closed-Loop System (HDCLS) $(\{H_i\}_{i\leq n},\ p^{\mathrm{hi}}, \{H_i^{\mathrm{hi}}\}_{i\leq n}, S^{\mathrm{hi}}, S^{\mathrm{lo}})$ consists of the following entities:

i) a PDCS $(\{H_i\}_{i\leq n}, p^{\mathrm{hi}}, \{H_i^{\mathrm{hi}}\}_{i\leq n})$ as in Definition 3.2,
ii) a high-level supervisor S^{hi}: $\mathcal{L}(H^{\mathrm{hi}}) \to \Gamma^{\mathrm{hi}}$ with the high-level closed-loop control system $S^{\mathrm{hi}}/H^{\mathrm{hi}}$,
iii) decentralized low-level supervisors $S_i^{\mathrm{lo}} : \mathcal{L}(H_i) \to \Gamma_i$ such that $\mathcal{L}(S_i^{\mathrm{lo}}/H_i) = \overline{K_i}$, where K_i is chosen according to Definition 4.2,
iv) the overall low-level supervisor S^{lo}: $\mathcal{L}(H) \to \Gamma$ with $S^{\mathrm{lo}}/H = S^{\mathrm{hi}}/H^{\mathrm{hi}} \| (\|_{i=1}^{n} S_i^{\mathrm{lo}}/H_i)$.

That is, the low-level model of a hierarchical and decentralized closed-loop system is a DCS which is abstracted by projecting to a superset of the shared events of its components. Because of the decentralized nature of the system, the controllability properties of the low-level events are directly transferred to the high-level in this approach, i.e., a high-level event is controllable if it is controllable in the low-level, and it is uncontrollable if it is uncontrollable in the low level.[2] Note that maximal permissiveness might be lost by this choice of the high-level controllability properties as opposed to e.g., [12], [13]. However, using our method, it is ensured that shared events in different subsystems agree on their controllability status. On the high level, standard supervisory control is applied, yielding the high-level supervisor. The translation of the high-level control action to each low level component H_i is performed based on the specification K_i that is designed to be controllable w.r.t. H_i and $\Sigma_{i,\mathrm{uc}}$.

The application of the hierarchical and decentralized architecture is shown in the following example.

Example 4.1: Let $(\{H_i\}_{i\leq 2}, p^{\mathrm{hi}}, \{H_i^{\mathrm{hi}}\}_{i\leq 2})$ be the PDCS in Example 3.1. The set of controllable events is $\Sigma_{\mathrm{c}} = \{\beta, \mathtt{b}, \mathtt{c}, \mathtt{e}\}$. We choose the high-level supervisor S^{hi} for the overall high-level system H^{hi} such that $S^{\mathrm{hi}}(\alpha\gamma) = \{\alpha, \gamma\}$ and $S^{\mathrm{hi}}(s^{\mathrm{hi}}) = \{\alpha, \beta, \gamma\}$ for all other $s^{\mathrm{hi}} \in \mathcal{L}(H^{\mathrm{hi}})$. The specifications $K_1 = \alpha a(\epsilon + c\beta)$ and $K_2 = \alpha\mathtt{d}(\epsilon + \beta + \gamma e)$ fulfill Definition 4.2. Thus, the low-level supervisors with $\mathcal{L}(S_i^{\mathrm{lo}}/H_i) = \overline{K_i}$, $i = 1, \ldots, 2$ evaluate to $S_1^{\mathrm{lo}}(s_1) = \Sigma_1$ for all $s_1 \in \mathcal{L}(H_1)$ and $S_2^{\mathrm{lo}}(s_2) = \{\mathtt{d}, \alpha, \gamma\}$ if $s_2 \in \{\alpha\mathtt{d}\beta, \alpha\mathtt{d}\gamma\mathtt{e}\}$ and $S_2^{\mathrm{lo}}(s_2) = \Sigma_2$ otherwise. The resulting low-level closed-loop behaviors are $S_1^{\mathrm{lo}}/H_1 = H_1$ and $\mathcal{L}(S_2^{\mathrm{lo}}/H_2) = \overline{\alpha\mathtt{d}(\beta + \gamma\mathtt{e})}$, $\mathcal{L}_{\mathrm{m}}(S_2^{\mathrm{lo}}/H_2) = \alpha\mathtt{d}(\epsilon + \beta + \gamma\mathtt{e})$.

Having introduced our hierarchical control architecture, we now identify sufficient conditions such that the HDCLS solves the control problem in Definition 4.1, i.e., the low-level supervisor is hierarchically consistent and nonblocking.

B. Hierarchical Consistency

First, it is proved that the HDCLS in Definition 4.3 is already hierarchically consistent. In this context, it is interesting to observe that this result is valid for all specifications K_i that fulfill the conditions in Definition 4.2.

Proposition 4.1 (Hierarchical Consistency): Let $(\{H_i\}_{i\leq n},\ p^{\mathrm{hi}}, \{H_i^{\mathrm{hi}}\}_{i\leq n})$ be a PDCS with a high-level supervisor S^{hi} and the high-level closed-loop system

[2]This assignment is called control delay freedom, e.g., in [27].

$S^{\mathrm{hi}}/H^{\mathrm{hi}}$. If S^{lo} is given as in Definition 4.3, then $(\{H_i\}_{i\leq n}, p^{\mathrm{hi}}, \{H_i^{\mathrm{hi}}\}_{i\leq n}, S^{\mathrm{hi}}, S^{\mathrm{lo}})$ is a hierarchically consistent HDCLS.

Proof: Because of Definition 4.2, K_i is controllable w.r.t. H_i and $\Sigma_{i,\mathrm{uc}}$, and $p_i^{\mathrm{dec}}(K_i) \supseteq p_i^{\mathrm{hi}}(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))$ for all $i = 1,\ldots,n$. Thus, for all i, there exists a $S_i^{\mathrm{lo}} : (\Sigma_i)^* \to \Gamma_i$ s.t. $\mathcal{L}(S_i^{\mathrm{lo}}/H_i) = \overline{K_i}$.

Implementing the overall low-level supervisor S^{lo} as in Definition 4.2 results in

$$\begin{aligned} p^{\mathrm{hi}}\left(\mathcal{L}\left(S^{\mathrm{lo}}/H\right)\right) &= p^{\mathrm{hi}}\left(\mathcal{L}\left(S^{\mathrm{hi}}/H^{\mathrm{hi}}\right) \parallel \left(\|_{i=1}^{n}\mathcal{L}\left(S_i^{\mathrm{lo}}/H_i\right)\right)\right) \\ &= \mathcal{L}\left(S^{\mathrm{hi}}/H^{\mathrm{hi}}\right) \| p^{\mathrm{hi}}\left(\|_{i=1}^{n}\mathcal{L}\left(S_i^{\mathrm{lo}}/H_i\right)\right) \end{aligned}$$

because of Lemma 3.1. Also with Lemma 3.1, it is true that

$$\begin{aligned} p^{\mathrm{hi}}\left(\|_{i=1}^{n}\mathcal{L}\left(S_i^{\mathrm{lo}}/H_i\right)\right) &= \|_{i=1}^{n} p_i^{\mathrm{dec}}\left(\mathcal{L}\left(S_i^{\mathrm{lo}}/H_i\right)\right) \\ &= \|_{i=1}^{n} p_i^{\mathrm{dec}}(\overline{K_i}) \supseteq \|_{i=1}^{n} p_i^{\mathrm{hi}}\left(\mathcal{L}\left(S^{\mathrm{hi}}/H^{\mathrm{hi}}\right)\right). \end{aligned}$$

Noting that $\|_{i=1}^{n} p_i^{\mathrm{hi}}\left(\mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})\right) \supseteq \mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$, it holds that $p^{\mathrm{hi}}\left(\mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})\|(\|_{i=1}^{n}\mathcal{L}(S_i^{\mathrm{lo}}/H_i))\right) = \mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$. Consequently, the hierarchical and decentralized control system is hierarchically consistent. ∎

1) Example 4.2: The HDCLS in Example 4.1 is hierarchically consistent. There is a corresponding string in the low-level closed-loop behavior for all strings in the high-level closed-loop behavior.

In summary, the HDCLS provides hierarchical consistency without further requirements on the system behavior.

C. Nonblocking Control

Although the hierarchical and decentralized closed-loop system guarantees hierarchically consistent control, the closed-loop system is in general not nonblocking. It is possible that while the desired high-level behavior can be achieved by low-level control, there are local paths which lead to deadlock or livelock situations as demonstrated in the following example.

Example 4.3: Consider the HDCLS in Example 4.1. The string $\alpha\mathtt{b} \in \mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})\|\left(\mathcal{L}(S_1^{\mathrm{lo}}/H_1)\|\mathcal{L}(S_2^{\mathrm{lo}}/H_2)\right)$ cannot be extended to a marked string in $\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}})\|\left(\mathcal{L}_{\mathrm{m}}(S_1^{\mathrm{lo}}/H_1)\|\mathcal{L}_{\mathrm{m}}(S_2^{\mathrm{lo}}/H_2)\right)$. Thus the overall system is blocking.

In the framework of hierarchical and decentralized closed-loop systems as introduced in Definition 4.3, there are four issues which can lead to a blocking low-level closed loop. These issues are illustrated on the basis of Fig. 3(b).

