Maximally Permissive Hierarchical Control of Decentralized Discrete Event Systems

Klaus Schmidt, *Member, IEEE*, and Christian Breindl

***Abstract*—The subject of this paper is the synthesis of *natural projections* that serve as *nonblocking* and *maximally permissive* abstractions for the hierarchical and decentralized control of large-scale discrete event systems. To this end, existing concepts for nonblocking abstractions such as *natural observers* and *marked string accepting (msa)-observers* are extended by *local control consistency* (LCC) as a novel sufficient condition for maximal permissiveness. Furthermore, it is shown that, similar to the natural observer condition and the msa-observer condition, also LCC can be formulated in terms of a *quasi-congruence*. Based on existing algorithms in the literature, this allows to algorithmically compute natural projections that are either natural observers or msa-observers and that additionally fulfill LCC. The obtained results are illustrated by the synthesis of nonblocking and maximally permissive supervisors for a manufacturing system.**

***Index Terms*—Decentralized control, discrete event systems, hierarchical control, large-scale systems, maximal permissiveness, supervisory control.**

I. Introduction

THE use of *hierarchical abstractions* is a common feature of various supervisory control approaches that aim at reducing the computational effort of the supervisor synthesis for large-scale discrete event systems (DES) [1]–[13]. While early work on this topic is mostly devoted to purely *vertical* system structure [1], [2], [6], more recent approaches additionally exploit the *horizontal* composition of large-scale DES so as to avoid the enumeration of the global state space and the related state space explosion. In this respect, techniques such as in [3]–[5], [12], [13] are stated for hierarchies with two levels, whereas the methods in [7]–[11] are applicable in hierarchies with multiple levels.

The major concern of the above approaches is the synthesis of *nonblocking* supervisors, where different sufficient conditions on the hierarchical abstractions are employed in order to guarantee that the closed loop is nonblocking. However, the *optimality* of the synthesis is not ensured in most of the cited methods such that the resulting supervisor might be more restrictive than a maximally permissive monolithic supervisor. Although it can be argued in many cases that it is already satisfactory to determine some supervisor that fulfills a given specification, there might be cases where the synthesis of a maximally permissive supervisor is essential. Such situation arises for instance if the hierarchical supervisor synthesis results in an empty closed loop. Then, it cannot be decided if the control problem does not have a solution or if the supervisor synthesis fails due to the loss of optimality.

In this paper, we propose a unified treatment of nonblocking and maximally permissive hierarchical supervisory control for DES based on the hierarchical and decentralized control architecture in [8]. In this architecture, different conditions on the *natural projections*, that are employed for hierarchical abstraction, such as the natural observer condition in [10] or the marked string accepting (msa)-observer condition in [14] are sufficient for nonblocking control. As an extension to these results, we develop *local control consistency* (LCC) as a novel sufficient condition for maximal permissiveness. We show that LCC is less restrictive than *output control consistency* (OCC) that is employed for maximally permissive control in earlier work [1], [2], [10].

Considering the sufficient conditions for nonblocking and maximally permissive hierarchical control developed in this paper, our further goal is to algorithmically find natural projections that fulfill these conditions. In this regard, we refer to the *natural observer extension algorithm* in [15] that iteratively extends a given alphabet until the related natural projection is a natural observer for a given language. We first note that this algorithm does not rely on the specific observer property to be achieved but only utilizes the fact that the observer property can be formulated as a *quasi-congruence* (i.e., a particular equivalence relation) on the state space of a recognizer for the given language. Hence, we propose a *generalized extension algorithm* that can be applied to compute natural projections with properties that can be formulated as quasi-congruences. Moreover, we show that also the msa-observer property and local control consistency can be stated in terms of quasi-congruences. Consequently, we obtain a unified method for the computation of natural observers or msa-observers which are additionally locally control consistent. In both cases, we achieve nonblocking and maximally permissive supervision in our hierarchical and decentralized control architecture.

The computation of hierarchical abstractions based on quasi-congruences for the recognizer of a given language is also studied in [14]–[16]. In [16], the computation of optimal abstractions requires the use of *causal reporter maps* instead of natural projections. [14] and [15] enable the computation of natural projections that are suitable for the framework established in this paper, where [14] is based on a relabeling scheme. Since we focus on the formulation of conditions on natural

Manuscript received March 25, 2009; September 22, 2009; accepted June 30, 2010. Date of publication August 16, 2010; date of current version April 06, 2011. Recommended by Associate Editor E. Fabre.

K. Schmidt is with the Electronic and Communication Engineering Department, Çankaya University, Ankara 06530, Turkey (e-mail: schmidt@cankaya.edu.tr).

C. Breindl is with the Institute for Systems Theory and Automatic Control, University of Stuttgart, Stuttgart D-70174, Germany (e-mail: christian.breindl@ist.uni-stuttgart.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TAC.2010.2067250

projections in terms of quasi-congruences, this paper employs the algorithm in [15] that does not require relabeling.

Further hierarchical abstraction techniques and control architectures for the nonblocking and/or maximally permissive supervisory control of DES are investigated in the literature. The approaches in [1], [2], [6] employ causal reporter maps in a two-level hierarchy that relies on a global system model. While nonblocking control is not addressed in [1], it is ensured in [2] for causal reporter maps with the observer property. The computation of such reporter maps is investigated in [16], while the algorithm for achieving output control consistency in [1] enables the computation of maximally permissive hierarchical abstractions in both approaches. The hierarchical abstraction in [6] results in a DES with *flexible marking*. Sufficient conditions such as the *weak observer* property and *deterministic reporter maps* are defined to ensure consistency between the high-level and low-level control, and it is indicated that appropriate abstractions can be determined algorithmically.

Recently, several approaches that employ *nondeterministic automata* in the hierarchical abstraction process were developed [7], [11], [13]. The method in [7] is based on heuristics for *supervision equivalent* hierarchical abstractions. It performs a compositional synthesis to obtain a nonblocking and maximally permissive supervisor in a centralized representation. Dropping the requirement of maximal permissiveness, [11] elaborates an incremental synthesis of nonblocking modular supervisors and *coordinating filters* in the same framework. A new hierarchical abstraction technique based on nondeterministic automata is introduced in [13]. It supports the synthesis of nonblocking supervisors, whereas maximal permissiveness is not guaranteed.

Natural projections are used in hierarchical and decentralized architectures with two levels [4], [12] and multiple levels [8]–[10]. The method in [4] is based on the definition of *interfaces* that enable the information exchange between the hierarchical levels. This technique makes the verification of the *level-wise* conditions for nonblocking control scalable, while optimality of the control may be lost. Necessary and sufficient conditions for the *coordination control* of DES are studied in [12]. In this work, the computation of hierarchical abstractions is not considered, and specifications are required to be *conditionally decomposable*. The approach in [10] is suitable for nonblocking and maximally permissive control. The computation of appropriate natural projections is achieved by a successive application of the natural observer extension algorithm in [15] and a variation of the algorithm in [1] to achieve OCC. However that work relies on the more conservative OCC condition and does not provide a unified framework for the computation of natural projections. DES with an *input/output* structure are considered in [9], while maximal permissiveness is not addressed. In this approach, high-level specifications serve as system abstractions, and supervisors are computed such that the closed loop is free of deadlock and livelocks. Our previous work in [8] establishes the msa-observer condition for natural projections as a sufficient condition for nonblocking hierarchical and decentralized control. In the present paper, we extend this work by maximal permissiveness and algorithmically compute appropriate hierarchical abstractions. To this end, results obtained in [14], [17] are incorporated in the presentation of our work.

The remainder of the paper is organized as follows. In Section II, we summarize basic notions related to the supervisory control of DES and to set theory. A detailed discussion of existing results for the hierarchical and decentralized control of DES is provided in Section III. These results are then extended by our novel conditions for maximally permissive control in Section IV and by our unified method for the computation of projections for the nonblocking and maximally permissive hierarchical and decentralized control in Section V. Section VI illustrates the proposed approach by a manufacturing system example, and Section VII gives conclusions.

II. Preliminaries

At first, basic notions of the supervisory control theory for DES are summarized [18], [19].

A. DES Notation

For a finite alphabet Σ, the set of all finite strings over Σ is denoted Σ^*. We write $s_1 s_2 \in \Sigma^*$ for the concatenation of two strings s_1, $s_2 \in \Sigma^*$ and $s_1 \leq s$ when s_1 is a *prefix* of s. The empty string is denoted $\varepsilon \in \Sigma^*$, i.e., $s\varepsilon = \varepsilon s = s$ for all $s \in \Sigma^*$. A *language* over Σ is a subset $L \subseteq \Sigma^*$. 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 $L = \overline{L}$.

The *natural projection* $p_i : \Sigma^* \to \Sigma_i^*$, $i = 1, 2$, for the (not necessarily disjoint) union $\Sigma = \Sigma_1 \cup \Sigma_2$ is defined iteratively: (1) let $p_i(\varepsilon) := \varepsilon$; (2) for $s \in \Sigma^*$, $\sigma \in \Sigma$, let $p_i(s\sigma) := p_i(s)\sigma$ if $\sigma \in \Sigma_i$, or $p_i(s\sigma) := p_i(s)$ otherwise. The set-valued inverse of p_i is denoted $p_i^{-1} : \Sigma_i^* \to 2^{\Sigma^*}$, $p_i^{-1}(t) := \{s \in \Sigma^* | p_i(s) = t\}$. The *synchronous product* $L_1 \| L_2 \subseteq \Sigma^*$ of two languages $L_i \subseteq \Sigma_i^*$ is $L_1 \| L_2 = p_1^{-1}(L_1) \cap p_2^{-1}(L_2) \subseteq \Sigma^*$.

A *nondeterministic* automaton is a five-tuple $H = (X, \Sigma, \delta, x_0, X_m)$ with the set of *states* X, the *alphabet* Σ, the *transition function* $\delta : X \times \Sigma \to 2^X$, the *initial state* x_0 and the set of *marked states* $X_m \subseteq X$. We write $\delta(x, \sigma)!$ if $\delta(x, \sigma) \neq \emptyset$. In order to extend δ to a partial function on $X \times \Sigma^*$, recursively let $\delta(x, \epsilon) := \{x\}$ and $\delta(x, s\sigma) := \bigcup_{x' \in \delta(x,s)} \delta(x', \sigma)$. If $\delta(x, \sigma)$ contains at most one element for any $x \in X$ and $\sigma \in \Sigma$, then H is denoted as *deterministic*. In that case, we consider $\delta : X \times \Sigma \to X$ as the transition function. In the sequel, we model DES by deterministic automata unless otherwise stated. $L(H) := \{s \in \Sigma^* : \delta(x_0, s)!\}$ and $L_m(H) := \{s \in L(H) : \delta(x_0, s) \in X_m\}$ are the *closed* and *marked language* generated by H, respectively. A formal definition of the synchronous composition $H_1 \| H_2$ of two automata H_1 and H_2 can be taken from, e.g., [20].

In the supervisory control context, we write $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$, where Σ_{uc} is the set of *uncontrollable events* and Σ_c is the set of *controllable events*. A *control pattern* is a set γ, $\Sigma_{uc} \subseteq \gamma \subseteq \Sigma$, and the set of all control patterns is denoted $\Gamma \subseteq 2^\Sigma$. A *supervisor* for an automaton H is a map $S: L(H) \to \Gamma$, where $S(s)$ represents the set of enabled events after the occurrence of the string $s \in L(H)$; i.e., a supervisor can disable controllable events only. The automaton S/H denotes an automaton H under supervision by S. The *closed-loop* language $L(S/H)$ generated by S/H is iteratively defined by (1) $\varepsilon \in L(S/H)$ and (2) $s\sigma \in L(S/H)$ iff $s \in L(S/H)$, $\sigma \in S(s)$ and $s\sigma \in L(H)$. To take into account the marked strings of S/H, let

$M \subseteq L_m(H)$ be the marking action of the supervisor S. Then, $L_m(S/H) := L(S/H) \cap M$, and S is denoted a *marking supervisor*. The closed-loop system is *nonblocking* if $\overline{L_m(S/H)} = L(S/H)$.

A language $K \subseteq \Sigma^*$ is controllable w.r.t. $L(H)$ and the uncontrollable events $\Sigma_{\rm uc} \subseteq \Sigma$ if $\overline{K}\Sigma_{\rm uc} \cap L(H) \subseteq \overline{K}$. The set of all controllable sublanguages w.r.t. $L(H)$ and $\Sigma_{\rm uc}$ is denoted as $\mathcal{C}(L(H)) = \{K \subseteq L(H) | \overline{K}\Sigma_{\rm uc} \cap L(H) \subseteq \overline{K}\}$. Since $\mathcal{C}(L(H))$ is closed under arbitrary union [19], for every *specification* language E, there uniquely exists a *supremal controllable sublanguage* of E w.r.t. $L(H)$ and $\Sigma_{\rm uc}$. It is formally defined as $\kappa_{L(H),\Sigma_{\rm uc}}(E) := \cup\{K \in \mathcal{C}(L(H)) | K \subseteq E\}$. A supervisor S that leads to $L_m(S/H) = \kappa_{L(H),\Sigma_{\rm uc}}(E)$ is said to be *maximally permissive*. It holds that a marking supervisor S such that $L_m(S/H) = \kappa_{L(H),\Sigma_{\rm uc}}(E \| L_m(H))$ exists whenever $\kappa_{L(H),\Sigma_{\rm uc}}(E \| L_m(H)) \neq \emptyset$ [19].