1) the high level considers the string $\alpha \in \mathcal{L}_{\mathrm{m}}(S_1^{\mathrm{hi}}/H_1^{\mathrm{hi}})$ as marked although some corresponding low-level strings have a marked predecessor string and some do not. For example $\alpha\mathtt{ac} \in \mathcal{L}(S_1^{\mathrm{lo}}/H_1)$ has a marked predecessor ($\alpha\mathtt{a}$) and $\alpha\mathtt{b} \in \mathcal{L}(S_1^{\mathrm{lo}}/H_1)$ does not.
2) the high level assumes that the event β can always be generated after the occurrence of $\alpha \in \mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$, but this is not possible after the low-level string $\alpha\mathtt{ab} \in \mathcal{L}(H_1)$.
3) the high level terminates its action after the marked string $\alpha\beta \in \mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$ but the low level can have a nonmarked extension $\alpha\mathtt{d}\beta\mathtt{e} \in \mathcal{L}(H_2)$.
4) the overall high-level supervisor allows the string $\alpha\beta$ but forbids $\alpha\gamma\beta$. This means that on the one hand $\alpha\beta = p_1^{\mathrm{hi}}(\alpha\beta) \in K_1$, i.e., all strings $s \in \mathcal{L}(H_1)$ with $p_1^{\mathrm{dec}}(s) = \alpha\beta$ are enabled, and on the other hand it can happen that β is not possible after the string $\alpha \in \mathcal{L}(H_1)$ if $\alpha\gamma$ occurred in H_2. In this case it would be favorable to disable $\mathtt{b}$ and $\mathtt{c}$ after the occurrence of $\alpha\mathtt{a}$ in H_1 to avoid blocking, which is not possible as it cannot be distinguished if β will happen or not.

It is interesting to note that the first three blocking issues from above only affect each individual projected control system $(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$ of a HDCLS. Because of this reason, we investigate three sufficient conditions for nonblocking hierarchical control of the projected control systems $(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$: *marked string acceptance*, *locally nonblocking condition* and *marked string controllability*.

1) Marked String Acceptance: One reason why nonblocking control fails in Example 4.3 is that not all local strings corresponding to marked high-level strings are also marked. A solution to this problem is the requirement that if the high-level observes a string in $\mathcal{L}_{\mathrm{m}}(H_i^{\mathrm{hi}})$, the low-level also has to pass a string in $\mathcal{L}_{\mathrm{m}}(H_i)$. This means if a high-level string s^{hi} is contained in the language $\mathcal{L}_{\mathrm{m}}(H_i^{\mathrm{hi}})$, then it must be guaranteed that any low-level string which is projected to s^{hi} and which has a high-level successor event, must have a prefix in $\mathcal{L}_{\mathrm{m}}(H_i)$ and with the same projection s^{hi}. This property is denoted *marked string acceptance*. It is based on the set of *exit strings* which contains all strings with a high-level successor event.

Definition 4.4 (Exit Strings): Let $(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$ be a PCS, and assume $s^{\mathrm{hi}} \in \mathcal{L}(H_i^{\mathrm{hi}})$. The set of exit strings of s^{hi} is

$$L_{i,\mathrm{ex}}(s^{\mathrm{hi}}) := \begin{array}{l}\{s \in \mathcal{L}(H_i) | p_i^{\mathrm{dec}}(s) = s^{\mathrm{hi}} \wedge \\ \quad (\exists \sigma^{\mathrm{hi}} \in \Sigma_i^{\mathrm{hi}} \text{ s.t. } s\sigma^{\mathrm{hi}} \in \mathcal{L}(H_i))\}\end{array} \subseteq \Sigma_i^*.$$

Definition 4.5 (Marked String Acceptance): Let $(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$ be a PCS. The string $s^{\mathrm{hi}} \in \mathcal{L}_{\mathrm{m}}(H_i^{\mathrm{hi}})$ is marked string accepting[3] if for all $s_{\mathrm{ex}} \in L_{i,\mathrm{ex}}(s^{\mathrm{hi}})$

$$\exists s' \leq s_{\mathrm{ex}} \text{ with } p_i^{\mathrm{dec}}(s') = s^{\mathrm{hi}} \text{ and } s' \in \mathcal{L}_{\mathrm{m}}(H_i).$$

$(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$ is marked string accepting if s^{hi} is marked string accepting for all $s^{\mathrm{hi}} \in \mathcal{L}_{\mathrm{m}}(H_i^{\mathrm{hi}})$.

Example 4.4: The hierarchical closed-loop system in Example 4.3 is not marked string accepting. The string $\alpha\mathtt{bc} \in \mathcal{L}(H_1)$ is an exit string in $L_{1,\mathrm{ex}}(\alpha)$ for the marked high-level string $\alpha \in \mathcal{L}_{\mathrm{m}}(H_1^{\mathrm{hi}})$, but there is no string $s' \in \mathcal{L}_{\mathrm{m}}(H_1)$ s.t. $s' \leq \alpha\mathtt{bc}$ and $p_1^{\mathrm{dec}}(s') = \alpha$.

2) Locally Nonblocking Condition: The second issue in Example 4.3 originates from the construction of the low-level supervisor and the possible interaction with other subsystems. The supervisor is based on the assumption that after a low-level string, all high-level events which are feasible in the corresponding high-level string can be generated. Furthermore, other decentralized subsystems only perceive the shared behavior, and thus assume that if a high-level event is feasible in the high level, there is also a low-level path such that the event can be executed. This is generally not the case and can lead to blocking behavior.

A string s is denoted *locally nonblocking* if for all high-level events which are feasible after the corresponding high-level

[3]Note that $s^{\mathrm{hi}} \in \mathcal{L}(H_i^{\mathrm{hi}}) - \mathcal{L}_{\mathrm{m}}(H_i^{\mathrm{hi}}) \Rightarrow (p^{\mathrm{hi}})^{-1}(s^{\mathrm{hi}}) \cap \mathcal{L}_{\mathrm{m}}(H_i) = \emptyset$.

string $p^{\mathrm{hi}}(s)$, a local path starting from s exists, such that the high-level event can occur.

Definition 4.6 (Locally Nonblocking): Let $(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$ be a PCS. The string $s^{\mathrm{hi}} \in \mathcal{L}(H_i^{\mathrm{hi}})$ is locally nonblocking if for all $s \in \mathcal{L}(H_i)$ with $p^{\mathrm{hi}}(s) = s^{\mathrm{hi}}$ and $\forall \sigma \in \Sigma_{\mathcal{L}(H_i^{\mathrm{hi}})}(s^{\mathrm{hi}})$, $\exists u_\sigma \in (\Sigma_i - \Sigma_i^{\mathrm{hi}})^*$ s.t. $su_\sigma\sigma \in \mathcal{L}(H_i)$. $(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$ is locally nonblocking if s^{hi} is locally nonblocking $\forall s^{\mathrm{hi}} \in \mathcal{L}(H_i^{\mathrm{hi}})$.

Remark 4.1: Our condition is equivalent to the observer property for prefix-closed languages as in [11], referring to the natural projection as the causal reporter map.

Example 4.5: The PCS $(H_1, p_1^{\mathrm{dec}}, H_1^{\mathrm{hi}})$ in Example 4.3 is not locally nonblocking. Although β is feasible after the high-level string $\alpha \in \mathcal{L}(H_1^{\mathrm{hi}})$, there is no local path after the corresponding low-level string $\alpha\mathtt{ab} \in \mathcal{L}(H_1)$ to generate β.

It is interesting to note that the locally nonblocking condition together with marked string acceptance imply that automata representations of the high-level models H_i^{hi} in our architecture are smaller than the corresponding low-level models H_i. It is shown in [19] that both conditions establish an equivalence relation on the state space of G_{H_i} such that the equivalence classes correspond to the states of $G_{H_i^{\mathrm{hi}}}$. Using this information and the properties of the synchronous composition, it can also be concluded that an automata representation of H^{hi} cannot have more states than the low-level model H. Furthermore, algorithms for the verification of the locally nonblocking condition and marked string acceptance are presented in [18]. Both algorithms employ a local reachability computation on G_{H_i} that in the worst case has to be carried out for all combinations of states in $G_{H_i^{\mathrm{hi}}}$ and G_{H_i}. Bounding the state count of G_{H_i} and $G_{H_i^{\mathrm{hi}}}$ with N_i and N_i^{hi}, respectively, the time complexity evaluates to $\mathcal{O}(N_i^{\mathrm{hi}} N_i^2)$.

3) Marked String Controllability: We first give definitions of *entry strings* and *local languages*. The set of *entry strings*[4] contains the shortest possible low-level strings which are projected to a given high-level string.

Definition 4.7 (Entry Strings [12]): Let $(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$ be a PCS and $s^{\mathrm{hi}} \in \mathcal{L}(H_i^{\mathrm{hi}})$. The set of entry strings of s^{hi} is

$$L_{i,\mathrm{en}}(s^{\mathrm{hi}}) := \begin{array}{l}\{s \in \mathcal{L}(H_i) | p_i^{\mathrm{dec}}(s) = s^{\mathrm{hi}} \wedge \\ \quad \nexists s' < s \text{ s.t. } p_i^{\mathrm{dec}}(s') = s^{\mathrm{hi}}\}\end{array} \subseteq \Sigma_i^*.$$

The local behavior after strings $s \in \mathcal{L}(H_i)$ is described next. It represents the possible local behavior after the observation of a high-level event. For any string $s \in \mathcal{L}(H_i)$, the continuation of s with all local strings $u \in (\Sigma_i - \Sigma_i^{\mathrm{hi}})^*$ in $\mathcal{L}(H_i)$ is a prefix-closed language. A second (not necessarily prefix-closed) language contains local continuations of s in $\mathcal{L}_{\mathrm{m}}(H_i)$.

Definition 4.8 (Local Languages [18]): Let H_i be a low-level system and $\Sigma_i^{\mathrm{hi}} \subseteq \Sigma_i$ an alphabet. Also let $s \in \mathcal{L}(H_i)$. The local prefix-closed language $L_{i,\mathrm{pre}}(s)$ is

$$L_{i,\mathrm{pre}}(s) := \{u \in (\Sigma_i - \Sigma_i^{\mathrm{hi}})^* | su \in \mathcal{L}(H_i)\} \subseteq (\Sigma_i - \Sigma_i^{\mathrm{hi}})^*$$

and the local marked language $L_{i,\mathrm{m}}(s)$ is

$$L_{i,\mathrm{m}}(s) := \{u \in (\Sigma_i - \Sigma_i^{\mathrm{hi}})^* | su \in \mathcal{L}_{\mathrm{m}}(H_i)\} \subseteq (\Sigma_i - \Sigma_i^{\mathrm{hi}})^*.$$

The local system is defined as $H_i(s) := (L_{i,\mathrm{pre}}(s), L_{i,\mathrm{m}}(s))$.