B. Set Theory

We present basic results from set theory as employed in [14]–[16]. We denote $\mathcal{E}(M)$ the set of all *equivalence relations* on the set M. For $\mu \in \mathcal{E}(M)$, $[m]_\mu$ is the *equivalence class* containing $m \in M$. The set of equivalence classes of μ is written as $M/\mu := \{[m]_\mu | m \in M\}$ and the *canonical projection* $\mathrm{cp}_\mu : M \to M/\mu$ maps an element $m \in M$ to its equivalence class $[m]_\mu$. Let $f : M \to N$ be a function. The equivalence relation $\ker f$ is the *kernel* of f and is defined as follows: for $m, m' \in M$

$$m \equiv m' \mod \ker f \quad \Leftrightarrow \quad f(m) = f(m'). \tag{1}$$

Given two equivalence relations η and μ on M, $\mu \leq \eta$, i.e., μ refines η, if $m \equiv m' \mathrm{mod}\mu \Rightarrow m \equiv m' \mathrm{mod}\eta$ for all m, $m' \in M$. In addition, we define the *meet* operation $\wedge$ for $\mathcal{E}(M)$ as follows. For any two elements $\mu, \eta \in \mathcal{E}(M)$, it holds for all $m, m' \in M$ that

$$m \equiv m' \mathrm{mod}(\mu \wedge \eta) \Leftrightarrow m \equiv m' \mathrm{mod}\mu \text{ and } m \equiv m' \mod \eta. \tag{2}$$

Let M and N be sets and $f : M \to 2^N$ be a set-valued function. It is also assumed that $\varphi \in \mathcal{E}(N)$, and the canonical projection cp_φ is naturally extended to sets. The equivalence relation $\varphi \circ f$ on M is defined for $m, m' \in M$ by

$$m \equiv m' \mod \varphi \circ f \Leftrightarrow \mathrm{cp}_\varphi(f(m)) = \mathrm{cp}_\varphi(f(m')). \tag{3}$$

Now let $f_i : M \to 2^M$ be functions, where i ranges over an index set $\mathcal{I}$. Then $S := (M, \{f_i | i \in \mathcal{I}\})$ is called a *dynamic system*[16]. The equivalence relation $\varphi \in \mathcal{E}(M)$ is called a *quasi-congruence* for S if

$$\varphi \leq \bigwedge_{i \in \mathcal{I}} (\varphi \circ f_i). \tag{4}$$

Finally, we introduce the (nondeterministic) *quotient automaton* (QA) $H_{\mu,\Sigma_0} = (Y, \Sigma_0 \cup \{\sigma_0\}, \nu, y_0, Y_m)$ of an automaton $H = (X, \Sigma, \delta, x_0, X_m)$ for an equivalence relation $\mu \in \mathcal{E}(X)$ and an alphabet $\Sigma_0 \subseteq \Sigma$ as in [16]. It holds that $Y := X/\mu$ is the *quotient set* with the associated *canonical projection* $\mathrm{cp}_\mu : X \to Y$. The initial state and the marked states in the QA are $y_0 = \mathrm{cp}_\mu(x_0)$ and $Y_m = \mathrm{cp}_\mu(X_m)$, respectively.

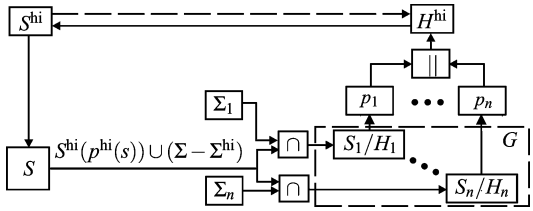

Fig. 1. Hierarchical and decentralized control architecture.

Also $\sigma_0 \notin \Sigma$ is an additional label. The nondeterministic *induced transition function* $\nu : Y \times (\Sigma_0 \cup \{\sigma_0\}) \to 2^Y$ of H_{μ,Σ_0} is defined as

$$\nu(y,\sigma) := \begin{cases} \{\mathrm{cp}_\mu(\delta(x,\sigma)) | x \in \mathrm{cp}_\mu^{-1}(y)\} & \text{if } \sigma \in \Sigma_0 \\ \{\mathrm{cp}_\mu(\delta(x,\gamma)) | \gamma \in (\Sigma - \Sigma_0), \\ x \in \mathrm{cp}_\mu^{-1}(y)\} - \{y\} & \text{if } \sigma = \sigma_0. \end{cases}$$

III. Nonblocking and Maximally Permissive Hierarchical Control

In this section, several previous results on the nonblocking and maximally permissive control are revisited in the scope of a hierarchical and decentralized control framework. In particular, a discussion of the existing results in Section III-F motivates the unified approach developed in this paper.

A. Hierarchical and Decentralized Control Architecture

Our work is based on the hierarchical and decentralized control architecture introduced in [21]. Variations of this architecture can also be found in [8], [10], [17], [22]. The representation chosen in this paper is that of [8], [17].

It is assumed that the DES *plant* is described by a set of plant *components* modeled by automata $H_1, \ldots, H_n$ over the respective alphabets $\Sigma_1, \ldots, \Sigma_n$. Each alphabet Σ_i, $i = 1, \ldots, n$ consists of the controllable events $\Sigma_{i,\rm c}$ and uncontrollable events $\Sigma_{i,\rm uc}$ such that $\Sigma_i = \Sigma_{i,\rm c} \dot\cup \Sigma_{i,\rm uc}$. Each plant component H_i, $i = 1, \ldots, n$ can share events with other components. This set of *shared events* is defined as $\Sigma_{i,\cap} := \bigcup_{j=1, j\neq i}^n (\Sigma_i \cap \Sigma_j)$, and it is assumed that all plant components agree on the controllability status of their shared events, i.e., for all $i, j = 1, \ldots, n$, it holds that $\Sigma_{i,\rm uc} \cap \Sigma_{j,\rm c} = \emptyset$. Then, the overall plant H is given by $H := \|_{i=1}^n H_i$ with the alphabet $\Sigma := \bigcup_{i=1}^n \Sigma_i$, the set of controllable events $\Sigma_{\rm c} = \bigcup_{i=1}^n \Sigma_{i,\rm c}$, the set of uncontrollable events $\Sigma_{\rm uc} := \bigcup_{i=1}^n \Sigma_{i,\rm uc}$, and the set of shared events $\Sigma_\cap := \bigcup_{i=1}^n \Sigma_{i,\cap}$.

Our supervisor synthesis is based on the practical assumption that the specification $K \subseteq \Sigma^*$ is given by local specifications $K_i \subseteq \Sigma_i^*$ for the plant components and a global specification $\hat{K} \subseteq \hat{\Sigma}^* \subseteq \Sigma^*$ such that

$$K = \hat{K} \| (\|_{i=1}^n K_i). \tag{5}$$

Then, the supervisor synthesis is performed as follows. First, *local supervisors* $S_i : L(H_i) \to \Gamma_i$ are computed such that $L_m(S_i/H_i) = \kappa_{L(H_i),\Sigma_{i,\rm uc}}(K_i \| L_m(H_i))$. For convenience, we write $G_i := S_i/H_i$ for $i = 1, \ldots, n$. The overall locally controlled plant is characterized by $G := \|_{i=1}^n S_i/H_i$ as in Fig. 1.

Since the global specification $\hat{K} \subseteq \hat{\Sigma}^* \subseteq \Sigma^*$ is given over a subset of the overall alphabet Σ, an abstracted plant model $H^{\rm hi}$

is employed in the next synthesis step. For each $i = 1, \ldots, n$, the abstraction alphabet Σ_i^{hi} is defined such that it contains all events that are either shared with other components or the specification, i.e., $\Sigma_{i,\cap} \cup (\Sigma_i \cap \hat{\Sigma}) \subseteq \Sigma_i^{\mathrm{hi}} \subseteq \Sigma_i$. Defining the high-level alphabet $\Sigma^{\mathrm{hi}} := \bigcup_{i=1}^n \Sigma_i^{\mathrm{hi}}$ and the natural projections $p_i : \Sigma_i^* \to \left(\Sigma_i^{\mathrm{hi}}\right)^*$ for $i = 1, \ldots, n$ and $p^{\mathrm{hi}} : \Sigma^* \to (\Sigma^{\mathrm{hi}})^*$, the high-level plant H^{hi} is computed such that

$$\begin{aligned} L(H^{\mathrm{hi}}) &= p^{\mathrm{hi}}(L(G)) = \|_{i=1}^n p_i(L(G_i)) \\ L_m(H^{\mathrm{hi}}) &= p^{\mathrm{hi}}(L_m(G)) = \|_{i=1}^n p_i(L_m(G_i)). \end{aligned} \tag{6}$$

This procedure can be seen on the right-hand side of Fig. 1. In the presented approach, the high-level controllable and uncontrollable events are chosen 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. The *high-level supervisor* $S^{\mathrm{hi}} : L(H^{\mathrm{hi}}) \to \Gamma^{\mathrm{hi}}$ is thus defined with $L_m(S^{\mathrm{hi}}/H^{\mathrm{hi}}) = \kappa_{L(H^{\mathrm{hi}}),\Sigma_{\mathrm{uc}}^{\mathrm{hi}}}(\hat{K} \| L_m(H^{\mathrm{hi}}))$. The resulting *low-level supervisor* $S : L(G) \to \Gamma$ is determined from S^{hi} by defining $S(s) := S^{\mathrm{hi}}(p^{\mathrm{hi}}(s)) \cup (\Sigma - \Sigma^{\mathrm{hi}})$ for each $s \in L(G)$ as can be seen on the left-hand side of Fig. 1. The local set of enabled events for each component G_i, $i = 1, \ldots, n$ is hence $S(s) \cap \Sigma_i$. Then, the overall closed-loop is described by

$$S/G = S^{\mathrm{hi}}/H^{\mathrm{hi}} \left\| G = S^{\mathrm{hi}}/H^{\mathrm{hi}} \right\| (\|_{i=1}^n S_i/H_i) \tag{7}$$

and we denote our control architecture as nonblocking if

$$\overline{L_m(S/G)} = L(S/G). \tag{8}$$

A further crucial point in hierarchical supervisory control is *maximal permissiveness*. It ensures that in spite of the information aggregation in the abstraction process, the optimality of the control is not lost, i.e., the hierarchical supervisor yields the same closed-loop behavior as a monolithic supervisor. It has to be noted that finding a nonblocking supervisor that fulfills the given specification might already be sufficient in many practical large-scale applications as long as the closed-loop behavior is not empty. However, in case maximal permissiveness is not guaranteed, it is possible that the closed-loop behavior is empty, while maximally permissive control would result in a non-empty closed loop.

We denote our control architecture as maximally permissive if the closed-loop behavior under the joint control action of the supervisors $S^{\mathrm{hi}}, S_1, \ldots, S_n$ defined in (7) is equal to the closed-loop behavior under monolithic supervisory control

$$L_m(S/G) = \kappa_{L(H),\Sigma_{\mathrm{uc}}}(K \| L_m(H)). \tag{9}$$

B. Nonblocking Hierarchical and Decentralized Control

In this section, two alternative conditions on natural projections are employed to achieve nonblocking control in the architecture in Section III-A.

1) Natural Observer: First, the natural observer is defined.

Definition 3.1 (Natural Observer [2]): Let $L \subseteq \Sigma^*$ be a language, and let $p_0 : \Sigma^* \to \Sigma_0^*$ be the natural projection for $\Sigma_0 \subseteq \Sigma$. p_0 is an L-observer iff for all $s \in \overline{L}$ and $t \in \Sigma_0^*$

$$\begin{aligned} &p_0(s)t \in p_0(L) \Rightarrow \exists u \in \Sigma^* \\ &\text{s.t. } su \in L \wedge p_0(su) = p_0(s)t. \end{aligned}$$

In words, p_0 is an L-observer if any string $s \in \overline{L}$ can be extended to a string in L whenever its projection $p_0(s)$ can be extended to a string in $p_0(L)$. It is stated in [10] [Proposition 7] that the $L_m(H_i)$-observer condition for p_i, $i = 1, \ldots, n$, yields nonblocking control in the described control architecture.

Theorem 3.1 (Natural Observer [10]): The control architecture in Section III-A is nonblocking if p_i is an $L_m(S_i/H_i)$-observer for $i = 1, \ldots, n$.

2) MSA-Observer: Second, the marked string accepting (msa)-observer condition is taken into account.

Definition 3.2 (MSA-Observer [14]): Let $L \subseteq \Sigma^*$ be a language and let $p_0 : \Sigma^* \to \Sigma_0^*$ be the natural projection for $\Sigma_0 \subseteq \Sigma$. p_0 is an msa-observer (w.r.t. L) iff p_0 is an $\overline{L}$-observer and for all strings $s \in \overline{L}$ such that $s\Sigma_0 \cap \overline{L} \neq \emptyset$ and $p_0(s) \in p_0(L)$

$$\exists s' \leq s \text{ s.t. } p_0(s') = p_0(s) \text{ and } s' \in L. \tag{10}$$

This means that on the one hand, the natural observer condition has to be fulfilled for the closed language $\overline{L}$. On the other hand, (10) concerns strings $s \in \overline{L}$ that can be extended by an event $\sigma \in \Sigma_0$ and such that the projection $p_0(s)$ is an element of $p_0(L)$. For such strings, it must hold that there is a prefix $s' \leq s$ with the same projection, i.e., $p_0(s') = p_0(s)$ and such that $s' \in L$ in order to ensure that whenever a string in the projected language $p_0(L)$ is passed, each corresponding string in $\overline{L}$ passes a string in L.

In addition, we define the notion of *liveness* of a language w.r.t. a given alphabet as follows.

Definition 3.3 (Liveness): Let $L \subseteq \Sigma^*$ be a prefix-closed language and $\Sigma_0 \subseteq \Sigma$. L is live w.r.t. Σ_0 if $\forall\, s \in L$

$$s(\Sigma - \Sigma_0)^* \Sigma_0 \cap L \neq \emptyset. \tag{11}$$

That is, L is live w.r.t. the alphabet Σ_0 if all of its strings can be extended to an event in $\Sigma_0 \subseteq \Sigma$.[1]

Combining the msa-observer condition and liveness, the following theorem constitutes a variation of the main result in [8] with high practical relevance (see also [23] [Theorem 4.1]): Liveness of $L(S^{\mathrm{hi}}/H^{\mathrm{hi}})$ w.r.t. all component alphabets Σ_i^{hi} ensures that no plant component can completely refrain from interacting with the other plant components.

Theorem 3.2 (MSA-Observer): The control architecture in Section III-A is nonblocking if p_i is an msa-observer w.r.t. $L_m(S_i/H_i)$ and $L(S^{\mathrm{hi}}/H^{\mathrm{hi}})$ is live w.r.t. Σ_i^{hi} for $i = 1, \ldots, n$.