[4]*Entry strings* are also called *vocal strings* or *relevant strings*.

The third problem in Example 4.3 deals with low-level strings in a component H_i that correspond to high-level strings without any successor events for this component in the high-level closed-loop behavior $S^{\mathrm{hi}}/H^{\mathrm{hi}}$. In this case, the nonblocking low-level supervisor S_i^{lo} must take care that the possible future local behavior does not cause blocking in the low level, as no more changes in the high-level can happen.

This is achieved by investigating all entry strings $s_{\mathrm{en}} \in L_{i,\mathrm{en}}(s^{\mathrm{hi}})$ of such high-level string s^{hi} and computing the supremal controllable and nonblocking language starting from the entry strings. A nonblocking supervisor can only be implemented if this supremal controllable and nonblocking sublanguage is nonempty. This is guaranteed if the *marked string controllability* condition is fulfilled.

Definition 4.9 (Marked String Controllability): Let $(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$ be a PCS. The string $s^{\mathrm{hi}} \in \mathcal{L}_{\mathrm{m}}(H_i^{\mathrm{hi}})$ is marked string controllable if $\kappa_{H_i(s_{\mathrm{en}}),\Sigma_{i,\mathrm{uc}}}(L_{i,\mathrm{m}}(s_{\mathrm{en}})) \neq \emptyset$ for all $s_{\mathrm{en}} \in L_{i,\mathrm{en}}(s^{\mathrm{hi}})$. $(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$ is marked string controllable if s^{hi} is marked string controllable for all $s^{\mathrm{hi}} \in \mathcal{L}_{\mathrm{m}}(H_i^{\mathrm{hi}})$.

Given the automaton G_{H_i} for H_i, the verification of marked string controllability requires the construction of a local automaton for each marked high-level state in $G_{H_i^{\mathrm{hi}}}$ and the computation of a supremal controllable sublanguage. This can be done with a complexity of $\mathcal{O}(N_i^{\mathrm{hi}} N_i^3)$.

As S_i^{lo} is computed for the specification K_i according to Definition 4.3, the information whether further high-level events can happen after strings in H_i has to be incorporated in K_i. Then, marked string controllability ensures that K_i is controllable w.r.t. H_i and $\Sigma_{i,\mathrm{uc}}$, and no blocking behavior can occur locally. Given a high-level closed loop $S^{\mathrm{hi}}/H^{\mathrm{hi}}$, a specification that could be used is $K_i = p_i^{\mathrm{hi}}(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))$. Unfortunately, the worst case complexity of the natural projection p_i^{hi} is exponential. In Lemma 4.1, we provide an appropriate specification $K_i \supseteq p_i^{\mathrm{hi}}(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))$ that can be constructed in polynomial time, and show that it is controllable w.r.t. H_i and $\Sigma_{i,\mathrm{uc}}$, i.e., K_i fulfills the conditions in Definition 4.2.

Lemma 4.1 (Decentralized Specification): Let $(\{H_i\}_{i\leq n}, p^{\mathrm{hi}}, \{H_i^{\mathrm{hi}}\}_{i\leq n})$ be a PDCS with its high-level abstraction $H^{\mathrm{hi}} = \|_{i=1}^n H_i^{\mathrm{hi}}$, and let $p_i^{\mathrm{hi}} : (\Sigma^{\mathrm{hi}})^* \to (\Sigma_i^{\mathrm{hi}})^*$, $i = 1, \ldots, n$ be natural projections. Given a nonblocking high-level supervisor S^{hi}, let $\tilde{K}_i \subseteq \Sigma_i^*$ be defined such that for all $s^{\mathrm{hi}} = \sigma_1\sigma_2\ldots\sigma_m$ with $\sigma_i \in \Sigma_i^{\mathrm{hi}}$, $i = 1, \ldots, m$ and $s \in (\Sigma_i - \Sigma_{p_i^{\mathrm{hi}}(\mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))}(\epsilon))^* \cdot \sigma_1(\Sigma_i - \Sigma_{p_i^{\mathrm{hi}}(\mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))}(\sigma_1))^*\sigma_2 \ldots \sigma_m(\Sigma_i - \Sigma_i^{\mathrm{hi}})^*$

$$s^{\mathrm{hi}} \in p_i^{\mathrm{hi}}\left(\mathcal{L}_{\mathrm{m}}\left(S^{\mathrm{hi}}/H^{\mathrm{hi}}\right)\right) \Leftrightarrow s \in \tilde{K}_i. \tag{1}$$

Define $K_i := \kappa_{H_i,\Sigma_{i,\mathrm{uc}}}(\tilde{K}_i)$. Then $p_i^{\mathrm{dec}}(K_i) \supseteq p_i^{\mathrm{hi}}(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))$, and K_i is controllable w.r.t. H_i and $\Sigma_{i,\mathrm{uc}}$ for all $i = 1, \ldots, n$, if $(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$ is marked string controllable.

The proof of Lemma 4.1 relies on Lemma 4.2. It states that if the decentralized projection of an entry string in $\mathcal{L}(H_i)$ is contained in $p_i^{\mathrm{hi}}(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))$, then this entry string is also contained in K_i. Appendix A provides a proof of Lemma 4.2.

Lemma 4.2: Given the notation as in Lemma 4.1, let $s^{\mathrm{hi}} \in p_i^{\mathrm{hi}}(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))$ and $s \in L_{i,\mathrm{en}}(s^{\mathrm{hi}})$. Then $s \in K_i$.

Now Lemma 4.1 can be proved.

Proof: First, we show that $p_i^{\mathrm{hi}}\big(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}})\big) \subseteq p_i^{\mathrm{dec}}(K_i)$, i.e., for each $s^{\mathrm{hi}} \in p_i^{\mathrm{hi}}\big(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}})\big)$ there exists a $s \in K_i$ s.t. $p_i^{\mathrm{dec}}(s) = s^{\mathrm{hi}}$. Let $s^{\mathrm{hi}} \in p_i^{\mathrm{hi}}\big(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}})\big)$. Then there is a $s \in L_{i,\mathrm{en}}(s^{\mathrm{hi}})$ s.t. $p_i^{\mathrm{dec}}(s) = s^{\mathrm{hi}}$ as $p_i^{\mathrm{hi}}\big(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}})\big) \subseteq p_i^{\mathrm{dec}}(\mathcal{L}(H_i))$. Because of Lemma 4.2, $s \in K_i$ and $p_i^{\mathrm{dec}}(s) = s^{\mathrm{hi}}$. As $s^{\mathrm{hi}} \in p_i^{\mathrm{hi}}\big(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}})\big)$ was chosen arbitrarily, we have that $p_i^{\mathrm{hi}}\big(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}})\big) \subseteq p_i^{\mathrm{dec}}(K_i)$. Finally, K_i is controllable w.r.t. H_i and $\Sigma_{i,\mathrm{uc}}$ by definition. ∎

In the light of Lemma 4.1, the specification K_i fulfills the conditions in Definition 4.2 and captures the information whether further high-level events are allowed to happen after strings in H_i. Furthermore, the construction of an automata representation of $\tilde{K}_i$ can be formulated as a *relational coarsest partition problem* as in [28] (see Example 4.6). Denoting N^{hi} the state count of $G_{S^{\mathrm{hi}}/H^{\mathrm{hi}}}$, the result can be computed with a complexity of $\mathcal{O}((N^{\mathrm{hi}})^3 \log N^{\mathrm{hi}})$ as the quotient automaton for an equivalence relation on the state space of $G_{S^{\mathrm{hi}}/H^{\mathrm{hi}}}$. Thus, the automata representation of $\tilde{K}_i$ has at most N^{hi} states.

4) Ambiguous Extensions: Finally, the fourth issue in Example 4.3 that involves the high-level coordination of the decentralized components has to be addressed. To this end, we introduce a further condition which requires that there are no *ambiguous extensions* of strings in the high-level closed-loop system w.r.t. the decentralized high-level alphabets.

Definition 4.10 (Ambiguous Extensions): Let $L \subseteq \Sigma^*$ be a regular language, $\Sigma' \subseteq \Sigma$ and $p : \Sigma^* \to (\Sigma')^*$ the natural projection. The string $u \in \Sigma^*\Sigma'$ is an ambiguous extension of the string $s \in L$ w.r.t. Σ' if $su \in L$ and there exists a string $s' \in L$ such that $p(s) = p(s')$ and there is no $u' \in \Sigma^*\Sigma'$ with $s'u' \in L$.

This means that the decentralized low-level supervisors of the HDCLS must be sure that either further strings are possible or no further strings can occur due to the overall high-level supervisor. In our approach it is required that the high-level closed-loop language $\mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$ does not have ambiguous extensions w.r.t. all alphabets Σ_i^{hi}. This property can be verified with complexity $\mathcal{O}((N^{\mathrm{hi}})^3 \log N^{\mathrm{hi}})$ using the quotient automaton that is constructed for the specification $\tilde{K}_i$. It can be shown that $\mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$ does not have ambiguous extensions w.r.t. Σ_i^{hi} iff the constructed quotient automaton does not contain unobservable transitions.

The following example illustrates the computation of K_i and the test for the existence of ambiguous extensions.

Example 4.6: Fig. 4 shows an automaton representation of the high-level supervisor in Example 4.1 (a). The specification $\tilde{K}_1$ is constructed by adding selfloops according to Lemma 4.1, and by replacing transitions with events in $\Sigma^{\mathrm{hi}} - \Sigma_1^{\mathrm{hi}}$ with ϵ-transitions. The quotient automaton $G_{(\overline{\tilde{K}_1},\tilde{K}_1)}$ corresponding to the relational coarsest partition problem for the construction of $\tilde{K}_1$ is depicted in Fig. 4(b), where the Nerode equivalence w.r.t. $\tilde{K}_1$[24] is employed to identify equivalent states (equivalence classes are characterized by shaded areas). $G_{(\overline{\tilde{K}_1},\tilde{K}_1)}$ contains an unobservable (ϵ-) transition between equivalence classes, i.e., there is an ambiguous extension due to the possible occurrence of γ as described in blocking issue 4). Note that K_1 in Example 4.1 evaluates to $K_1 = \kappa_{H_i,\Sigma_{i,\mathrm{uc}}}(\tilde{K}_1)$.