Remark 3.1: Note that, as stated in [8], the conditions in Definition 3.1 and 3.2 are incomparable. This fact can be seen in Fig. 2 considering the automata H_1 and H_1' over the alphabet $\Sigma = \{\alpha, \beta, \mathtt{a}, \mathtt{b}\}$. Here, the projection $p : \Sigma^* \to \Sigma_0^*$ with $\Sigma_0 = \{\alpha, \beta\}$ is an $L_m(H_1)$-observer but not an msa-observer w.r.t. $L_m(H_1)$, whereas p is an msa-observer w.r.t. $L_m(H_1')$ but no $L_m(H_1')$-observer. Furthermore, it has to be clarified that liveness in Theorem 3.2 does not constitute a severe restriction. It simply states that each system component should always be able to at least generate some of its shared events in the closed loop, which is a natural requirement in practice (otherwise the component stops its participation in the system operation). For

[1]Note that this definition of liveness is stronger than the usual definition where $\Sigma_0 = \Sigma$.

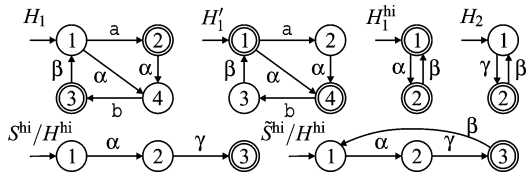

Fig. 2. Comparison of natural and msa-observers.

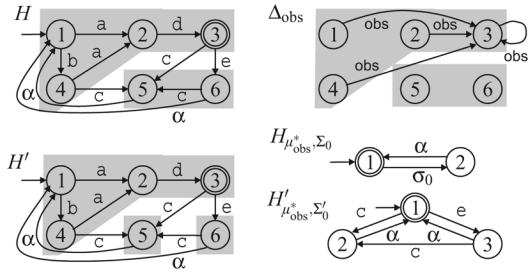

Fig. 3. Verification of the $L_m(H)$- and $L_m(H')$-observer condition.

example, we consider the system that is composed of the components $H_1\,(H_1')$ and H_2 in Fig. 2 with the abstractions H_1^{hi} and $H_2^{\mathrm{hi}} = H_2$ and the shared event β. Assuming the high-level closed loop $S^{\mathrm{hi}}/H^{\mathrm{hi}}\left(H^{\mathrm{hi}} = H_1^{\mathrm{hi}} \| H_2^{\mathrm{hi}}\right)$, nonblocking behavior is achieved for the system with H_1, whereas the system with H_1' is blocking since liveness of $L(S^{\mathrm{hi}}/H^{\mathrm{hi}})$ w.r.t. Σ_0 is violated. However, also the nonblocking case with H_1 is undesirable in practice since that component terminates its operation after the occurrence of α. In contrast, the desirable case would be the closed loop $\tilde{S}^{\mathrm{hi}}/H^{\mathrm{hi}}$ that is live w.r.t. Σ_0 and hence ensures nonblocking behavior for both H_1 and H_1'. Together, it depends on each particular supervisory control problem which of the above conditions is preferable. Section VI-A shows a practical example where the msa-observer condition is beneficial.

C. Verification of Observer Conditions

Regarding the sufficient conditions for nonblocking hierarchical and decentralized control in Theorem 3.1 and 3.2, a problem of great interest is to algorithmically verify if given natural projections fulfill the respective conditions. To this end, approaches that are based on the computation of quasi-congruences for particular dynamic systems have been developed for natural observers ([16]) and msa-observers ([14]).

1) Natural Observer: Let $H = (X, \Sigma, \delta, x_0, X_m)$ be a nonblocking automaton[2], and let $\Sigma_0 \subseteq \Sigma$ with the natural projection $p_0 : \Sigma^* \to \Sigma_0^*$. Then, the dynamic system $\tilde{H}_{\mathrm{obs}} = (X, \{\Delta_\sigma | \sigma \in \Sigma_0\} \cup \Delta_{\mathrm{obs}})$ is defined with

$$\Delta_\sigma : X \to 2^X : x \to \{\delta(x, u\sigma u') | uu' \in (\Sigma - \Sigma_0)^*\},$$
$$\Delta_{\mathrm{obs}} : X \to 2^{X_m} : x \to \{\delta(x,u) \in X_m | u \in (\Sigma - \Sigma_0)^*\}.$$

According to [16], the coarsest quasi-congruence $\mu^*_{\mathrm{obs}} \in \mathcal{E}(X)$ for $\tilde{H}_{\mathrm{obs}}$ exists and can be computed with the algorithm in [24] with a complexity of $\mathcal{O}\left(N_X^3 \cdot N_T\right)$, where N_X and N_T denote the number of states and transitions of H, respectively. With H, Σ_0 and μ^*_{obs}, the observer condition in Definition 3.1 can be verified by means of the QA $H_{\mu^*_{\mathrm{obs}},\Sigma_0}$.

Theorem 3.3 (Observer Verification [16]): The projection p_0 is an $L_m(H)$-observer iff $H_{\mu^*_{\mathrm{obs}},\Sigma_0}$ is deterministic and contains no σ_0-transitions.

Example 3.1: To illustrate Theorem 3.3, we first investigate H in Fig. 3 with the alphabet $\Sigma_0 = \{\alpha\}$. The corresponding dynamic system $\tilde{H}_{\mathrm{obs}} = (X, \{\Delta_\alpha, \Delta_{\mathrm{obs}}\})$ fulfills $\Delta_{\mathrm{obs}}(1) = \Delta_{\mathrm{obs}}(2) = \Delta_{\mathrm{obs}}(3) = \Delta_{\mathrm{obs}}(4) = \{3\}$, $\Delta_{\mathrm{obs}}(5) = \Delta_{\mathrm{obs}}(6) = \emptyset$ and $\Delta_\alpha(x) = X$ for all states $x \in X$. Δ_{obs} is depicted in Fig. 3 by transitions with the label obs. The coarsest quasi-congruence μ^*_{obs} on $\tilde{H}_{\mathrm{obs}}$ is indicated by the shaded areas in Fig. 3, i.e.

$$\mu^*_{\mathrm{obs}} = \sup\{\mu \in \mathcal{E}(X) | \mu \leq (\mu \circ \Delta_\alpha) \wedge (\mu \circ \Delta_{\mathrm{obs}})\}.$$

Since the QA $H_{\mu^*_{\mathrm{obs}},\Sigma_0}$ has a σ_0-transition, the projection p_0 with $\Sigma_0 = \{\alpha\}$ is not an $L_m(H)$-observer.

Choosing $\Sigma_0' = \{\alpha, \mathtt{c}, \mathtt{e}\}$, the evaluation of μ^*_{obs} as shown in the lower part of Fig. 3.3 for H' suggests that the associated projection $p_0' : \Sigma^* \to {\Sigma_0'}^*$ is an $L_m(H')$-observer, since the QA $H'_{\mu^*_{\mathrm{obs}},\Sigma_0'}$ is deterministic and has no σ_0-transitions.

2) MSA-Observer: In a similar way, the msa-observer property can be formulated in terms of a quasi-congruence. Here, the dynamic system $\tilde{H}_{\mathrm{msa}} = (X, \{\Delta_\sigma | \sigma \in \Sigma_0\} \cup \Delta_{\mathrm{msa}})$ for a nonblocking automaton $H = (X, \Sigma, \delta, x_0, X_m)$ with the projection alphabet Σ_0 is defined to address the condition in Definition 3.2. Δ_σ is defined as above. In order to introduce $\Delta_{\mathrm{msa}} : X \to 2^X$ as in [14], we first write $X_0 := \{x_0\} \cup \{x \in X | x = \delta(x_0, s)$ for $s \in \Sigma^*\Sigma_0\}$ for the set of states reachable immediately after an event in the abstraction alphabet including the initial state. Then, it holds that

$$\Delta_{\mathrm{msa}}(x) := \begin{cases} \bigcup\limits_{\sigma \in \Sigma_0} \Delta_\sigma(x) & \text{if } \exists x' \in X_0, x'' \in X, \\ & u \in (\Sigma - \Sigma_0)^*, \\ & v \in (\Sigma - \Sigma_0)^*\Sigma_0 \\ & \text{s.t. } x = \delta(x', u),\ x'' = \delta(x, v) \\ & \text{and for all } u' < uv \\ & \text{it holds that } \delta(x', u') \notin X_m \\ \emptyset & \text{otherwise.} \end{cases}$$

Hence, Δ_{msa} maps a state $x \in X$ to all states in $\bigcup_{\sigma \in \Sigma_0} \Delta_\sigma(x)$ if for a string $s \in \Sigma^*$ with $x = \delta(x_0, s)$ one of the following alternatives hold: (1) s violates the condition in (10), or (2) there is no string $s' \in \Sigma^*$ such that $p_0(s') = p_0(s)$ and $s' \in p_0(L_m(H))$, with $p_0 : \Sigma^* \to \Sigma_0^*$. The following theorem cites a result from [14] [Theorem 4.2]. Computing the coarsest quasi-congruence μ^*_{msa} for $\tilde{H}_{\mathrm{msa}}$ with the algorithm in [24], the resulting QA $H_{\mu^*_{\mathrm{msa}},\Sigma_0}$ can be used to verify the msa-observer condition in polynomial time $\left(\mathcal{O}\left(N_X^3 \cdot N_T\right)\right)$.

Theorem 3.4 (MSA-Observer Verification [14]): The projection p_0 is an msa-observer w.r.t. $L_m(H)$ iff $H_{\mu^*_{\mathrm{msa}},\Sigma_0}$ is deterministic and contains no σ_0-transitions.

Example 3.2: Theorem 3.4 is further explained in Fig. 4. With the automaton H and the alphabet $\Sigma_0 = \{\alpha\}$, the dynamic system is $\tilde{H}_{\mathrm{msa}} = (X, \{\Delta_\alpha, \Delta_{\mathrm{msa}}\})$ with $\Delta_\alpha(x) = X$ for all $x \in X$ and $\Delta_{\mathrm{msa}}(x) = X$ for $x \in \{1, 4, 5\}$, while $\Delta_{\mathrm{msa}}(x) = \emptyset$ for $x \in \{2, 3, 6\}$. The coarsest quasi-congruence μ^*_{msa} is again

[2] In the sequel, we consider H as the canonical recognizer of $L_m(H)$.

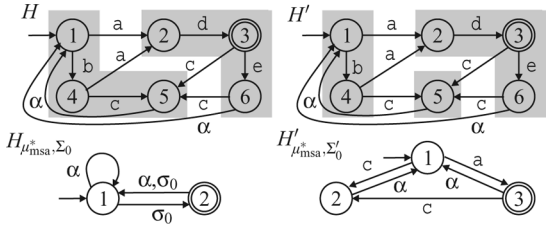

Fig. 4. Verification of the msa-observer condition.

depicted by the shaded areas in H. The computation of the QA $H_{\mu^*_{\mathrm{msa}},\Sigma_0}$ shows that p_0 with $\Sigma_0 = \{\alpha\}$ is not an msa-observer.

Conversely, the analogous discussion for the automaton H' and the alphabet $\Sigma'_0 = \{\alpha, \mathtt{a}, \mathtt{c}\}$ with the QA $H'_{\mu^*_{\mathrm{msa}},\Sigma'_0}$ in Fig. 4 suggests that $p'_0 : \Sigma^* \to {\Sigma'_0}^*$ is an msa-observer.

D. Computation of Natural Observers

In practice, it is not only interesting to verify if the conditions in the previous section are fulfilled but it is also relevant to address the case where the verification fails. Then, it is desired to find a minimal extension of a given projection alphabet such that the sufficient conditions for nonblocking hierarchical and decentralized control hold. This *event set extension* problem is studied in [15] for the natural observer condition. It is first noted that finding a minimal extension is NP-hard. Then, the following polynomial-time algorithm that computes acceptable extensions is proposed.

The algorithm is based on the computation of the QA $H_{\mu^*_{\mathrm{obs}},\Sigma_0}$ that is performed in step 1. Then, either the verification of the natural observer condition according to Theorem 3.1 is successful and the current alphabet Σ_0 is returned, or an extension of Σ_0 is required. In the latter case, the event set extension algorithm in [15] is applied. It adds events to Σ_0 in order to remove σ_0-transitions and resolve the possible nondeterminism in $H_{\mu^*_{\mathrm{obs}},\Sigma_0}$. The observer algorithm iterates until an appropriate alphabet extension is found. Its complexity, which is $\mathcal{O}\left(N_X^4 \cdot N_T^3\right)$, is dominated by that of the event set extension.

Note that there is currently no analogous result that addresses the event set extension for the msa-observer condition.

E. Conditions for Maximally Permissive Control

In the literature, there is one result in [10] that employs *output control consistency* (OCC) as introduced in [1] as a sufficient condition for maximal permissiveness in the control architecture in Section III-A.

Definition 3.4 (OCC): Let H be an automaton, let $\Sigma_{\mathrm{uc}} \subseteq \Sigma$ be a set of uncontrollable events, and let $\Sigma_0 \subseteq \Sigma$. The natural projection $p_0 : \Sigma^* \to \Sigma_0^*$ is output control consistent (occ) w.r.t. $L(H)$ and Σ_{uc} if for every $s \in L(H)$ of the form

$$s = \sigma_1 \cdots \sigma_k \text{ or } s = s'\sigma_0\sigma_1 \cdots \sigma_k, \quad k \geq 1$$

where σ_0, $\sigma_k \in \Sigma_0$ and $\sigma_i \in \Sigma - \Sigma_0$ for $i = 1, \ldots, k-1$, we have the property that $\sigma_k \in \Sigma_{\mathrm{uc}} \Rightarrow (\forall\, i = 1, \ldots, k)\sigma_i \in \Sigma_{\mathrm{uc}}$.

This means that, whenever σ_k is an uncontrollable event in $\Sigma_{\mathrm{uc}} \cap \Sigma_0$, its immediately preceding events in $\Sigma - \Sigma_0$ must all be uncontrollable, such that its nearest controllable event is an element of Σ_0.

With the additional assumption that all plant components do not share any events, [10] states the following result.

Theorem 3.5: The control architecture in Section III-A is nonblocking and maximally permissive if $\Sigma_{\cap,i} = \emptyset$ and p_i is an $L_m(S_i/H_i)$-observer and occ w.r.t. $L(S_i/H_i)$ and $\Sigma_{i,\mathrm{uc}}$, for $i = 1, \ldots, n$.

[10] also suggests an algorithm with complexity $\mathcal{O}\left(N_X^2\right)$ in order to modify a given natural projection to be occ.