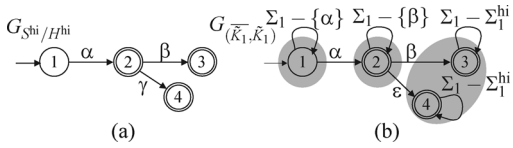


Fig. 4. Construction of the specification language K_i.

With the additional requirement of absence of ambiguous extensions, the HDCLS in Definition 4.3 solves the control problem in Definition 4.1.

Theorem 4.1 (Main Result): Let $(\{H_i\}_{i\le n}, p^{\mathrm{hi}}, \{H_i^{\mathrm{hi}}\}_{i\le n}, S^{\mathrm{hi}}, S^{\mathrm{lo}})$ be a HDCLS with the specifications $\{K_i\}_{i\le n}$ in Lemma 4.1. Assume that, for $i = 1,\ldots,n$, no string $s \in \mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$ has ambiguous extensions w.r.t. Σ_i^{hi} and all PCSs $(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$ are marked string accepting, locally nonblocking and marked string controllable. Then $(\{H_i\}_{i\le n}, p^{\mathrm{hi}}, \{H_i^{\mathrm{hi}}\}_{i\le n}, S^{\mathrm{hi}}, S^{\mathrm{lo}})$ is nonblocking and hierarchically consistent.

Lemma 4.3 supports the proof of Theorem 4.1. It states that if a string in the high-level closed-loop system can be extended to a marked string, then any corresponding string in a decentralized low-level system can be extended to a corresponding marked string. Lemma 4.3 is proved in Appendix B.

Lemma 4.3: Let $(\{H_i\}_{i\le n}, p^{\mathrm{hi}}, \{H_i^{\mathrm{hi}}\}_{i\le n}, S^{\mathrm{hi}}, S^{\mathrm{lo}})$ be a HDCLS with the specifications $\{K_i\}_{i\le n}$ in Lemma 4.1. Assume that all PCSs $(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$, $i = 1,\ldots,n$ are marked string accepting, locally nonblocking and marked string controllable. Let $s_i \in \mathcal{L}(S_i^{\mathrm{lo}}/H_i)$ and $s^{\mathrm{hi}} \in \mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$ s.t. $p_i^{\mathrm{dec}}(s_i) = p_i^{\mathrm{hi}}(s^{\mathrm{hi}})$. If $t \in (\Sigma^{\mathrm{hi}})^*$ s.t. $s^{\mathrm{hi}}t \in \mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$ and $p_i^{\mathrm{hi}}(t) \neq \epsilon$, then there exists a $u_i \in \Sigma_i^*$ s.t. $s_iu_i \in \mathcal{L}_{\mathrm{m}}(S_i^{\mathrm{lo}}/H_i)$ and $p_i^{\mathrm{dec}}(s_iu_i) = p_i^{\mathrm{hi}}(s^{\mathrm{hi}}t)$.

With this result, Theorem 4.1 can be proved.

Proof: Proposition 4.1 implies hierarchical consistency.

For proving nonblocking behavior, we show that $s \in \overline{\mathcal{L}_{\mathrm{m}}(S_i^{\mathrm{lo}}/H)}$ for any $s \in \mathcal{L}(S_i^{\mathrm{lo}}/H)$ and $s^{\mathrm{hi}} = p^{\mathrm{hi}}(s) \in \mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$. Note that $\mathcal{L}(S_i^{\mathrm{lo}}/H) \neq \emptyset$ as $\mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}}) \neq \emptyset$ and $p^{\mathrm{hi}}(\mathcal{L}(S_i^{\mathrm{lo}}/H)) = \mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$.

Because of Definition 4.3, $s \in \mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})||\big(||_{i=1}^n \mathcal{L}(S_i^{\mathrm{lo}}/H_i)\big)$. Thus $s_i := p_i(s) \in \mathcal{L}(S_i^{\mathrm{lo}}/H_i)$ and $s_i^{\mathrm{hi}} := p_i^{\mathrm{dec}}(s_i) \in p_i^{\mathrm{hi}}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$. Let

$$\mathcal{I} := \Big\{ i | 1 \le i \le n \wedge \exists u_i \in (\Sigma_i - \Sigma^{\mathrm{hi}})^*, \sigma_i \in \Sigma_i^{\mathrm{hi}} \text{ s.t. } s_iu_i\sigma_i \in \mathcal{L}\left(S_i^{\mathrm{lo}}/H_i\right) \Big\}.$$

The following algorithm is performed to find an appropriate string leading to a marked string in the high-level.

1. $k = 1$, $t_0 = \epsilon$, $\tilde{\mathcal{I}} = \mathcal{I}$.

2. choose $i_k \in \tilde{\mathcal{I}}$.

3. find $t_k \in (\Sigma^{\mathrm{hi}})^*$ s.t. $s^{\mathrm{hi}}t_0t_1\cdots t_k \in \mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$ and $p_{i_k}^{\mathrm{hi}}(t_k) \neq \epsilon$.

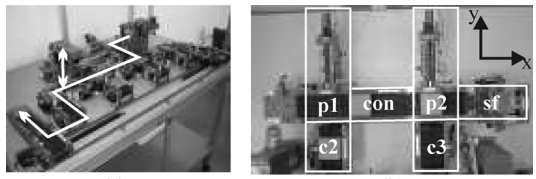
4. remove all j with $p_j^{\mathrm{hi}}(t_k) \neq \epsilon$ from $\tilde{\mathcal{I}}$.

5. **if** $\tilde{\mathcal{I}} = \emptyset$: set $k^* := k$ and **terminate**

else $k := k + 1$ and **go to** 2.

TABLE I
COMPUTATIONAL COMPLEXITY FOR THE ALGORITHMIC PROCEDURES

Condition	Complexity
marked string acceptance	$\mathcal{O}(N_i^{\mathrm{hi}} N_i^2)$
locally nonblocking condition	$\mathcal{O}(N_i^{\mathrm{hi}} N_i^2)$
marked string controllability	$\mathcal{O}(N_i^{\mathrm{hi}} N_i^3)$
construction of K_i	$\mathcal{O}((N^{\mathrm{hi}})^3 \log N^{\mathrm{hi}})$
ambiguous extensions	$\mathcal{O}((N^{\mathrm{hi}})^3 \log N^{\mathrm{hi}})$



(a) (b)

Fig. 5. (a) Fischertechnik simulation model. (b) Distribution system.

First note that the string t_k in 3. always exists and $\Sigma_{\mathrm{p}_i^{\mathrm{hi}}(\mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))}(s_i^{\mathrm{hi}}) \neq \emptyset$ for all $i_k \in \tilde{\mathcal{I}}$. Assume that $\Sigma_{\mathrm{p}_i^{\mathrm{hi}}(\mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))}(s_i^{\mathrm{hi}}) = \emptyset$ for some i_k. Then there is no $\sigma \in \Sigma_i^{\mathrm{hi}}$ s.t. $s_i^{\mathrm{hi}}\sigma \in p_i^{\mathrm{dec}}(\overline{K_i})$. But then, there is no $u_i \in (\Sigma_i - \Sigma_i^{\mathrm{hi}})^*$, $\sigma_i \in \Sigma_i^{\mathrm{hi}}$ s.t. $s_i u_i \sigma_i \in \mathcal{L}(S_i^{\mathrm{lo}}/H_i)$. Consequently $i_k \notin \tilde{\mathcal{I}}$.

Now we construct t_k. To initialize the inductive argument, we observe that $p_{i_k}^{\mathrm{hi}}(s^{\mathrm{hi}}t_0) \in \mathcal{L}(H_{i_k}^{\mathrm{hi}})$ and $st_0 \in \mathcal{L}(H^{\mathrm{hi}})$. The induction assumption is $p_{i_k}^{\mathrm{hi}}(s^{\mathrm{hi}}t_1 \dots t_{k-1}) \in \mathcal{L}(H_{i_k}^{\mathrm{hi}})$ and $st_0t_1 \cdots t_{k-1} \in \mathcal{L}(H^{\mathrm{hi}})$. As $\Sigma_{\mathrm{p}_i^{\mathrm{hi}}(\mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))}(s_i^{\mathrm{hi}}) \neq \emptyset$ and $\mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$ does not contain ambiguous extensions w.r.t. $\Sigma_{i_k}^{\mathrm{hi}}$, there exists a $\sigma_{i_k} \in \Sigma_{i_k}$ and $\hat{t}_k \in (\Sigma^{\mathrm{hi}})^*$ with $p_{i_k}^{\mathrm{hi}}(\hat{t}_k) = \sigma_{i_k}$ such that $s^{\mathrm{hi}}t_0t_1 \cdots t_{k-1}\hat{t}_k \in \mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$ (and also $p_{i_k}^{\mathrm{hi}}(s^{\mathrm{hi}}t_0t_1 \cdots t_{k-1}\hat{t}_k) \in \mathcal{L}(S_i^{\mathrm{hi}}/H_{i_k}^{\mathrm{hi}})$). But as $S^{\mathrm{hi}}/H^{\mathrm{hi}}$ is nonblocking, there exists a $\tilde{t}_k \in (\Sigma^{\mathrm{hi}})^*$ such that $s^{\mathrm{hi}}t_1 \dots t_{k-1}\hat{t}_k\tilde{t}_k \in \mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$. Thus, $t_k := \hat{t}_k\tilde{t}_k$ fulfills the condition in item 3. of the algorithm.