F. Discussion of Existing Results

In summary, the state of the research concerning the hierarchical and decentralized architecture described in Section III-A offers sufficient conditions for the nonblocking supervisory control that can be verified in polynomial time. However, the algorithmic computation of natural projections for nonblocking control is only applicable to the natural observer condition, and maximally permissive control can only be verified for the restrictive case where the plant components do not share events. Moreover, the computation of natural projections for nonblocking and maximally permissive hierarchical and decentralized control currently relies on the iterative application of the observer extension algorithm in Section III-D and the algorithm for OCC as indicated in Section III-E.

In the subsequent section, we identify *local control consistency* (LCC) of natural projections as a sufficient condition for maximally permissive control that is suitable for our control architecture and less restrictive than OCC. Furthermore, we show that the assumption of *mutual controllability* allows the maximally permissive control for the case where plant components share events. Then, we investigate the algorithmic computation of appropriate natural projections for nonblocking and maximally permissive hierarchical and decentralized control in Section V. We first deduce a *generalized observer extension algorithm* from Algorithm 1 in order to compute natural projections that are msa-observers. Moreover, we derive an appropriate formulation of LCC that allows to use the generalized observer extension algorithm for the computation of natural projections that are locally control consistent.

Algorithm 1 (Observer Extension): **Input:** H, Σ_0

1. Compute the quasi-congruence μ^*_{obs} and the QA $H_{\mu^*_{\mathrm{obs}},\Sigma_0}$.
2. **if** $H_{\mu^*_{\mathrm{obs}},\Sigma_0}$ is deterministic and contains no σ_0-transitions

   return Σ_0

   else

   event set extension of Σ_0 as in **[15]** and **go to** 1.

IV. Maximally Permissive Control

The goal of this section is the development of conditions that are sufficient for nonblocking and maximally permissive control in conjunction with the existing results on nonblocking control in Theorem 3.1 and 3.2. As opposed to the previous work in [10], it is desired that the requirement of mutually disjoint alphabets

of the plant components H_i, $i = 1, \ldots, n$ in Theorem 3.5 is relaxed.

In principle, it has to be considered that the fulfillment of (9) both depends on the plant abstractions based on the projections p_i, $i = 1, \ldots, n$ and on the specification $K = \hat{K} || (||_{i=1}^{n} K_i)$ as in (5). In this paper, we intend to determine p_i, $i = 1, \ldots, n$ such that (9) holds for all possible specifications. In order to achieve a unified treatment, instead of deriving extensions to the conditions in Theorem 3.1 and 3.2 separately, our investigation is based on two common properties: The control architecture is nonblocking and all projections p_i, $i = 1, \ldots, n$ are $L(G_i)$-observers. Now we address the following problem.

Problem 1: Assume that the control architecture in Section III-A is nonblocking and that p_i is an $L(G_i)$-observer for $i = 1, \ldots, n$. We want to find sufficient conditions for the natural projections p_i, $i = 1, \ldots, n$ such that the control architecture is also maximally permissive.

A. Conditions for Maximally Permissive Control

We first state a condition that supports the stepwise solution of Problem 1. It holds that the control architecture in Section III-A is maximally permissive if the projection of the closed-loop language in the case of monolithic supervisory control is controllable w.r.t. the closed language of the high-level plant.

Proposition 4.1: Assume that the control architecture in Section III-A is nonblocking, and define $p^{\mathrm{hi}} : \Sigma^* \to (\Sigma^{\mathrm{hi}})^*$. Also let $\hat{K} \subseteq \hat{\Sigma}^* \subseteq \Sigma^*$ and write $\hat{K}_{\sup} := \kappa_{L(G),\Sigma_{\mathrm{uc}}}(\hat{K} || L_m(G))$. If $p^{\mathrm{hi}}(\hat{K}_{\sup})$ is controllable w.r.t. $L(H^{\mathrm{hi}})$ and $\Sigma_{\mathrm{uc}}^{\mathrm{hi}}$, then the control architecture is maximally permissive.

Proof: According to (9), it has to be shown that $L_m(S/G) = \hat{K}_{\sup}$. As $L(S/G)$ is controllable w.r.t. $L(G)$, the fact that S is nonblocking establishes that $L_m(S/G)$ is controllable w.r.t. $L(G)$. Together with $L_m(S/G) \subseteq L_m(G) || \hat{K}$, this implies that $L_m(S/G) \subseteq \hat{K}_{\sup}$. To show the reverse inclusion, we observe that $p^{\mathrm{hi}}(\hat{K}_{\sup}) || L_m(G) \subseteq L_m(S^{\mathrm{hi}}/H^{\mathrm{hi}}) || L_m(G)$ as $p^{\mathrm{hi}}(\hat{K}_{\sup})$ is controllable w.r.t. $L(H^{\mathrm{hi}})$ and $\Sigma_{\mathrm{uc}}^{\mathrm{hi}}$. Since $\hat{K}_{\sup} \subseteq (p^{\mathrm{hi}})^{-1}(p^{\mathrm{hi}}(\hat{K}_{\sup}))$ and $\hat{K}_{\sup} \subseteq L_m(G)$, also $\hat{K}_{\sup} \subseteq (p^{\mathrm{hi}})^{-1}(p^{\mathrm{hi}}(\hat{K}_{\sup})) \cap L_m(G) = p^{\mathrm{hi}}(\hat{K}_{\sup}) || L_m(G) \subseteq L_m(S^{\mathrm{hi}}/H^{\mathrm{hi}}) || L_m(G) = L_m(S/G)$. ∎

As a consequence, it must hold for the realization of the supremal controllable sublanguage $\hat{K}_{\sup}$ that the supervisor S only needs to disable controllable high-level events in $\Sigma_{\mathrm{c}}^{\mathrm{hi}}$. In the next definition, we introduce *local control consistency* (LCC) as a novel condition for the projection p^{hi} that implies this requirement as shown in the subsequent Lemma 4.1.

Definition 4.1 (LCC [17]): Let G be an automaton over the alphabet Σ, let $\Sigma_{\mathrm{uc}} \subseteq \Sigma$ be a set of uncontrollable events, and let $\Sigma_0 \subseteq \Sigma$. The natural projection $p_0 : \Sigma^* \to \Sigma_0^*$ is locally control consistent (lcc) w.r.t. a string $s \in L(G)$ and Σ_{uc} if for all $\sigma_{\mathrm{uc}} \in \Sigma_0 \cap \Sigma_{\mathrm{uc}}$ s.t. $p_0(s)\sigma_{\mathrm{uc}} \in p_0(L(G))$, it holds that either $\nexists u \in (\Sigma - \Sigma_0)^*$ s.t. $su\sigma_{\mathrm{uc}} \in L(G)$ or there is a $u \in (\Sigma_{\mathrm{uc}} - \Sigma_0)^*$ s.t. $su\sigma_{\mathrm{uc}} \in L(G)$. Furthermore, we call p_0 lcc w.r.t. a language $L' \subseteq L(G)$ and Σ_{uc} if p_0 is lcc for all $s \in L'$.

In words, a natural projection is locally control consistent w.r.t. a string $s \in L(G)$, if for each uncontrollable event $\sigma_{\mathrm{uc}} \in \Sigma_{\mathrm{uc}} \cap \Sigma^{\mathrm{hi}}$ that is feasible after the corresponding projected string, there is either no continuation or an uncontrollable continuation of s that terminates with σ_{uc}. Hence, if σ_{uc} is possible after s, then it cannot be prevented.

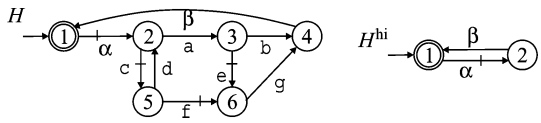

Fig. 5. Illustration of local control consistency.

Example 4.1: We illustrate LCC by the automaton H over the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \alpha, \beta\}$ in Fig. 5. Using the abstraction alphabet, $\Sigma^{\mathrm{hi}} = \{\alpha, \beta\}$ and the projection $p^{\mathrm{hi}} : \Sigma^* \to (\Sigma^{\mathrm{hi}})^*$, the high-level plant H^{hi} is obtained. Note that transitions with controllable events are marked by a tick, i.e., $\Sigma_{\mathrm{c}} = \{\mathtt{c}, \mathtt{e}, \mathtt{f}, \alpha\}$. Then it holds for all strings in $L(H^{\mathrm{hi}})$ that can be extended by the uncontrollable high-level event β that the corresponding low-level strings (leading to the states 2, 3, 4, 5 and 6) have at least one uncontrollable extension in $(\Sigma_{\mathrm{uc}} - \Sigma^{\mathrm{hi}})^*$ after which β is feasible. Hence, p^{hi} is lcc w.r.t. $L(H)$ and $\Sigma_{\mathrm{uc}} = \{\mathtt{a}, \mathtt{b}, \mathtt{d}, \mathtt{g}, \beta\}$.

Lemma 4.1: Assume that the control architecture in Section III-A is nonblocking and that p^{hi} is an $L(G)$-observer. Also let $\hat{K} \subseteq \hat{\Sigma}^* \subseteq \Sigma^*$ and write $\hat{K}_{\sup} := \kappa_{L(G),\Sigma_{\mathrm{uc}}}(\hat{K} || L_m(G))$. Then it holds that $p^{\mathrm{hi}}(\hat{K}_{\sup})$ is controllable w.r.t. $L(H^{\mathrm{hi}})$ and $\Sigma_{\mathrm{uc}}^{\mathrm{hi}}$ if p^{hi} is lcc w.r.t. $L(G)$ and Σ_{uc}.

Proof: We assume that p^{hi} is lcc w.r.t. $L(G)$ and Σ_{uc}. It has to be shown that $p^{\mathrm{hi}}(\hat{K}_{\sup})$ is controllable w.r.t. $\underline{L(H^{\mathrm{hi}})}$ and $\Sigma_{\mathrm{uc}}^{\mathrm{hi}}$. Assume the contrary, i.e., we have $t \in \underline{p^{\mathrm{hi}}(\hat{K}_{\sup})}$ and $\sigma_{\mathrm{uc}} \in \underline{\Sigma_{\mathrm{uc}}^{\mathrm{hi}}}$ s.t. $t\sigma_{\mathrm{uc}} \in L(H^{\mathrm{hi}})$ but $t\sigma_{\mathrm{uc}} \notin p^{\mathrm{hi}}(\hat{K}_{\sup})$. Since $t \in p^{\mathrm{hi}}(\hat{K}_{\sup})$, it follows that $(\Sigma^* \Sigma^{\mathrm{hi}} \cup \{\varepsilon\}) \cap (p^{\mathrm{hi}})^{-1}(t) \cap L(G) \neq \emptyset$. Considering that $t\sigma_{\mathrm{uc}} \in L(H^{\mathrm{hi}})$, and p^{hi} is an $L(G)$-observer, for all $s \in (\Sigma^* \Sigma^{\mathrm{hi}} \cup \{\varepsilon\}) \cap (p^{\mathrm{hi}})^{-1}(t) \cap L(G)$, there must be a $u' \in (\Sigma - \Sigma^{\mathrm{hi}})^*$ s.t. $su'\sigma_{\mathrm{uc}} \in L(G)$. Hence, Definition 4.1 implies that there is also a $u \in (\Sigma_{\mathrm{uc}} - \Sigma_0)^*$ s.t. $su\underline{\sigma_{\mathrm{uc}} \in} L(G)$ which contradicts the assumption that $t\sigma_{\mathrm{uc}} \notin p^{\mathrm{hi}}(\hat{K}_{\sup})$. ∎

The conditions for maximal permissiveness in our previous considerations rely on the fact that the locally controlled plant G is given explicitly. Since our architecture is designed to avoid the explicit evaluation of G, we now investigate how the presented results can be applied for the maximally permissive control in the decentralized case. The following theorem states that LCC is sufficient for maximal permissiveness if only the realization of the global specification $\hat{K}$ is taken into account.

Theorem 4.1: Assume that the control architecture in Section III-A is nonblocking and that p_i is an $L(G_i)$-observer for $i = 1, \ldots, n$. If $L(G_i)$ is lcc w.r.t. $L(G_i)$ and $\Sigma_{i,\mathrm{uc}}$ for all $i = 1, \ldots, n$, then $L_m(S/G) = \kappa_{L(G),\Sigma_{\mathrm{uc}}}(\hat{K} || L_m(G))$.

Proof: We verify the conditions in Lemma 4.1. With [10] [Proposition 5], $p^{\mathrm{hi}} : \Sigma^* \to (\Sigma^{\mathrm{hi}})^*$ is an $L(G)$-observer. Hence, we show that p^{hi} is lcc w.r.t. $L(G)$ and Σ_{uc}.

Let $t \in L(H^{\mathrm{hi}})$, $s \in L(G)$ and $\sigma_{\mathrm{uc}} \in \Sigma_{\mathrm{uc}}$ s.t. $t = p^{\mathrm{hi}}(s)$ and $t\sigma_{\mathrm{uc}} \in L(H^{\mathrm{hi}})$. Define $\theta_i : \Sigma^* \to \Sigma_i^*$ for $i = 1, \ldots, n$. Since $s \in L(G)$, $s_i := \theta_i(s) \in L(G_i)$ for all $i = 1, \ldots, n$. Furthermore, for all i s.t. $\sigma_{\mathrm{uc}} \in \Sigma_i$, $p_i(s_i)\sigma_{\mathrm{uc}} \in p_i(L(G_i)))$.

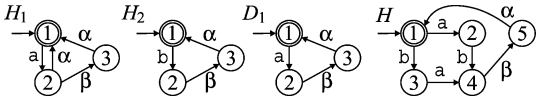

Fig. 6. Illustration of mutual controllability.