Furthermore, the algorithm terminates as $\mathcal{I}$ is a finite index set which is reduced in every step.

Hence, the above algorithm provides a high-level string $t := t_0t_1 \cdots t_{k^*}$ s.t. $s^{\mathrm{hi}}t \in \mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$ and $p_i^{\mathrm{hi}}(t) \neq \epsilon$ for all $i \in \mathcal{I}$. Then, because of Lemma 4.3, $\forall i \in \mathcal{I}$, $\exists u_i \in \Sigma_i^*$ s.t. $s_iu_i \in \mathcal{L}_{\mathrm{m}}(S_i^{\mathrm{lo}}/H_i)$ and $p_i^{\mathrm{dec}}(s_iu_i) = p_i^{\mathrm{hi}}(s^{\mathrm{hi}}t)$.

For $i \notin \mathcal{I}$, it holds that $\Sigma_{\mathrm{p}_i^{\mathrm{hi}}(\mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))}(s_i^{\mathrm{hi}}) = \emptyset$. Thus, because of marked string controllability, there is a $u_i \in (\Sigma - \Sigma^{\mathrm{hi}})^*$ s.t. $s_iu_i \in \mathcal{L}_{\mathrm{m}}(S_i^{\mathrm{lo}}/H_i)$. Then $\forall u \in \|_{i=1}^n u_i$, it holds that $su \in \|_{i=1}^n s_iu_i \subseteq \|_{i=1}^n \mathcal{L}_{\mathrm{m}}(S_i^{\mathrm{lo}}/H_i)$ and $p^{\mathrm{hi}}(su) = s^{\mathrm{hi}}t \in \mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$. Thus $su \in \mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}}) \| \overline{\left(\|_{i=1}^n \mathcal{L}_{\mathrm{m}}(S_i^{\mathrm{lo}}/H_i)\right)} = \mathcal{L}_{\mathrm{m}}(S^{\mathrm{lo}}/H)$, which proves that $s \in \overline{\mathcal{L}_{\mathrm{m}}(S^{\mathrm{lo}}/H)}$. ■

The computational complexity for verifying the sufficient conditions is summarized in Table I. As this evaluation has to be carried out for all subsystems, the overall verification of the sufficient conditions for nonblocking hierarchical control is also polynomial in the number of individual subsystems.

Furthermore, we briefly discuss what can be done if any of the sufficient conditions is violated. A polynomial time algorithm for enforcing the marked string acceptance and the locally nonblocking condition by changing the high-level alphabet for each PCS $(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$, $i = 1, \dots n$ has been presented in [18]. It is currently investigated which sublanguages of a given high-level closed-loop language $\mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}})$ can be implemented if marked string controllability is violated or there are ambiguous extensions.

It also has to be noted that it can be shown that Theorem 4.1 still holds if marked string acceptance, marked string controllability and the absence of ambiguous extensions are replaced by the $\mathcal{L}_{\mathrm{m}}(H_i)$-observer property for each natural projection p_i^{dec} (see [11]), and the specification $K_i = \Sigma_i^*$ is used instead of K_i in Lemma 4.1[5]. As both sets of conditions are incomparable, it depends on each individual compound system and high-level specification which set of conditions yields better results.

V. LABORATORY CASE STUDY

A. Manufacturing System

An example for a large-scale compound discrete event system is the Fischertechnik simulation model of the Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg. It represents a manufacturing system (see Fig. 5(a)) that consists of a stack feeder, conveyor belts, pushers, rotary tables, machines and a rail transport system. The purpose of the manufacturing system is to process workpieces that enter the system from the stack feeder (`sf`) of the *distribution system* (`ds`) (compare Fig. 5). From there, the workpieces are distributed by a long conveyor belt (`c1`). The two pushers (`p1` and `p2`) attached to this conveyor belt transport workpieces to the actual production part of the system that is entered via two conveyor belts (`c2` and `c3`). Workpieces can be processed by the machines and then transported to one of the roll conveyors located at both ends of the rail transport system [see sample path in Fig. 5(a)]. The behavior of the manufacturing system can be described as a compound DES with interacting components.

Altogether, the system comprises 28 components, and a monolithic automaton model reaches an estimated number of 10^{24} states. Due to the size of the manufacturing system, monolithic supervisor synthesis is not advisable.

Common engineering practice suggests to form modular groups of subsystems according to local specifications, and to synthesize supervisors for such groups before combining locally controlled groups in order to impose superordinate specifications. Iterative application of this idea leads to a multilevel hierarchy of controllers, where subsystems that are related either because of the system structure or because of given specifications are grouped together. In this section, we will demonstrate how the theoretical results of Section IV can be applied to such a multilevel hierarchy constructed for the manufacturing system in Fig. 5(a). We first employ the proposed modeling and synthesis procedure for the distribution system in detail, and then give performance results for the overall manufacturing system.

In the laboratory case study, we use finite automata as a representation of control systems. As the supervisor design for the distribution system involves more than two hierarchical levels, the following notation is used for system components on different hierarchical levels.

[5]A technical proof of this statement is not in the scope of this paper.

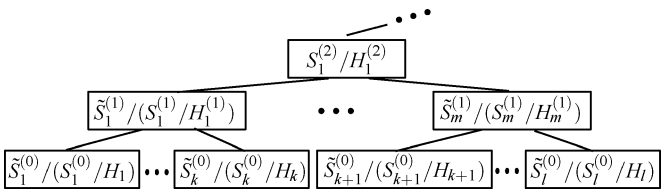


Fig. 6. Multilevel hierarchy.

1) Technical Comments: We write plant automata $G_j^{(i)}$, where i denotes the level of the hierarchy where the automaton model resides and j is the name of the component according to the schematic overview in Fig. 5(b). Given the alphabet $\Sigma_j^{(i)}$ of $G_j^{(i)}$ and the abstraction alphabet $\Sigma_j^{(i+1)} \subseteq \Sigma_j^{(i)}$, we denote $p_j^{(i,i+1)} : (\Sigma_j^{(i)})^* \rightarrow (\Sigma_j^{(i+1)})^*$ the respective natural projection. The control system corresponding to $G_j^{(i)}$ is $H_{G_j^{(i)}}$. Also, we represent high-level transitions in figures by dashed arrows, and a tick on an arrow indicates a controllable event.

The following statements clarify how the framework in Section IV-D can be applied to a multilevel hierarchy. The ideas are presented along with Fig. 6. In a multilevel hierarchy, groups of high-level closed-loops (e.g., $S_1^{(1)}/H_1^{(1)}, \ldots, S_m^{(1)}/H_m^{(1)}$) are again used as low-level models for the next higher level of the hierarchy (e.g., $H_1^{(2)}$). If marked string acceptance, locally nonblocking and marked string controllability are fulfilled for the respective PCS, Theorem 4.1 can be applied. Having computed a supervisor without ambiguous extensions for the current high-level system ($S_1^{(2)}$), an additional low-level supervisor according to Theorem 4.1 (e.g., $\tilde{S}_1^{(1)}$) has to be synthesized to implement the control action of the respective supervisor on the next higher level (e.g., $S_1^{(2)}$). The combined control action of the supervisors of such a subsystem (e.g., $\tilde{S}_1^{(1)}$ and $S_1^{(1)}$ for $H_1^{(1)}$) in turn has to be implemented for the corresponding lower-level systems according to Theorem 4.1 (e.g., $\tilde{S}_1^{(0)}$ for $\tilde{S}_1^{(1)}/\ (S_1^{(1)}/H_1^{(1)})$). Observing that the combined control action of the respective supervisors $\tilde{S}_j^{(i)}$ and $S_j^{(i)}$ can be represented as a single supervisor, we denote the resulting supervisor $S_j^{(i)}$. Consequently, our method can directly be used in a multilevel hierarchy by successive application of the described low-level supervisor computation starting from the supervisor on the highest level.

B. Supervisor Synthesis for the Distribution System

The distribution system as shown in Fig. 5(b) consists of the stack feeder `sf`, the two conveyor belts `c2` and `c3`, and the long conveyor belt `c1` with the pushers `p1` and `p2`. The conveyor belt `c1` is split into three parts for modeling convenience. The first part (`p2`) describes `c1` between the stack feeder and the pusher `p2`. The second part (`p1`) models `c1` from the pusher `p1` to the end of `c1`, and the third part (`con`) accounts for the physical connection between `p2` and `p1` via the belt. We construct the hierarchical architecture for the distribution system.

1) Stack Feeder: The stack feeder consists of a stack and a belt that can shove workpieces to the conveyor belt `c1` using small blocks that are attached to the belt. The belt's motion and end of motion are triggered by the events `sfmv` (move) and `sfstp` (stop), respectively. The stack feeder is equipped with a photoelectric barrier which detects if a workpiece is present. Arrival and departure of a workpiece generate the events `sfar` and `sflv`, respectively. The rest position of the small block is detected by a magnetic sensor which triggers the events `sfr` (rest position) and `sfnr` (not in the rest position).

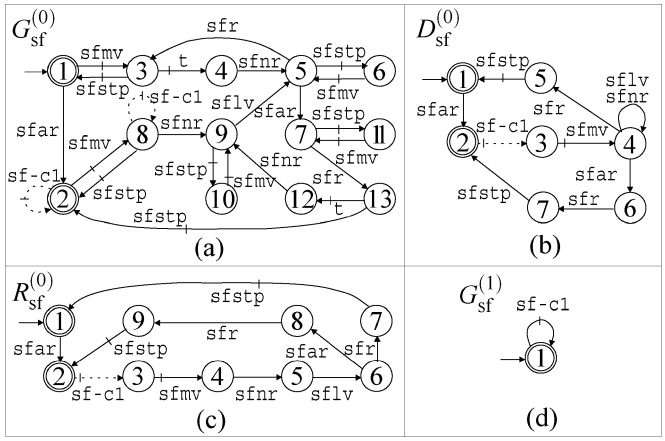


Fig. 7. Stack feeder: (a) Plant $G_{\mathrm{sf}}^{(0)}$; (b) Specification $D_{\mathrm{sf}}^{(0)}$; (c) Closed loop $R_{\mathrm{sf}}^{(0)}$; (d) Abstraction $G_{\mathrm{sf}}^{(1)}$.