As p_i is an $L(G_i)$-observer and lcc w.r.t. $L(G_i)$ and $\Sigma_{i,\mathrm{uc}}$, for all i such that $\sigma_{\mathrm{uc}} \in \Sigma_i$, there exists a $u_i \in \left(\Sigma_{i,\mathrm{uc}} - \Sigma_{i,\mathrm{uc}}^{\mathrm{hi}}\right)^*$ s.t. $s_i u_i \sigma_{\mathrm{uc}} \in L(G_i)$. For all remaining i, let $u_i = \varepsilon$. Defining $u := u_1 \dots u_n$ and noting that $u_i \in (\Sigma_{i,\mathrm{uc}} - \Sigma_{i,\cap})^*$, we have $u \in \|_{i=1}^n \{u_i\}$. Hence, $su\sigma_{\mathrm{uc}} \in L(G) = \|_{i=1}^n L(G_i)$, and $u \in \left(\Sigma_{\mathrm{uc}} - \Sigma_{\mathrm{uc}}^{\mathrm{hi}}\right)^*$. Since t, s and σ_{uc} were arbitrary, it follows that p^{hi} is lcc w.r.t. $L(G)$ and Σ_{uc}. ∎

In order to achieve maximal permissiveness for the overall control architecture, the realization of the local specifications K_i, $i = 1, \dots, n$ also has to be considered. To this end, we investigate H_1 and H_2 in Fig. 6. Let $\Sigma_1^{\mathrm{hi}} = \Sigma_2^{\mathrm{hi}} = \{\alpha, \beta\}$, define $H := H_1 \| H_2$, and assume that all events are uncontrollable. Then, the local specification $K_1 \| L_m(H_1)$ with $K_1 := L_m(D_1)$ is not controllable w.r.t. $L(H_1)$, while $K_1 \| L_m(H)$ is controllable w.r.t. $L(H)$. Here, maximal permissiveness is violated since the local supervisor synthesis with K_1 and H_1 neglects the fact that α cannot occur after $\mathtt{a} \in L(H_1)$ because of the synchronization with H_2. This situation can be avoided if the plant components are *mutually controllable* [25], which ensures that after any string of a composed system, the occurrence of an uncontrollable shared event is either feasible in all components that share it or it is infeasible in any component.

Definition 4.2 (Mutual Controllability): Let H_i, H_k be automata and define the projections $p_{i,k} : (\Sigma_i \cup \Sigma_k)^* \to \Sigma_k^*$ and $p_{k,i} : (\Sigma_k \cup \Sigma_i)^* \to \Sigma_i^*$. H_i and H_k are mutually controllable if

$$L(H_i)(\Sigma_{k,\mathrm{uc}} \cap \Sigma_i) \cap p_{k,i}\left(p_{i,k}^{-1}(L(H_k))\right) \subseteq L(H_i),$$
$$L(H_k)(\Sigma_{i,\mathrm{uc}} \cap \Sigma_k) \cap p_{i,k}\left(p_{k,i}^{-1}(L(H_i))\right) \subseteq L(H_k).$$

Extending the conditions in Theorem 4.1 with mutual controllability is sufficient for maximally permissive control in our control architecture including local specifications.

Theorem 4.2 (Maximal Permissiveness): Problem 1 is solved, i.e., the control architecture in Section III-A is maximally permissive, if p_i is lcc w.r.t. $L(G_i)$ and $\Sigma_{i,\mathrm{uc}}$ for all $i = 1, \dots, n$, and all local components H_i, H_k, $i, k = 1, \dots, n$, $i \neq k$, are mutually controllable.

The proof of Theorem 4.2 relies on the following lemmas.

Lemma 4.2: Assume that automata H_i with the alphabets Σ_i and languages $K_i \subseteq \Sigma_i^*$, $i = 1, \dots, n$ are given. Furthermore, define $H := \|_{i=1}^n H_i$ and let H_i and H_k, $i, k = 1, \dots, n$, $i \neq k$, be mutually controllable. Then, $\kappa_{L(H),\Sigma_{\mathrm{uc}}}(L_m(H)\|K_1\| \cdots \|K_n) \subseteq \|_{i=1}^n \kappa_{L(H_i),\Sigma_{i,\mathrm{uc}}}(L_m(H_i)\|K_i) = L_m(G)$.

Lemma 4.3: Let H be an automaton over Σ, $\Sigma_{\mathrm{uc}} \subseteq \Sigma$ the uncontrollable event set, and $K_1, K_2 \subseteq \Sigma^*$ specifications. Then

$$\kappa_{L(H),\Sigma_{\mathrm{uc}}}(L_m(H)\|K_1\|K_2) = \kappa_{\overline{\kappa_{L(H),\Sigma_{\mathrm{uc}}}(L_m(H)\|K_1)},\Sigma_{\mathrm{uc}}}(\kappa_{L(H),\Sigma_{\mathrm{uc}}}(L_m(H)\|K_1)\|K_2).$$

Lemma 4.2 is shown in Appendix A, while Lemma 4.3 is stated in [19] [Exercise 3.7.13]. Now, Theorem 4.2 is proved.

Proof: Let $L_{\max} := \kappa_{L(H),\Sigma_{\mathrm{uc}}}(L_m(H)\|K_1\| \cdots \|K_n)$. We first show that $L_{\max} = \kappa_{L(G),\Sigma_{\mathrm{uc}}}(L_m(G))$. On the one hand, $\overline{L_{\max}}\Sigma_{\mathrm{uc}} \cap L(G) \subseteq \overline{L_{\max}}\Sigma_{\mathrm{uc}} \cap L(H) \subseteq \overline{L_{\max}}$. That is, $L_{\max}$ is controllable w.r.t. $L(G)$ and Σ_{uc}. According to Lemma 4.2, also $L_{\max} \subseteq L_m(G)$. With this $\kappa_{L(H),\Sigma_{\mathrm{uc}}}(L_m(H)\|K_1\| \cdots \|K_n) = L_{\max} \subseteq \kappa_{L(G),\Sigma_{\mathrm{uc}}}(L_m(G))$ follows.

To show the other inclusion let $s \in \kappa_{L(G),\Sigma_{\mathrm{uc}}}(L_m(G))$. It holds that $L_m(G) \subseteq (L_m(H)\|K_1\| \cdots \|K_n)$, and every sublanguage of $L_m(G)$ that is controllable w.r.t. $L(G)$ and Σ_{uc} is also controllable w.r.t. $L(H)$ and Σ_{uc}, since $L(G)$ is controllable w.r.t. $L(H)$ and Σ_{uc}. Hence, $\kappa_{L(G),\Sigma_{\mathrm{uc}}}(L_m(G)) = \kappa_{L(H),\Sigma_{\mathrm{uc}}}(L_m(G)) \subseteq \kappa_{L(H),\Sigma_{\mathrm{uc}}}(L_m(H)\|K_1\| \cdots \|K_n)$ such that $s \in \kappa_{L(H),\Sigma_{\mathrm{uc}}}(L_m(H)\|K_1\| \cdots \|K_n)$. Hence, Lemma 4.3 implies the formula shown at the bottom of the page. Now applying Lemma 4.3 and then Theorem 4.1, we obtain

$$\begin{aligned}
\kappa_{L(H),\Sigma_{\mathrm{uc}}}(K) &= \kappa_{\overline{\kappa_{L(G),\Sigma_{\mathrm{uc}}}(L_m(G))},\Sigma_{\mathrm{uc}}}(\kappa_{L(G),\Sigma_{\mathrm{uc}}}(L_m(G))\|\hat{K}) \\
&= \kappa_{L(G),\Sigma_{\mathrm{uc}}}(L_m(G)\|\hat{K}) = L_m(S/G) \\
&= L_m\left(\frac{S^{\mathrm{hi}}}{H^{\mathrm{hi}}}\right)\|L_m(G) \\
&= L_m\left(S^{\mathrm{hi}}/H^{\mathrm{hi}}\right)\|L_m(S_1/H_1)\| \\
&\cdots\|L_m(S_n/H_n).
\end{aligned}$$

∎

Finally, we combine the conditions in Section III-B and in this section to achieve nonblocking and maximally permissive hierarchical and decentralized supervisory control.

Corollary 4.1: The control architecture in Section III-A is nonblocking and maximally permissive if

1) p_i is lcc w.r.t. $L(G_i)$ and $\Sigma_{i,\mathrm{uc}}$ for $i = 1, \dots, n$;
2) H_i, H_k are mutually controllable for $i, k = 1, \dots, n$, $i \neq k$;
3) (a) p_i is an $L_m(S_i/G_i)$-observer for $i = 1, \dots, n$ or (b) p_i is an msa-observer w.r.t. $L_m(S_i/G_i)$ and $L(S^{\mathrm{hi}}/H^{\mathrm{hi}})$ is live w.r.t. Σ_i^{hi} for $i = 1, \dots, n$.

$$\begin{aligned}
\kappa_{L(H),\Sigma_{\mathrm{uc}}}(K) &= \kappa_{L(H),\Sigma_{\mathrm{uc}}}(L_m(H)\|\hat{K}\|K_1\| \cdots \|K_n) \\
&= \kappa_{\overline{\kappa_{L(H),\Sigma_{\mathrm{uc}}}(L_m(H)\|K_1\|\cdots\|K_n)},\Sigma_{\mathrm{uc}}}(\kappa_{L(H),\Sigma_{\mathrm{uc}}}(L_m(H)\|K_1\| \cdots \|K_n)\|\hat{K}) \\
&= \kappa_{\overline{\kappa_{L(G),\Sigma_{\mathrm{uc}}}(L_m(G))},\Sigma_{\mathrm{uc}}}(\kappa_{L(G),\Sigma_{\mathrm{uc}}}(L_m(G))\|\hat{K}).
\end{aligned}$$

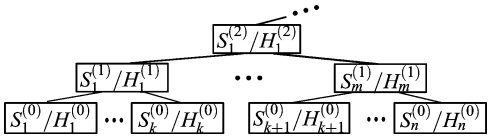

Fig. 7. Multi-level hierarchical and decentralized control architecture.

Proof: The proof of Corollary 4.1 follows from Theorem 4.2 considering that both 3.(a) and 3.(b) imply that S is nonblocking and p_i is an $L(G_i)$-observer for $i = 1, \ldots, n$. ∎

The results in this section are elaborated for a control architecture with two hierarchical levels. It is shown in [17] that the same conditions can be employed in a multi-level hierarchy as depicted in Fig. 7. Here, each high-level closed-loop $S_l^{(q)}/H_l^{(q)}$, $q > 0$ (q indicates the hierarchical level) can be used as a low-level plant for a further hierarchical synthesis.

B. Related Work

In order to relate our novel result for maximal permissiveness to the existing condition in [10], we show that OCC as described in Section III-E is more restrictive than LCC.

Lemma 4.4: Let H be an automaton over Σ, let $\Sigma_{\mathrm{uc}} \subseteq \Sigma$ be a set of uncontrollable events, and let $\Sigma_0 \subseteq \Sigma$. Then, it holds for $p_0 : \Sigma^* \to \Sigma_0^*$ that

$$p_0 \text{ is occ w.r.t. } L(H) \text{ and } \Sigma_{\mathrm{uc}} \Rightarrow p_0 \text{ is lcc w.r.t.} L(H) \text{ and } \Sigma_{\mathrm{uc}}.$$

Proof: Let p_0 be occ w.r.t. $L(H)$ and Σ_{uc}, and assume that $s \in L(H)$ for some $t \in p_0(L(H))$ with $p_0(s) = t$ and s.t. $t\sigma_{\mathrm{uc}} \in p_0(L(H))$ for some $\sigma_{\mathrm{uc}} \in \Sigma_{\mathrm{uc}}$. Then, either $\nexists u \in (\Sigma - \Sigma_0)^*$ s.t. $su\sigma_{\mathrm{uc}} \in L(H)$ or there is a $u \in (\Sigma - \Sigma_0)^*$ s.t. $su\sigma_{\mathrm{uc}} \in L(H)$. In the first case, LCC holds by definition. Otherwise, since p_0 is occ, it must hold that $u \in (\Sigma_{\mathrm{uc}} - \Sigma_0)^*$. Noting that t, σ_{uc}, and $s \in L$ were chosen arbitrarily, p_0 is lcc w.r.t. $L(H)$ and Σ_{uc}. ∎

With this result, it is readily observed that the conditions in [10] are more restrictive than our comparable conditions (1., 2., 3.(a)) in Corollary 4.1. The work in [10] requires OCC instead of 1., and assumes that the plant components do not share any events (i.e., mutual controllability is trivially fulfilled) instead of 2. Furthermore, the conditions 1., 2. and 3.(b) in Corollary 4.1 constitute a novel result for the nonblocking and maximally permissive hierarchical control.

V. Unified Approach for the Computation of Natural Projections

In this section, we investigate the algorithmic computation of projections that are suitable for the nonblocking and maximally permissive hierarchical and decentralized control. To this end, we present a generalized extension algorithm that allows the unified computation of projections that are natural observers and lcc, and msa-observers and lcc.

A. Generalized Extension Algorithm

The application of Algorithm 1 does not depend on the $L_m(H)$-observer property to be achieved: both the verification of the $L_m(H)$-observer condition (step 1. and the **if** clause in 2.) and the event set extension in step 2. only use the fact that the $L_m(H)$-observer property can be formulated as a quasi-congruence μ^*_{obs} on the state set of H with the corresponding QA $H_{\mu^*_{\mathrm{obs}},\Sigma_0}$ as stated in Theorem 3.1. Hence, Algorithm 1 can be employed for imposing any condition on the natural projection $p_0(L')$ of a language $L' \subseteq L(H)$ that can be formulated based on a quasi-congruence μ^* and such that the QA H_{μ^*,Σ_0} is deterministic and does not have σ_0-transitions. We denote this modification of Algorithm 1 with a general quasi-congruence μ^* instead of μ^*_{obs} and the corresponding QA H_{μ^*,Σ_0} instead of $H_{\mu^*_{\mathrm{obs}},\Sigma_0}$ as the *generalized extension algorithm.*

B. Computation of MSA-Observers

As a first application of the generalized extension algorithm, we observe that the msa-observer condition in Definition 3.2 is a condition for the projection $p_0(L_m(H))$ of the language $L_m(H) \subseteq L(H)$ that can be formulated as a quasi-congruence μ^*_{msa} with the corresponding QA $H_{\mu^*_{\mathrm{msa}},\Sigma_0}$ as stated in Theorem 3.4. Hence, the generalized extension algorithm enables the algorithmic computation of msa-observers, where $\mu^* = \mu^*_{\mathrm{msa}}$ and $H_{\mu^*,\Sigma_0} = H_{\mu^*_{\mathrm{msa}},\Sigma_0}$. Taking into account that μ^*_{msa} can be computed based on the dynamic system $\tilde{H}_{\mathrm{msa}}$ in Section III-C, the generalized extension algorithm is performed with a complexity of $\mathcal{O}\left(N_X^4 N_T^3\right)$.

C. Maximally Permissive Control

The goal of this section is the computation of projections $p_0 : \Sigma^* \to \Sigma_0^*$ that are lcc w.r.t the closed language $L(H)$ of an automaton H and a set of uncontrollable events Σ_{uc}. Referring to the considerations in Section V-A, we formulate LCC in terms of a quasi-congruence. We define the dynamic system $\tilde{H}_{\mathrm{lcc}} = (X, \{\Delta_\sigma | \sigma \in \Sigma_0\} \cup \{\Delta_{\sigma_{\mathrm{uc}},\mathrm{lcc}} | \sigma_{\mathrm{uc}} \in \Sigma_{0,\mathrm{uc}}\})$, with $\Delta_{\sigma_{\mathrm{uc}},\mathrm{lcc}} : X \to 2^X$ s.t.

$$\Delta_{\sigma_{\mathrm{uc}},\mathrm{lcc}}(x) := \begin{cases} \bigcup\limits_{\sigma \in \Sigma_0} \Delta_\sigma(x) & \text{if } \nexists u \in (\Sigma - \Sigma_0)^* \\ & \text{s.t. } \delta(x, u\sigma_{\mathrm{uc}})! \\ & \text{or } \exists u \in (\Sigma_{\mathrm{uc}} - \Sigma_{0,\mathrm{uc}})^* \\ & \text{s.t. } \delta(x, u\sigma_{\mathrm{uc}})! \\ \emptyset & \text{otherwise.} \end{cases} \tag{12}$$

That is, $\Delta_{\sigma_{\mathrm{uc}},\mathrm{lcc}}$ maps a state $x \in X$ to all states in $\bigcup_{\sigma \in \Sigma_0} \Delta_\sigma(x)$ if and only if the strings leading to the state fulfill LCC. Computing the coarsest quasi-congruence μ^*_{lcc} on $\tilde{H}_{\mathrm{lcc}}$ and the QA $H_{\mu^*_{\mathrm{lcc}},\Sigma_0}$, lcc can be verified in conjunction with the $L(H)$-observer condition with the following theorem.

Theorem 5.1 (LCC Verification): The projection p_0 is an $L(H)$-observer and lcc w.r.t. $L(H)$ and Σ_{uc} iff $H_{\mu^*_{\mathrm{lcc}},\Sigma_0}$ is deterministic and contains no σ_0-transitions.

Example 5.1: Theorem 5.1 is illustrated in Fig. 8. With the automaton H and the alphabets $\Sigma_0 = \{\alpha\}$, $\Sigma_{\mathrm{uc}} = \{\alpha, \mathtt{b}, \mathtt{c}, \mathtt{e}\}$, the dynamic system is $\tilde{H}_{\mathrm{lcc}} = (X, \{\Delta_\alpha, \Delta_{\alpha,\mathrm{lcc}}\})$ with $\Delta_\alpha(x) = X$ for all $x \in X$ and $\Delta_{\alpha,\mathrm{lcc}}(x) = X$ for $x \in \{2,3,5\}$, while $\Delta_{\alpha,\mathrm{lcc}}(1) = \Delta_{\alpha,\mathrm{lcc}}(4) = \emptyset$. The coarsest quasi-congruence μ^*_{lcc} is depicted by the shaded areas in H. The computation of the QA $H_{\mu^*_{\mathrm{lcc}},\Sigma_0}$ shows that p_0 with $\Sigma_0 = \{\alpha\}$ is not lcc. A projection that fulfills lcc is achieved when adding $\mathtt{a}$ to the abstraction alphabet, i.e., $\Sigma_0' = \{\mathtt{a}, \alpha\}$. Then, the corresponding QA is $H_{\mu^*_{\mathrm{msa}},\Sigma_0'}$ in Fig. 8.

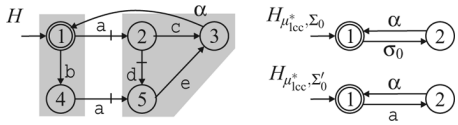

Fig. 8. Verification of local control consistency.

In order to prove Theorem 5.1, we introduce the *Nerode equivalence* $\equiv_L$ for a language $L \subseteq \Sigma^*$ [19]. Let $s, s' \in \overline{L}$. Then

$$s \equiv s' \mod \equiv_L \text{ iff } \forall\, u \in \Sigma^* : su \in L \Leftrightarrow s'u \in L. \quad (13)$$

Based on this definition, we now relate $\mu^*_{\rm lcc}$ and the Nerode equivalence $[\equiv_{p_0(L(H))} \circ p_0]$ for the projected language $p_0(L(H))$ in the following proposition. It shows that the projection of two strings in $L(H)$ leads to the same Nerode equivalence class in $p_0(L(H))$ if and only if the two strings lead to states in the same equivalence class of $\mu^*_{\rm lcc}$.

Proposition 5.1: Let $H = (X, \Sigma, \delta, x_0, X_m)$ be an arbitrary automaton and $[\equiv_{p_0(L(H))} \circ p_0], \mu^*_{\rm lcc}$ be defined as above, let p_0 be an $L(H)$-observer and lcc w.r.t. $L(H)$ and $\Sigma_{\rm uc}$. Also let x, $x' \in X$ and s, $s' \in L(H)$ s.t. $x = \delta(x_0, s)$ and $x' = \delta(x_0, s')$. Then

$$x \equiv x' \mod \mu^*_{\rm lcc} \Leftrightarrow s \equiv s' \mod [\equiv_{p_0(L(H))} \circ p_0]. \quad (14)$$

Proposition 5.1 is proved in Appendix B. Based on this result, Theorem 5.1 follows. To this end, we extend Δ_σ to strings: $\Delta_\varepsilon(x) := x$ and $\Delta_{t\sigma} := \bigcup\{\Delta_\sigma(x') | x' \in \Delta_t(x)\}$ for $x \in X$ and $t\sigma \in \Sigma_0^*$.

Proof: "$\Rightarrow$": It holds that p_0 is an $L(H)$-observer and lcc w.r.t. $L(H)$ and $\Sigma_{\rm uc}$. We show that $H_{\mu^*_{\rm lcc},\Sigma_0}$ is deterministic and has no σ_0-transitions.

We first prove that $H_{\mu^*_{\rm lcc},\Sigma_0}$ is deterministic. Assume the contrary and denote the state set and transition function of $H_{\mu^*_{\rm lcc},\Sigma_0}$ as Y and ν, respectively. Then, there are $y, \hat{y}, \hat{y}' \in Y$, $\sigma \in \Sigma_0$ s.t. $\hat{y} \neq \hat{y}'$ and $\{\hat{y}, \hat{y}'\} \subseteq \nu(y, \sigma)$. Then, there exist $x, x', \hat{x}, \hat{x}' \in X$ with $\{x, x'\} \subseteq cp^{-1}_{\mu^*_{\rm lcc}}(y)$, $\hat{x} \in cp^{-1}_{\mu^*_{\rm lcc}}(\hat{y})$, $\hat{x}' \in cp^{-1}_{\mu^*_{\rm lcc}}(\hat{y}')$ and $\delta(x, \sigma) = \hat{x}$, $\delta(x', \sigma) = \hat{x}'$ according to the QA construction. Let $s, s' \in \Sigma^*$ s.t. $\delta(x_0, s) = x$ and $\delta(x_0, s') = x'$, i.e., $s \equiv s' \mod [\equiv_{p_0(L(H))} \circ p_0]$ according to Proposition 5.1. Then, $\hat{x} \not\equiv \hat{x}' \mod \mu^*_{\rm lcc}$ (since $\hat{y} \neq \hat{y}'$) implies that $s\sigma \not\equiv s'\sigma \mod [\equiv_{p_0(L(H))} \circ p_0]$ (Proposition 5.1), i.e., w.l.o.g., there is $t \in \Sigma_0^*$ s.t. $p_0(s\sigma)t \in p_0(L(H))$ but $p_0(s'\sigma)t \notin p_0(L(H))$. But then, $p_0(s)\sigma t \in p_0(L(H))$, while $p_0(s')\sigma t \notin p_0(L(H))$ contradicts that $s \equiv s' \mod [\equiv_{p_0(L(H))} \circ p_0]$.

We now show that $H_{\mu^*_{\rm lcc},\Sigma_0}$ does not have σ_0-transitions. Assume the contrary. Then, there are $y, \hat{y} \in Y$ with $y \neq \hat{y}$ s.t. $\hat{y} \in \nu(y, \sigma_0)$ and $x, \hat{x} \in X$, $\sigma \in \Sigma - \Sigma_0$ s.t. $x \in cp^{-1}_{\mu^*_{\rm lcc}}(y)$, $\hat{x} \in cp^{-1}_{\mu^*_{\rm lcc}}(\hat{y})$, and $\hat{x} = \delta(x, \sigma)$ according to the QA construction. Let $x = \delta(x_0, s)$ for some $s \in L(H)$. Again, since $x \not\equiv \hat{x} \mod \mu^*_{\rm lcc}$, also $s \not\equiv s\sigma \mod [\equiv_{p_0(L(H))} \circ p_0]$. Then, w.l.o.g., there is $t \in \Sigma_0^*$ s.t. $p_0(s)t \in p_0(L(H))$ but $p_0(s\sigma)t \notin p_0(L(H))$. Since $p_0(s\sigma)t = p_0(s)t$, this leads to contradiction.

"$\Leftarrow$": It holds that $H_{\mu^*_{\rm lcc},\Sigma_0}$ is deterministic and has no σ_0-transitions. We show that p_0 is an $L(H)$-observer and lcc w.r.t. $L(H)$ and $\Sigma_{\rm uc}$.

We first prove that p_0 is an $L(H)$-observer. Let $s \in L(H)$ and $p_0(s)t \in p_0(L(H))$ for $t \in \Sigma_0^*$. Let $x := \delta(x_0, s)$ and $y := cp_{\mu^*_{\rm lcc}}(x)$. Then, there are two cases. If $\nu(y, t)!$, $\exists x' \in cp^{-1}_{\mu^*_{\rm lcc}}(y)$ s.t. $\Delta_t(x') \neq \emptyset$. Since $\mu^*_{\rm lcc}$ is a quasi-congruence for $\tilde{H}_{\rm lcc}$, also $\Delta_t(x) \neq \emptyset$. Hence, there is $u \in \Sigma^*$ s.t. $su \in L(H)$ and $p_0(su) = p_0(s)t$. If $\nu(y, t)$ does not exist, there must be $y' \neq y$ s.t. $\nu(y', t)!$ and $x' \in cp^{-1}_{\mu^*_{\rm lcc}}(y')$ with $\delta(x_0, s') = x'$ and $p_0(s') = p_0(s)$. But this is only possible if $H_{\mu^*_{\rm lcc},\Sigma_0}$ is nondeterministic or contains Σ_0-transitions, which leads to contradiction.

We finally show that p_0 is lcc w.r.t. $L(H)$ and $\Sigma_{\rm uc}$. Let $s \in L(H)$ and $p_0(s)\sigma \in p_0(L(H))$ for $\sigma \in \Sigma_{\rm uc} \cap \Sigma_0$. We write $x := \delta(x_0, s)$ and $y := cp_{\mu^*_{\rm lcc}}(x)$. Again, there are two cases. If $\nu(y, \sigma)!$, there is $x' \in cp^{-1}_{\mu^*_{\rm lcc}}(y)$ s.t. $\hat{x}' := \delta(x', \sigma)$ exists. Then, $\hat{x}' \in \Delta_\sigma(x')$ and $\hat{x}' \in \Delta_{\sigma,{\rm lcc}}(x')$ imply that $\Delta_\sigma(x') \cap \Delta_{\sigma,{\rm lcc}}(x') \neq \emptyset$. Since $\mu^*_{\rm lcc}$ is a quasi-congruence for $\tilde{H}_{\rm lcc}$, also $\Delta_\sigma(x) \cap \Delta_{\sigma,{\rm lcc}}(x) \neq \emptyset$. But this implies that $\exists u \in (\Sigma_{\rm uc} - \Sigma_0)^*$ s.t. $su\sigma \in L(H)$, i.e., p_0 is lcc for $\Sigma_{\rm uc}$. If $\nu(y, \sigma)$ does not exist, there must be $y' \neq y$ s.t. $\nu(y', \sigma)!$ and $s' \in L(H)$, $x' := \delta(x_0, s')$ with $cp_{\mu^*_{\rm lcc}}(x') = y'$ s.t. $p_0(s') = p_0(s)$. This is only possible if $H_{\mu^*_{\rm lcc},\Sigma_0}$ is nondeterministic or contains σ_0-transitions. ■

Hence, the generalized extension algorithm can be used for the computation of projections that are lcc if $\mu^* = \mu^*_{\rm lcc}$ and $H_{\mu^*,\Sigma_0} = H_{\mu^*_{\rm lcc},\Sigma_0}$ are chosen in Algorithm 2. Again, the computation of $\mu^*_{\rm lcc}$ based on the dynamic system $\tilde{H}_{\rm lcc}$ allows for a computational complexity of $\mathcal{O}\left(N_X^4 \cdot N_T^3\right)$. Furthermore, the computation of nonblocking projections and maximally permissive projections can be performed together. To this end, we define the dynamic systems $\tilde{H}_{\rm obs,lcc} := (X, \{\Delta_\sigma | \sigma \in \Sigma_0\} \cup \Delta_{\rm obs} \cup \{\Delta_{\sigma,{\rm lcc}} | \sigma \in \Sigma_{0,{\rm uc}}\})$ and $\tilde{H}_{\rm msa,lcc} := (X, \{\Delta_\sigma | \sigma \in \Sigma_0\} \cup \Delta_{\rm msa} \cup \{\Delta_{\sigma,{\rm lcc}} | \sigma \in \Sigma_{0,{\rm uc}}\})$. Then, Algorithm 2 with $\mu^* = \mu^*_{\rm obs,lcc}$ and the QA $H_{\mu^*,\Sigma_0} = H_{\mu^*_{\rm obs,lcc},\Sigma_0}$ is suitable for the unified computation of projections that are natural observers and lcc, while Algorithm 2 with $\mu^* = \mu^*_{\rm msa,lcc}$ and the QA $H_{\mu^*,\Sigma_0} = H_{\mu^*_{\rm msa,lcc},\Sigma_0}$ enables the unified computation of projections that are msa-observers and lcc with complexity $\mathcal{O}\left(N_X^4 \cdot N_T^3\right)$.