The stack feeder is a control system. Fig. 7(a) shows the automata representation $G_{\mathrm{sf}}^{(0)}$ of its uncontrolled behavior with the set of controllable events $\Sigma_{\mathrm{sf,c}}^{(0)} = \{\mathtt{sfmv}, \mathtt{sfstp}, \mathtt{sf\text{-}c1}, \mathtt{t}\}$. The additional event `sf-c1` indicates that interaction with the neighboring component is possible in the respective state, i.e., a workpiece can be transported to the conveyor belt `c1`. The event `t` represents the elapse of a nonzero time period until the occurrence of the next event and captures the physical property that when the small block arrives at the rest position, the belt can be stopped before the rest position is left.

We specify that the stack feeder shall only move if a workpiece has been detected at the sensor (`sfar`), and if cooperation with the long conveyor belt $c1$ is possible via `sf-c1`. Also, the belt is allowed to stop only if the small block reaches the rest position ($D_{\mathrm{sf}}^{(0)}$ in Fig. 7(b)). The resulting closed-loop behavior $\kappa_{H_{\mathrm{sf}}^{(0)},\Sigma_{\mathrm{sf,uc}}^{(0)}}\left(L_{\mathrm{m}}(D_{\mathrm{sf}}^{(0)})\right)$ is implemented by the automaton $R_{\mathrm{sf}}^{(0)}$ in Fig. 7(c). The stack feeder is connected to the rest of the distribution system via the conveyor belt `c1`, and we choose the high-level alphabet $\Sigma_{\mathrm{sf}}^{(1)} = \{\mathtt{sf\text{-}c1}\}$ with the shared event `sf-c1`. The projection $p_{\mathrm{sf}}^{(0,1)}$ yields the high-level automaton model $G_{\mathrm{sf}}^{(1)}$ depicted in Fig. 7(d).

2) Conveyor Belt c1 and Pusher p2: The conveyor belt `c1` transports workpieces while moving into the negative x-direction (`c1-x`, see also coordinate plane in Fig. 5(b)). The arrival of a workpiece is detected at the pusher `p2` (`p2ar`). A workpiece that is present at `p2` can either continue its transport on `c1` (`sf-3`) or be pushed. In the latter case, `p2` extends toward `c2` (`sf-2`), delivers the workpiece, and retracts to the rest position (`p2rdy`). The low-level model $G_{\mathrm{p2}}^{(0)}$ has 43 states and is abstracted to the level 1 model $G_{\mathrm{p2}}^{(1)}$ in Fig. 8(a).

3) Conveyor Belt c1 and Pusher p1: The design process for the combination of `c1` and `p1` is analogous to the synthesis in Section V-B.II. To sum up, this component is able to detect workpieces at the sensor of pusher `p1` and push the workpieces

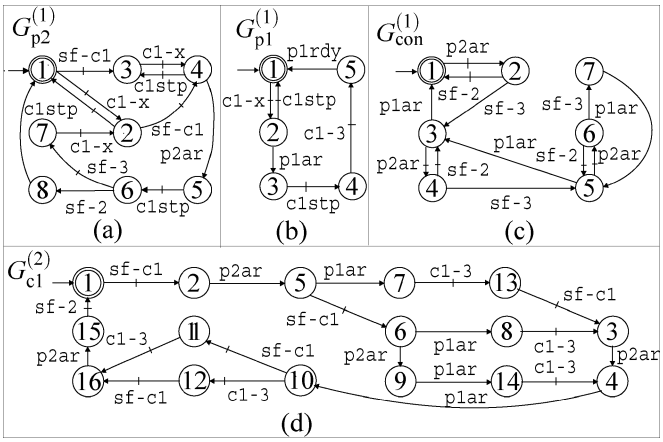


Fig. 8. c1: (a) $G^{(2)}_{\mathtt{c1}}$; (b) $G^{(1)}_{\mathtt{p2}}$; (c) $G^{(1)}_{\mathtt{p1}}$; (d) $G^{(1)}_{\mathtt{con}}$.

to the conveyor belt c3. It is important to mention that the component p1 is physically connected to p2 via the long conveyor belt. Consequently, the events c1-x and c1stp are shared between these components. The resulting model on level 1 is depicted in Fig. 8(b).

It can be verified that $(H_{R^{(0)}_{\mathtt{sf}}}, p^{(0,1)}_{\mathtt{sf}}, H^{(1)}_{\mathtt{sf}})$, $(H_{R^{(0)}_{\mathtt{p2}}}, p^{(0,1)}_{\mathtt{p2}}, H^{(1)}_{\mathtt{p2}})$ and $(H_{R^{(0)}_{\mathtt{p1}}}, p^{(0,1)}_{\mathtt{p1}}, H^{(1)}_{\mathtt{p1}})$ are locally nonblocking, marked string accepting and marked string controllable. Hence, it is possible to use $G^{(1)}_{\mathtt{sf}}$, $G^{(1)}_{\mathtt{p2}}$ and $G^{(1)}_{\mathtt{p1}}$ as components in a compound system on level 1.

4) Connection Between p2 and p1: There are logical conditions that restrict the synchronized behavior of p2 and p1. A workpiece can only arrive at the sensor of p1 (p1ar) if it has passed the sensor of p2 (p2ar). In addition to that, there cannot be more than three workpieces within the space between the two pushers. The automaton $G^{(1)}_{\mathtt{con}}$ as shown in Fig. 8(c) captures these constraints.

5) Entire Conveyor Belt c1: Composing the above automata, the model of the entire conveyor belt c1 is obtained as $G^{(1)}_{\mathtt{c1}} = G^{(1)}_{\mathtt{p2}} || G^{(1)}_{\mathtt{con}} || G^{(1)}_{\mathtt{p1}}$. The resulting automaton has 67 states.

For safety reasons, we require that the conveyor belt has to stop (c1stp) if a workpiece arrives at one of the pushers (p1ar or p2ar). If the conveyor belt is empty, it is desired to start moving (c1-x) only if a workpiece is delivered from the stack feeder (sf-c1). Otherwise, if c1 is not empty, a workpiece can be delivered from the stack feeder only if the conveyor belt is moving. Further on, we implement an (arbitrary) desired manufacturing routine: The first two workpieces must always be pushed by p1 (c1-3) and then, one workpiece has to be pushed by p2 (sf-2), and so on.

The closed-loop behavior of c1 is represented by an automaton $R^{(1)}_{\mathtt{c1}}$ with 39 states. For hierarchical abstraction, the shared events sf-c1, c1-3 and sf-2 as well as the events p1ar and p2ar are contained in the high-level alphabet $\Sigma^{(2)}_{\mathtt{c1}} = \{\mathtt{sf\text{-}c1}, \mathtt{c1\text{-}3}, \mathtt{sf\text{-}2}, \mathtt{p1ar}, \mathtt{p2ar}\}$. The automaton $G^{(2)}_{\mathtt{c1}}$, representing the abstracted behavior is shown in Fig. 8(d). The PCS $(H_{R^{(1)}_{\mathtt{c1}}}, p^{(1,2)}_{\mathtt{c1}}, H^{(2)}_{\mathtt{c1}})$ is locally nonblocking, marked string accepting and marked string controllable.

6) Conveyor Belts c2 and c3: The conveyor belts c2 and c3 are used in the same mode of operation. Both conveyor belts receive workpieces from the respective pusher and transport them to the production part of the manufacturing system.

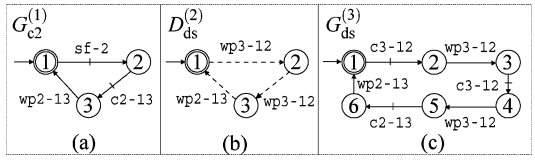


Fig. 9. (a) c2 on level 2 $G^{(1)}_{\mathtt{c2}}$; Distribution system: (b) Specification $D^{(2)}_{\mathtt{ds}}$ (c) Abstraction $G^{(3)}_{\mathtt{ds}}$.

The shared events of c2 with the other components of the manufacturing system are $\Sigma^{(1)}_{\mathtt{c2}} = \{\mathtt{sf\text{-}2}, \mathtt{c2\text{-}13}, \mathtt{wp2\text{-}13}\}$. The automata representation $G^{(1)}_{\mathtt{c2}}$ on level 1 is shown in Fig. 9(a).

7) Overall Distribution System: The overall distribution system is constructed on level 2 of the hierarchy. To this end, let the models of the stack feeder and the conveyor belts cb2 and cb3 on level 2 be equal to the respective models on level 1, i.e., $G^{(2)}_{\mathtt{sf}} = G^{(1)}_{\mathtt{sf}}$, $G^{(2)}_{\mathtt{c2}} = G^{(1)}_{\mathtt{c2}}$ and $G^{(2)}_{\mathtt{c3}} = G^{(1)}_{\mathtt{c3}}$. Then, the automata representation of the distribution system is $G^{(2)}_{\mathtt{ds}} = G^{(2)}_{\mathtt{sf}} || G^{(2)}_{\mathtt{c1}} || G^{(2)}_{\mathtt{c2}} || G^{(2)}_{\mathtt{c3}}$. It has 144 states. We now specify that always two workpieces leave the conveyor belt c3 before one workpiece can leave c2 [see Fig. 9(b)].

The supervisor automaton $R^{(2)}_{\mathtt{ds}}$ on level 2 has 138 states. We choose the set of shared events with the production part of the manufacturing system as the high-level alphabet $\Sigma^{(3)}_{\mathtt{ds}} = \{\mathtt{c3\text{-}12}, \mathtt{wp3\text{-}12}, \mathtt{c2\text{-}13}, \mathtt{wp2\text{-}13}\}$. Abstracting the distribution system to the third level results in the automaton $G^{(3)}_{\mathtt{ds}}$ as depicted in Fig. 9(c). The projected control system $(H_{R^{(2)}_{\mathtt{ds}}}, p^{(2,3)}_{\mathtt{ds}}, H_{G^{(3)}_{\mathtt{ds}}})$ is locally nonblocking, marked string accepting and marked string controllable. Thus, the level 3 model of the distribution system can be used as a subsystem in the overall model of the manufacturing system.