Algorithm 2 (Generalized Extension): **Input:** H, Σ_0

1. Compute the quasi-congruence μ^* and the QA H_{μ^*,Σ_0}.
2. **if** H_{μ^*,Σ_0} is deterministic and contains no σ_0-transitions

   return Σ_0

   else

   event set extension of Σ_0 as in **[15]** and **go to** 1.

VI. Application Example

In this section, we apply the concepts presented in the previous sections to several components of the Fischertechnik simulation model of the Chair of Automatic Control, University of Erlangen-Nuremberg in Fig. 9 (see also [8], [23]). We first illustrate the computation of different abstractions for a *stack feeder* in Section VI-A. Then, we employ the approach in Section V-C to synthesize maximally permissive hierarchical control for a

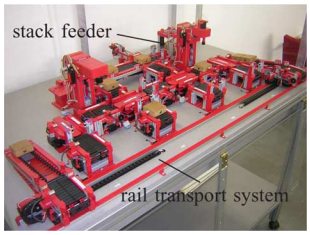

Fig. 9. Fischertechnik manufacturing system model.

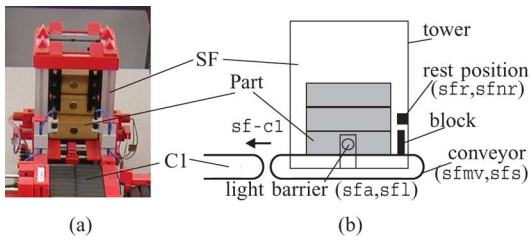

Fig. 10. Stack Feeder (SF): (a) front view; (b) side view.

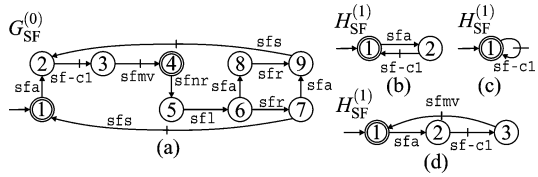

Fig. 11. SF level 0: (a) supervisor; SF abstraction level 1 (b) natural observer; (c) msa-observer (with and without LCC); (d) natural observer with LCC.

rail transport system in Section VI-B, and compare the different abstractions by means of a supervisor synthesis for the overall manufacturing system in Section VI-C. All computations are carried out with the *observer-plugin* of the `libFAUDES` software library for DES [26].

A. Stack Feeder

As is shown in Fig. 10, the stack feeder (SF) comprises a tower that can hold wooden parts and a conveyor belt with a small block attached to it that can push parts towards the neighboring conveyor belt C1. A *light barrier* detects if parts arrive or leave the SF which is modeled by the events `sfa` and `sfl`, respectively. In addition, the conveyor belt of the SF can start and stop moving (events `sfmv` and `sfs`), and a magnetic sensor detects if the small block attached to the belt reaches (`sfr`) or leaves (`sfnr`) its rest position. The motion of the belt is initiated by the event `sf-c1` that is shared with C1. The closed-loop behavior of the SF according to a supervisor design performed in [8] is given by the automaton $G^{(0)}_{\mathrm{SF}}$ in Fig. 11(a). It is desired that a present part is transported to C1 before the belt stops at its rest position.

We now investigate possible abstractions for a hierarchical supervisor design. The initial abstraction alphabet is $\Sigma^{(1)}_{\mathrm{SF}} = \{\texttt{sf-c1}\}$ with the only shared event `sf-c1`. The projection alphabets for the computation of $L_m\left(G^{(0)}_{\mathrm{SF}}\right)$-observers (Section III-D) and msa-observers (Section V-B) are $\Sigma^{(1)\prime}_{\mathrm{SF}} = \{\texttt{sf-c1}, \texttt{sfa}\}$ and $\Sigma^{(1)\prime}_{\mathrm{SF}} = \{\texttt{sf-c1}\}$, respectively. The corresponding abstracted plant models $H^{(1)}_{\mathrm{SF}}$ are shown

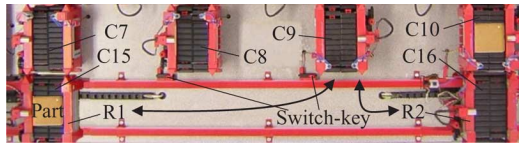

Fig. 12. Rail transport system (R) with connected conveyor belts (C).

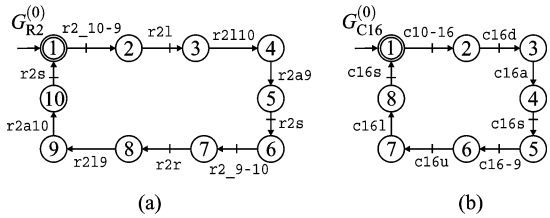

Fig. 13. Level 0 supervisors: (a) R2; (b) C16.

in Fig. 11(b) and (c), which suggests that the msa-observer condition is beneficial for this example. In addition, we compute locally control consistent projections that are suitable for maximally permissive control as proposed in Section V-C. Here, the $L_m(G^{(0)}_{\mathrm{SF}})$-observer computation with LCC yields $\Sigma^{(1)\prime}_{\mathrm{SF}} = \{\texttt{sf-c1}, \texttt{sfa}, \texttt{sfmv}\}$ (Fig. 11(d)) and the msa-observer computation with LCC leads to $\Sigma^{(1)\prime}_{\mathrm{SF}} = \{\texttt{sf-c1}\}$ (Fig. 11(c)) since there is no uncontrollable high-level event. It is interesting to note that the projection to $\{\texttt{sf-c1}, \texttt{sfa}, \texttt{sfmv}\}$ is suitable for maximally permissive control but does not fulfill OCC.

B. Rail Transport System

The rail transport system (R) is depicted in Fig. 12. It consists of two carts R1 and R2 that travel between different positions on a shared rail. A conveyor belt is mounted on each cart (C15 on R1 and C16 on R2) such that parts can be loaded. In our study, we assume that R1 can serve the conveyor belts C7 and C9, while R2 moves between C10 and C9. Switch-keys in front of each conveyor belt indicate the presence of R1 or R2.

1) R2: The closed-loop of R2 after a local supervisor synthesis is depicted in Fig. 13(a). R2 initially waits in front of C10. If a trip to C9 is requested (`r2_10 – 9`), R2 moves to the left (`r2l`), leaves the switch-key at C10 (`r2l10`), arrives (`r2a9`) and stops (`r2s`) at C9. The travel back to C10 is initiated by `r2_9-10`. Then, R2 moves to the right (`r2r`), leaves C9 (`r2l9`), arrives (`r2a10`) and stops at C10.

2) C16: $G^{(0)}_{\mathrm{C16}}$ in Fig. 13(b) represents the closed-loop of C16. The shared events `c10-16` and `c16-9` model the transport of a part from C10 to C16 and from C16 to C9, respectively. When a part arrives/leaves, `c16a`/`c16l` occurs. Furthermore, the motion of C16 is described by `c16d` (down), `c16u` (up) and `c16s` (stop).

3) R2 and C16: We now consider the joint action of R2 and C16, where it is desired that parts are transported from C10 to C9, while R2 and C16 are not allowed to move at the same time. This behavior is specified by $L_m\left(D^{(1)}_{\mathrm{R2C16}}\right)$ in Fig. 14(a). Following the hierarchical approach in Section III-A, we compute abstractions with the initial alphabet $\Sigma^{(1)}_{\mathrm{R2}} = \{\texttt{r2\_10-9}, \texttt{r2\_9-10}, \texttt{r2s}, \texttt{r2l9}, \texttt{r2l10}\}$ for $G^{(0)}_{\mathrm{R2}}$ and with $\Sigma^{(1)}_{\mathrm{C16}} = \{\texttt{c10-16}, \texttt{c16-9}, \texttt{c16s}\}$ for $G^{(0)}_{\mathrm{C16}}$, where `r2l9`

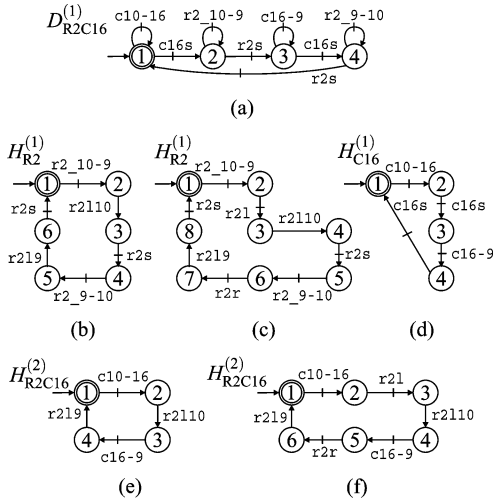

Fig. 14. Level 1 specification: (a) R2 and C16; Level 1 abstractions: (b) R2 (natural and msa-observer); (c) R2 (natural and msa-observer with LCC); (d) C16 (all abstractions); Level 2 abstractions: (e) R2 and C16 (natural and msa-observer); (f) R2 and C16 (natural and msa-observer with LCC).

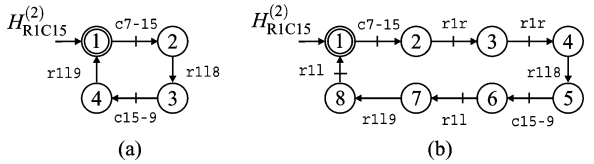

Fig. 15. Level 2 abstractions: (a) R1 and C15 (natural and msa-observer); (b) R1 and C15 (natural and msa-observer with LCC).

and r2l10 are included in $\Sigma^{(1)}_{\mathrm{R2}}$ for a later supervisor synthesis. The resulting abstracted plant model of R2 for the case of nonblocking control (both natural and msa-observer) and maximally permissive control (additional fulfillment of LCC) are depicted in Fig. 14(b) and (c), respectively. The computation of abstractions for C16 always yields the abstracted plant model $G^{(1)}_{\mathrm{C16}}$ in Fig. 14(d).

After the synthesis of a nonblocking supervisor for $L\left(G^{(1)}_{\mathrm{R2}} \| G^{(1)}_{\mathrm{C16}}\right)$ and $L_m\left(D^{(1)}_{\mathrm{R2C16}}\right)$, we obtain a further abstraction with the initial alphabet $\Sigma^{(2)}_{\mathrm{R2C16}} = \{\mathtt{c10\text{-}16}, \mathtt{c16\text{-}9}, \mathtt{r2l9}, \mathtt{r2l10}\}$ that contains all events that are shared with the connected components and further specifications used in the subsequent sections. The resulting abstractions for nonblocking control and maximally permissive control are shown in Fig. 15(e) and (f), respectively. In addition, it can be verified that all high-level closed-loops are live w.r.t. to their component alphabets, i.e., in each of the states of $H^{(2)}_{\mathrm{R2C16}}$, there exists a string that ends with an event in the alphabet of $H^{(1)}_{\mathrm{C16}}$ and $H^{(1)}_{\mathrm{R2}}$, respectively.

4) R1 and C15: An analogous synthesis as for R2 and C16 is performed for R1 and C15. Initially, R1 waits in front of C7. The main difference is that R1 has to pass C8 before C9 can be reached. The abstracted plant model $H^{(2)}_{\mathrm{R1C15}}$ is shown in Fig. 15 (a) (nonblocking control) and (b) (maximally permissive control).

5) Overall Rail Transport System: The overall rail transport system is composed of the abstracted plant models $H^{(2)}_{\mathrm{R1C15}}$ and $H^{(2)}_{\mathrm{R2C16}}$, and it is desired that R1 and R2 do not travel towards C9 simultaneously as specified by $D^{(3)}_{\mathrm{R1R2}}$ in Fig. 16(a). We design a nonblocking supervisor for $L\left(H^{(2)}_{\mathrm{R1C15}} \| H^{(2)}_{\mathrm{R2C16}}\right)$ and $L_m(D^{(3)}_{\mathrm{R1R2}})$ for the case of purely nonblocking control with 5 states and additional maximal permissive control with 26 states (it can be verified that mutual controllability is fulfilled for all plant components). Furthermore, we abstract the computed closed-loop behaviors in order to obtain a model of R that can be used in a supervisor synthesis for the overall manufacturing system. In the first case, we compute a nonblocking abstraction (natural observer and msa-observer) with the initial alphabet $\Sigma^{(3)}_{\mathrm{R1R2}} = \{\mathtt{c10\text{-}16}, \mathtt{c16\text{-}9}, \mathtt{c7\text{-}15}, \mathtt{c15\text{-}9}\}$ and the abstracted plant model with 3 states in Fig. 16(b). In the second case, the abstracted plant model with the projection alphabet $\Sigma^{(3)}_{\mathrm{R1R2}} \cup \{\mathtt{r1l}, \mathtt{r1r}, \mathtt{r2l}, \mathtt{r2r}\}$ has 17 states. The projection of this model to the alphabet $\Sigma^{(3)}_{\mathrm{R1R2}}$ is shown in Fig. 16(c). Comparing to Fig. 16(b), it is readily verified that a larger closed-loop behavior is achieved by enforcing maximally permissive control by means of projections that fulfill LCC. However, it has to be taken into account that the abstracted plant models for maximally permissive control are potentially larger than the models that are obtained if maximally permissive control is not enforced.

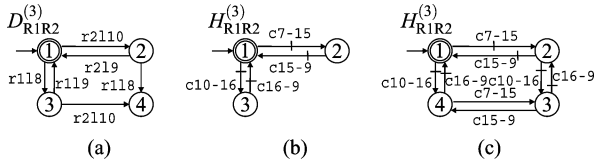

Fig. 16. Level 3 specification: (a) $D^{(3)}_{\mathrm{R1R2}}$; Level 3 abstraction: (b) R1 and R2 (natural and msa-observer); (b) R1 and R2 (natural and msa-observer with LCC).

This effect can also be seen in Fig. 17 that illustrates the hierarchical architecture for the rail transport system. Here, the shaded boxes indicate the automata that have to be used to implement the supervisor according to (7), and the numbers next to the automata names state the respective number of states for the nonblocking (left) and the maximally permissive (right) case. Together, the supervisors needed to achieve nonblocking control have a sum of 75 states, while the supervisors required for maximally permissive control have a state count of 106. In comparison, a monolithic synthesis evaluates a plant with 165 620 states and a specification with 168 states to compute a supervisor with 404 states.