The entire hierarchy for the distribution system is shown in Fig. 10. The numbers next to the automata names represent the respective number of states. Note that in this example, the iterative computation of low-level supervisors starting from the highest-level supervisor, as described in Section V-A, does not change the respective low-level closed-loop systems.

C. Performance Evaluation

The level 3 model of the distribution system has been constructed starting from low-level models of the different components sf, p2, p1, con, c2 and c3. To classify the efficiency of our hierarchical and decentralized method, it is compared to the monolithic approach. The composite plant automaton $G^{(0)}_{\mathtt{ds}} = G^{(0)}_{\mathtt{sf}} || G^{(0)}_{\mathtt{p1}} || G^{(0)}_{\mathtt{p2}} || G^{(0)}_{\mathtt{c1_{con}}} || G^{(0)}_{\mathtt{c2}} || G^{(0)}_{\mathtt{c3}}$ has 360 000 states, and the monolithic specification is represented by an automaton $D^{(0)}_{\mathtt{ds}}$ with $3 \cdot 10^6$ states. Applying standard supervisory control, the closed-loop automaton $R^{(0)}_{\mathtt{ds}}$ has 400 000 states.

Different from that, the decentralized approach results in five decentralized supervisors with a sum of 71 states on the low level, one supervisor with 39 states on level 1 and one supervisor with 69 states on the second level. Together, the hierarchical and decentralized supervisors have 179 states. The considerable discrepancy in the state sizes of the different supervisor implementations originates from the fact that the state sizes of the compo-

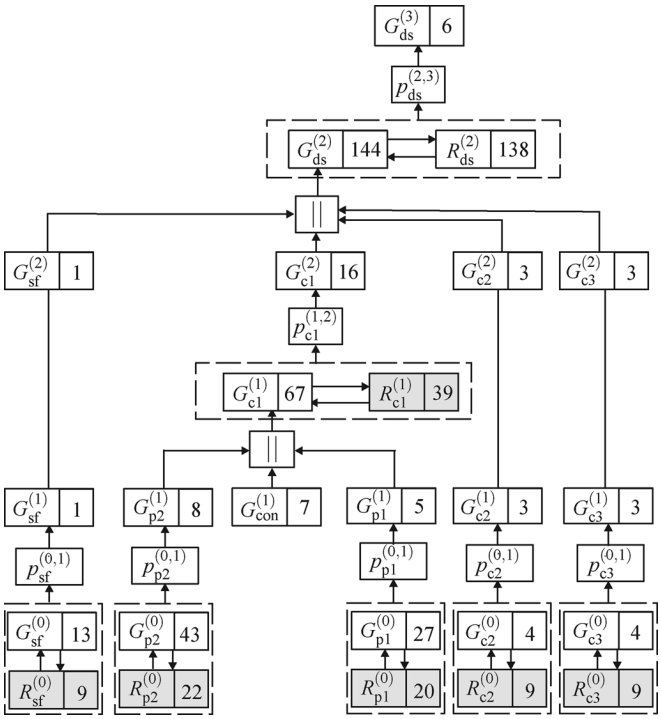


Fig. 10. Hierarchical architecture for the distribution system.

nents have to be multiplied for the monolithic approach while they are just summed up for the hierarchical and decentralized method. It is also important to note that not only the number of states of the decentralized supervisors is smaller, but also the complexity for computing the supervisors is by far lower than for the monolithic approach.

Analogous to the supervisor synthesis for the distribution system, controllers for the remaining components of the manufacturing system in Fig. 5 and an overall supervisor for the compound plant have been designed (see [18]). While a monolithic supervisor for the manufacturing system would have an estimated number of 10^{30} states, synthesis in the hierarchical and decentralized framework results in 39 individual supervisors whose state count sums up to 5388 states. It can be verified that none of the high-level closed-loop behaviors contains ambiguous strings and that all projected control systems fulfill the sufficient conditions for nonblocking low-level control. Correct operation of the manufacturing system could be verified after generating PLC-code from the hierarchical and decentralized supervisors.

VI. Conclusion

The supervisory control of large-scale compound discrete event systems involves computations on state spaces which grow exponentially with the number of plant components. To circumvent this limitation, we present a method for exploiting the decentralized structure of compound systems in combination with hierarchical control.

We develop an abstraction procedure that makes it possible to first abstract the decentralized low-level system components and then compose the resulting decentralized high-level components to the overall high-level model. We also determine a hierarchically consistent decentralized low-level supervisor implementation and formulate sufficient conditions for nonblocking closed-loop behavior of the original system. Furthermore, our approach guarantees that the high-level model is less complex than the low-level model. On the one hand, both the abstraction method and the low-level supervisor implementation do not refer to the overall low-level model, and thus the possibly very large state space need not be enumerated explicitly. On the other hand, maximal permissiveness of the low-level supervisor might be lost, i.e., in the worst case, it is possible that the closed-loop language is empty while a maximally permissive monolithic supervisor could yield non-empty closed-loop behavior.

The computational benefit of our method is illustrated by the manufacturing system example in Section V. It comprises 28 components, and it has an estimated number of 10^{24} states. The 39 individual supervisors designed by using our hierarchical and decentralized method with four levels of abstraction and control have an average number of 140 states. In comparison, we would expect a supervisor of order 10^{30} states for a monolithic implementation.

Ongoing work aims at automating the hierarchical abstraction process such that the sufficient conditions for nonblocking low-level control are fulfilled. A first result in this direction is provided in [29], where we present an algorithm that enforces the locally nonblocking and marked string accepting condition by changing the high-level alphabet for each decentralized system component.

Appendix A
Proof of Lemma 4.2

Proof: There are two cases. First assume that $\Sigma_{\mathrm{p}_i^{\mathrm{hi}}(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))}(s^{\mathrm{hi}}) = \emptyset$. Note that marked string controllability implies that $\kappa_{H_i(\mathrm{s}),\Sigma_{i,\mathrm{uc}}}(L_{i,\mathrm{m}}(s)) \neq \emptyset$, and thus there is no $v \in \Sigma_{i,\mathrm{uc}}^*$ s.t. $sv \in \mathcal{L}(H_i)$ and $sv \notin \overline{\tilde{K}_i}$. Next, we show that also for $s' < s$ there is no $v \in \Sigma_{i,\mathrm{uc}}^*$ s.t. $s'v \in \mathcal{L}(H_i)$ and $s'v \notin \overline{\tilde{K}_i}$ by contradiction. Let $v = v_1\tau_1 \cdots v_l\tau_l \in \Sigma_{i,\mathrm{uc}}^*$ be a shortest string s.t. $s'v \in \mathcal{L}(H_i)$ and $s'v \notin \overline{\tilde{K}_i}$ for some $s' < s$, $v_j \in (\Sigma_{i,\mathrm{uc}} - \Sigma_{i,\mathrm{uc}}^{\mathrm{hi}})^*$, $\tau_j \in \Sigma_{i,\mathrm{uc}}^{\mathrm{hi}}$ for $j = 1, \ldots, l$. Thus, $s'v_1 \ldots, \tau_{l-1}v_l \in \overline{\tilde{K}_i}$ and $s'v \notin \overline{\tilde{K}_i}$. Also, it must be true that $\Sigma_{\mathrm{p}_i^{\mathrm{hi}}(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))}\big(p_i^{\mathrm{dec}}(s'\tau_1 \cdots \tau_{l-1})\big) = \emptyset$, as otherwise $s'v_1 \cdots v_{l-1}\tau \in \overline{\tilde{K}_i}$ for all $\tau \in \Sigma_i$ due to the construction of $\tilde{K}_i$. Observing that marked string controllability is valid, $v_{l-1}\tau_l \notin \mathcal{L}(H_i(s'v_1 \cdots v_{l-1}\sigma_{l-1}))$ because otherwise $\kappa_{H_i(s'v_1 \cdots v_{l-1}\sigma_{l-1}),\Sigma_{i,\mathrm{uc}}}(L_{i,\mathrm{m}}(s'v_1 \cdots v_{l-1}\sigma_{l-1})) = \emptyset$. Together, this implies that v as defined above does not exist for any $s' \leq s$.

Secondly, assume that $\Sigma_{\mathrm{p}_i^{\mathrm{hi}}(\mathcal{L}_{\mathrm{m}}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))}(s^{\mathrm{hi}}) \neq \emptyset$. Then, the same argument as above shows that there is no $v \in \Sigma_{i,\mathrm{uc}}^*$ s.t. $sv \in \mathcal{L}(H_i)$ and $sv \notin \overline{\tilde{K}_i}$.

Hence, as $s \in \overline{\tilde{K}_i} \cap \mathcal{L}(H_i)$ and $\nexists v \in \Sigma_{i,\mathrm{uc}}^*$ s.t. $s'v \in \mathcal{L}(H_i)$ and $s'v \notin \overline{\tilde{K}_i}$ for any $s' \leq s$, it holds that $s \in K_i = \kappa_{H_i,\Sigma_{i,\mathrm{uc}}}(\tilde{K}_i)$. ■

APPENDIX B
PROOF OF LEMMA 4.3

Proof: Let s^{hi}, s_i and t be given as in Lemma 4.3. Defining $v_0 = \sigma_0 = \epsilon$, t can be represented as $t = \sigma_0\sigma_1\sigma_2\cdots\sigma_m$ with $\sigma_j \in \Sigma_i^{\mathrm{hi}}$ for $j = 1,\ldots,m$. First it is shown that there exists a string $u_i' = v_0\sigma_0 v_1\sigma_1\cdots v_m\sigma_m \in \Sigma_i^*$ with $v_j \in (\Sigma_i - \Sigma_i^{\mathrm{hi}})^*$ for $j = 1,\ldots,m$ s.t. $s_iu_i' \in \mathcal{L}(H_i)$ by induction. The base case is easily verified as $s_iv_0\sigma_0 = s_i \in \mathcal{L}(H_i)$. For the induction step, let $v_0\sigma_0v_1\sigma_1\cdots v_j\sigma_j \in \mathcal{L}(S_i^{\mathrm{lo}}/H_i)$ for $j \in \{1,\ldots,m-1\}$. Then, as $(H_i, p_i^{\mathrm{dec}}, H_i^{\mathrm{hi}})$ is locally nonblocking (Definition 4.6), there exists $v_{j+1} \in (\Sigma_i - \Sigma_i^{\mathrm{hi}})^*$ s.t. $s_iv_0\sigma_0v_1\cdots\sigma_jv_{j+1}\sigma_{j+1} \in \mathcal{L}(H_i)$. As this applies for all $j = 0,\ldots,m$, it holds that $s_iu_i' = s_iv_0\sigma_0\cdots v_m\sigma_m \in \mathcal{L}(H_i)$ and $p^{\mathrm{hi}}(u_i') = t$. Furthermore, because of the construction of u_i', $s_iu_i' \in L_{i,\mathrm{en}}(s^{\mathrm{hi}}t)$. According to Lemma 4.2, $s_iu_i' \in K_i$, and hence $s_iu_i' \in \mathcal{L}(S_i^{\mathrm{lo}}/H_i)$.

Next, a string $v_i' \in (\Sigma_i - \Sigma_i^{\mathrm{hi}})^*$ s.t. $s_iu_iv_i' \in \mathcal{L}_{\mathrm{m}}(S_i^{\mathrm{lo}}/H_i)$ is obtained. There are two cases. If $\Sigma_{\mathrm{p}_i^{\mathrm{hi}}(\mathcal{L}(S^{\mathrm{hi}}/H^{\mathrm{hi}}))} = \emptyset$, then there exists a $v_i' \in \kappa_{H_i(s_iu_i'),\Sigma_{i,\mathrm{uc}}}(L_{i,\mathrm{m}}(s_iu_i'))$ s.t. $s_iu_iv_i' \in \mathcal{L}_{\mathrm{m}}(H_i)$ because of marked string controllability. Otherwise, because of marked string acceptance, there is a string $v_i' \in (\Sigma_i - \Sigma_i^{\mathrm{hi}})^*$ s.t. $s_iu_iv_i' \in \mathcal{L}_{\mathrm{m}}(H_i)$. In both cases, $s_iu_iv_i' \in K_i$. Again, as K_i is controllable w.r.t H_i, $s_iu_iv_i' \in \mathcal{L}_{\mathrm{m}}(S_i^{\mathrm{lo}}/H_i)$. Finally, choosing $u_i = u_iv_i'$, we have that $s_iu_i \in \mathcal{L}_{\mathrm{m}}(S_i^{\mathrm{lo}}/H_i)$ and $p_i^{\mathrm{dec}}(s_iu_i) = p_i^{\mathrm{hi}}(s^{\mathrm{hi}}t)$. ■

REFERENCES

[1] P. J. Ramadge and W. M. Wonham, "Modular supervisory control of discrete event systems," *Math. Control Discrete Event Syst.*, vol. 1, no. 1, pp. 13–30, 1988.

[2] M. H. de Queiroz and J. E. R. Cury, "Modular supervisory control of large scale discrete event systems," in *Proc. Workshop Discrete Event Syst.*, 2000, pp. 103–110.

[3] J. Komenda, J. H. van Schuppen, B. Gaudin, and H. Marchand, "Modular supervisory control with general indecomposable specification languages," in *Proc. Conf. Decision Control*, 2005, pp. 3474–3479.

[4] T. Yoo and S. Lafortune, "A generalized architecture for decentralized supervisory control of discrete event systems," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 12, no. 3, pp. 335–377, Jul. 2002.

[5] S. Jiang, V. Chandra, and R. Kumar, "Decentralized control of discrete event systems with multiple local specializations," in *Proc. Amer. Control Conf.*, 2001, pp. 653–660.

[6] S.-H. Lee and K. C. Wong, "Structural decentralised control of concurrent DES," *Eur. J. Control*, vol. 35, pp. 1125–1134, 2002.

[7] J. Komenda and J. H. van Schuppen, "Optimal solutions of modular supervisory control problems with indecomposable specification languages," in *Proc. Workshop Discrete Event Syst.*, 2006, pp. 143–148.

[8] C. Ma, "Nonblocking Supervisory Control of State Tree Structures," Ph.D. dissertation, Dept. of Elect. and Comput. Eng., Univ. of Toronto, , Toronto, ON, Canada, 2004.

[9] B. Gaudin and H. Marchand, "Safety control of hierarchical synchronous discrete event systems: A state-based approach," in *Proc. Mediterranean Conf. Control Automat.*, 2005, pp. 889–895.

[10] H. Zhong and W. M. Wonham, "On the consistency of hierarchical supervision in discrete-event systems," *IEEE Trans. Automat. Control*, vol. 35, no. 10, pp. 1125–1134, Oct. 1990.

[11] K. C. Wong and W. M. Wonham, "Hierarchical control of discrete-event systems," *Discrete Event Dyn. Syst.: Theory Appl.*, vol. 6, no. 3, pp. 241–273, Jul. 1996.

[12] A. E. C. da Cunha, J. E. R. Cury, and B. H. Krogh, "An assume guarantee reasoning for hierarchical coordination of discrete event systems," in *Proc. Workshop Discrete Event Syst.*, 2002, pp. 75–80.

[13] P. Hubbard and P. E. Caines, "Dynamical consistency in hierarchical supervisory control," *IEEE Trans. Automat. Control*, vol. 47, no. 1, pp. 37–52, Jan. 2002.

[14] T. Moor, J. Raisch, and J. M. Davoren, "Admissibility criteria for a hierarchical design of hybrid control systems," in *Proc. Conf. Anal. Design Hybrid Syst.*, 2003, pp. 389–394.

[15] K. Schmidt, J. Reger, and T. Moor, "Hierarchical control of structural decentralized DES," in *Proc. Workshop Discrete Event Syst.*, Reims, France, 2004, pp. 289–295.

[16] R. Hill and D. Tilbury, "Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction," in *Proc. Workshop Discrete Event Syst.*, 2006, pp. 3–8.

[17] L. Feng and W. M. Wonham, "Computationally efficient supervisor design: Abstraction and modularity," in *Proc. Workshop Discrete Event Syst.*, 2006, pp. 399–406.

[18] K. Schmidt, "Hierarchical Control of Decentralized Discrete Event Systems: Theory and Application," Ph.D. dissertation, Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg, Erlangen, Germany, 2005.

[19] K. Schmidt, H. Marchand, and B. Gaudin, "Modular and decentralized supervisory control of concurrent discrete event systems using reduced system models," in *Proc. Workshop Discrete Event Syst. (WODES)*, Ann Arbor, MI, 2006, pp. 149–154.

[20] S. Perk, T. Moor, and K. Schmidt, "Hierarchical discrete event systems with inputs and outputs," in *Proc. Workshop Discrete Event Syst. (WODES)*, Ann Arbor, MI, 2006, pp. 427–432.

[21] C. C. Torrico and J. E. R. Cury, "Hierarchical supervisory control of discrete event systems based on state aggregation," in *Proc. IFAC World Congress*, 2002, [CD ROM].

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

[23] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE, Special Issue Discrete Event Dyn. Syst.*, vol. 77, no. 1, pp. 81–98, Jan. 1989.

[24] W. M. Wonham, "Notes on Control of Discrete Event Systems," Department of Electrical Engineering, University of Toronto, Toronto, ON, Canada, 2004.

[25] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*. Reading, MA: Addison-Wesley, 1979.

[26] J. J. M. M. Rutten, "Coalgebra, Concurrency, and Control," Centrum voor Wiskunde en Informatica, Tech. Rep. SEN-R9921, 1999.

[27] K. C. Wong and W. M. Wonham, "Modular control and coordination of discrete event systems," *Discrete Event Dyn. Syst.*, vol. 8, no. 3, pp. 908–929, 1998.

[28] J.-C. Fernandez, "An implementation of an efficient algorithm for bisimulation equivalence," *Sci. Comput. Program.*, vol. 13, pp. 219–236, 1990.

[29] K. Schmidt and T. Moor, "Computation of marked string accepting observers for discrete event systems," in *Proc. Workshop Discrete Event Syst. (WODES)*, Ann Arbor, MI, 2006, pp. 413–418.

Klaus Schmidt received the Diploma and Ph.D. (with distinction) degrees in electrical, electronic, and communication engineering from the University of Erlangen-Nuremberg, Erlangen, Germany, in 2002 and 2005, respectively.

He is currently a Post-Doctoral Researcher with the Chair of Automatic Control, University of Erlangen-Nuremberg. His research interests include controller synthesis for discrete event systems, networked control systems, and vehicular communication networks.

Thomas Moor received the Diploma degree in mathematics from the University of Hamburg, Hamburg, Germany, in 1996 and the Ph.D. degree in electrical engineering from the University of the Federal Armed Forces, Hamburg, Germany, in 1999.

From 2000 to 2003, he was a Research Fellow with the Department of Systems Engineering, Research School of Information Sciences and Engineering, Australian National University, Canberra. Since 2003, he has been Head of the Discrete Event Systems Group, Chair of Automatic Control, University of Erlangen-Nuremberg, Erlangen, Germany. His research interests include hierarchical, decentralized, and modular control systems, discrete event systems, and hybrid systems.

Sebastian Perk received the Diploma degree in electrical, electronic and communication engineering from the University of Erlangen-Nuremberg, Erlangen, Germany, in 2005 where he is currently pursuing the Ph.D. degree.

His research interests include controller synthesis for discrete event systems and input-output based design of discrete event systems.