C. Comparison of Different Abstractions

We follow the hierarchical design in [8], [23] to synthesize supervisors for the manufacturing system in Fig. 9 with its 25 subsystems. For abstraction, we compute natural observers and msa-observers and their maximally permissive counterparts. Table I shows the accumulated state counts of the resulting 39 supervisors.[3] It can be observed that the use of msa-observers is beneficial for this application. In addition, the results in Table I

[3]A monolithic supervisor would be in the order of 10^{30} states [8].

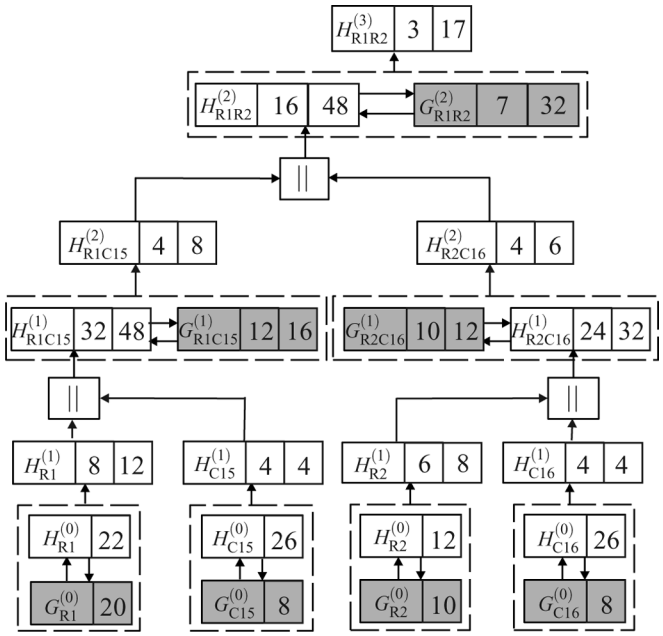

Fig. 17. Hierarchical architecture of the rail transport system.

TABLE I
SUPERVISOR STATE COUNTS FOR DIFFERENT ABSTRACTIONS

abstraction	natural observer	msa-observer	natural observer with LCC	msa-observer with LCC
state count	8138	3946	13729	4050

confirm that maximally permissive abstractions potentially lead to larger supervisors.

Note that although natural observers and msa-observers are not sufficient for maximally permissive control, supervisors obtained by these abstractions can still be maximally permissive. Considering large-scale systems, this property can be verified by dividing the supervisor product in (7) into smaller groups of supervisor products of manageable state sizes. If all such groups prove to be equal for both the purely nonblocking and the nonblocking and maximally permissive control, then the purely nonblocking supervisor synthesis also results in maximally permissive control. Hence, in case maximal permissiveness can be verified by such comparison, the smaller supervisors can be implemented while still achieving maximally permissive control. Since our example exhibits this property, it is possible to choose the smallest supervisor with an accumulated state count of 3946 for maximally permissiveness.[4]

VII. CONCLUSION

The abstraction-based synthesis of supervisors for large-scale discrete event systems (DES) is a powerful tool to cope with the potential state space explosion. In this respect, appropriate abstractions have to be chosen so as to guarantee desirable properties of the closed-loop system such as *nonblocking* behavior or *maximal permissiveness*.

[4]Note that the specification for the rail transport system in this application is different from the specification in Section VI-B.

In this paper, *natural projections* with certain properties serve as abstractions in a hierarchical and decentralized control architecture. As an extension to earlier work that introduces *natural observers* and *msa-observers* as sufficient conditions for nonblocking control, our work proposes *local control consistency* as a novel sufficient condition for maximally permissive control. In particular, it is shown that local control consistency is less conservative than *output control consistency* which was previously used to achieve maximal permissiveness.

Furthermore, we develop a concise method for the computation of natural projections that fulfill the derived sufficient conditions for nonblocking and maximally permissive hierarchical and decentralized control. To this end, we first point out that the *natural observer extension algorithm* by Feng and Wonham can be generalized to determine natural projections with properties that can be formulated as *quasi-congruences*. Then, we show that not only the natural observer condition and the msa-observer condition but also local control consistency can be stated in terms of a quasi-congruence. Hence, it is possible to employ this *generalized extension algorithm* for the computation of natural projections that are nonblocking (natural observers or msa-observers) and maximally permissive (locally control consistent).

The effectiveness of our approach is illustrated by the supervisor synthesis for several subsystems of a manufacturing system. It is observed that the condition on the natural projections that should preferably be employed depends on each specific application. In our manufacturing system, the msa-observer condition yields abstractions on smaller state spaces. Moreover, it is verified that adding the requirement of locally control consistent natural projections for maximally permissive control potentially leads to larger state spaces of the abstracted plant models. This result illustrates the trade-off between the size of the resulting supervisor and the possible conservativeness of the control.

APPENDIX A
PROOF OF LEMMA 4.2

We define $\theta_i : \Sigma^* \to \Sigma_i^*$ and state the following Lemma.

Lemma A.1([23], [Lemma A.8)]: Let $H := \|_{j=1}^{n} H_j$, let H_i, H_k, $i, k = 1, \ldots, n$, $i \neq k$, be mutually controllable, and let $s_i \in L(H_i)$ and $\sigma \in \Sigma_{i,\mathrm{uc}}^{\mathrm{hi}}$ s.t. $s_i\sigma \in L(H_i)$. Then, for all $s \in L(H)$ s.t. $\theta_i(s) = s_i$, it holds that $s\sigma \in L(H)$.

Using Lemma A.1, Lemma 4.2 can be proved.

Proof: Let $s \in \kappa_{L(H)}(L_m(H)\|K_1\| \cdots \|K_n)$. Then, $s \in L_m(H)$, $s_i := \theta_i(s) \in K_i$ for $i = 1, \ldots, n$, and $\nexists u \in \Sigma_{\mathrm{uc}}^*$ s.t. $su \in L(H)$ and $su \notin \overline{L_m(H)\|K_1\| \cdots \|K_n}$. Now assume that $s \notin \|_{i=1}^{n} \kappa_{L(H_i)}(L_m(H_i)\|K_i)$. Then, for some k, $s_k \notin \kappa_{L(H_k)}(L_m(H_k)\|K_k)$, i.e., there is a $u_k \in \Sigma_{k,\mathrm{uc}}^*$ s.t. $s_k u_k \notin \overline{K_k}$ but $s_k u_k \in L(H_k)$. Let $u_k = v_1\sigma_1 \cdots v_m\sigma_m v_{m+1}$, where $v_j \in \left(\Sigma_{k,\mathrm{uc}} - \Sigma_{k,\mathrm{uc}}^{\mathrm{hi}}\right)^*$, $j = 1, \ldots, m+1$ and $\sigma_j \in \Sigma_{k,\mathrm{uc}}^{\mathrm{hi}}$, $j = 1, \ldots, m$. Because of mutual controllability, repeated application of Lemma A.1 yields $su_k \in L(H)$ but $su_k \notin \overline{\|_{i=1}^{n} K_i}$ since $s_k u_k \notin \overline{K_k}$. This violates the assumption $s \in \kappa_{L(H)}(L_m(H)\|K_1\| \cdots \|K_n)$, and hence, $s \in \|_{i=1}^{n} \kappa_{L(H_i)}(L_m(H_i)\|K_i)$. ■

APPENDIX B
PROOF OF PROPOSITION 5.1

We first establish three lemmas.

Lemma B.1: Let $\mu \in \mathcal{E}(X)$ be a quasi-congruence on $\tilde{H}_{\rm lcc}$, and $x, x' \in X$. Then $x \equiv x' \mathrm{mod}\mu$ implies

1) $\forall\, t \in \Sigma_0^*: \Delta_t(x) \neq \emptyset \Leftrightarrow \Delta_t(x') \neq \emptyset$;
2) $\forall\, \sigma \in \Sigma_{0,\rm uc}: \Delta_{\sigma,\rm lcc}(x) \neq \emptyset \Leftrightarrow \Delta_{\sigma,\rm lcc}(x') \neq \emptyset$.

Proof: To show 1), assume that $t \in \Sigma_0^*$ s.t. $\Delta_t(x) \neq \emptyset$. We show that $\Delta_t(x') \neq \emptyset$ by induction. Let $t = \sigma_1 \cdots \sigma_m$, where $\sigma_1 = \varepsilon$ and $\sigma_i \in \Sigma_0$ for $i = 2, \ldots, m$. As the induction base, we observe that $x_1 := x \in \Delta_\varepsilon(x_1)$, $x_1' := x' \in \Delta_\varepsilon(x_1')$ and $x_1 \equiv x_1' \mathrm{mod}\mu$. Now, assume that for some $i \in \{1, \ldots, m-1\}$, $x_i \in \Delta_{\sigma_1 \cdots \sigma_i}(x_1)$, $x_1' \in \Delta_{\sigma_1 \cdots \sigma_i}(x_1')$ and $x_i \equiv x_i' \mathrm{mod}\mu$. Since $\Delta_t(x_1) \neq \emptyset$, there is $x_{i+1} \in \Delta_{\sigma_{i+1}}(x_i)$. As μ is a quasi-congruence for $\Delta_{\sigma_{i+1}}$, there must be $x_{i+1}' \in \Delta_{\sigma_{i+1}}(x_i')$ s.t. $x_{i+1} \equiv x_{i+1}' \mathrm{mod}\mu$. Hence, $x_{i+1} \in \Delta_{\sigma_1 \cdots \sigma_{i+1}}(x_1)$, $x_{i+1}' \in \Delta_{\sigma_1 \cdots \sigma_{i+1}}(x_1')$, and $x_{i+1} \equiv x_{i+1}' \mathrm{mod}\mu$. But then, induction on the length of t shows that $\Delta_t(x') = \Delta_{\sigma_1 \cdots \sigma_m}(x_1') \neq \emptyset$.

For 2), let $\Delta_{\sigma,\rm lcc}(x) \neq \emptyset$ for some $\sigma \in \Sigma_{0,\rm uc}$. Thus, there is $u \in (\Sigma_{\rm uc} - \Sigma_{0,\rm uc})^*$ s.t. $\hat{x} := \delta(x, u\sigma)$ exists and $\hat{x} \in \Delta_{\sigma,\rm lcc}(x)$. Since μ is a quasi-congruence for $\Delta_{\sigma,\rm lcc}$, there must be $\hat{x}' \in \Delta_{\sigma,\rm lcc}(x')$ s.t. $\hat{x} \equiv \hat{x}' \mathrm{mod}\mu$. Hence, $\Delta_{\sigma,\rm lcc}(x') \neq \emptyset$. ∎

Lemma B.2: Let p_0 be an $L(H)$-observer and lcc w.r.t. $L(H)$ and $\Sigma_{\rm uc} \subseteq \Sigma$. Define $\mu_{\rm lcc} \in \mathcal{E}(X)$ s.t. for any $x, x' \in X$ and $s, s' \in L(H)$ with $x = \delta(x_0, s)$ and $x' = \delta(x_0, s')$, it holds that $x \equiv x' \mathrm{mod}\mu_{\rm lcc} \Leftrightarrow s \equiv s' \,\mathrm{mod}[\equiv_{p_0(L(H))} \circ p_0]$. Then, $\mu_{\rm lcc}$ is a quasi-congruence for $\tilde{H}_{\rm lcc}$.

Proof: Let $x, x' \in X$ s.t. $x \equiv x' \mathrm{mod}\mu_{\rm lcc}$, and $s, s' \in L(H)$ s.t. $x = \delta(x_0, s)$ and $x' = \delta(x_0, s')$.

First assume that $\sigma \in \Sigma_0$ s.t. $\hat{x} \in \Delta_\sigma(x)$. Then, there are $u, \hat{u} \in (\Sigma - \Sigma_0)^*$ s.t. $\hat{x} = \delta(x, u\sigma\hat{u})$ and $p_0(su\sigma\hat{u}) = p_0(s)\sigma$. It has to be shown that $\exists \hat{x}' \in \Delta_\sigma(x')$ s.t. $\hat{x} \equiv \hat{x}' \mathrm{mod}\mu_{\rm lcc}$. Assume that $t \in \Sigma_0^*$ s.t. $p_0(s)\sigma t \in p_0(L(H))$. Since $s \equiv s' \mathrm{mod}[\equiv_{p_0(L(H))} \circ p_0]$, and $p_0(s)\sigma t \in p_0(L(H))$, also $p_0(s')\sigma t \in p_0(L(H))$. Since p_0 is an $L(H)$-observer, there are $u', \hat{u}' \in (\Sigma - \Sigma_0)^*$, $\tilde{u}' \in \Sigma^*$ s.t. $s'u'\sigma\hat{u}'\tilde{u}' \in L(H)$ and $p_0(s'u'\sigma\hat{u}'\tilde{u}') = p_0(s')\sigma t$. Then, $su\sigma\hat{u} \equiv su'\sigma\hat{u}' \mathrm{mod}[\equiv_{p_0(L(H))} \circ p_0]$. Hence, with $\hat{x}' := \delta(x', u'\sigma\hat{u}')$, we have that $\hat{x} \equiv \hat{x}' \mathrm{mod}\mu_{\rm lcc}$. As $\sigma \in \Sigma_0$ was arbitrary, this implies that $\mu_{\rm lcc}$ is a quasi-congruence for $(X, \{\Delta_\sigma | \sigma \in \Sigma_0\})$.

Now assume that $\hat{x} \in \Delta_{\sigma,\rm lcc}(x)$ for some $\sigma \in \Sigma_{0,\rm uc}$. Then, there are $u, \hat{u} \in (\Sigma - \Sigma_0)^*$, $\tilde{u} \in (\Sigma_{\rm uc} - \Sigma_{0,\rm uc})^*$ and $\hat{\sigma} \in \Sigma_0$ s.t. $\hat{x} = \delta(x, u\hat{\sigma}\hat{u})$ and $\delta(x, \tilde{u}\sigma)$ exists. It has been shown above that there is $\hat{x}' \in \Delta_{\hat{\sigma}}(x')$ s.t. $\hat{x} \equiv \hat{x}' \mathrm{mod}\mu_{\rm lcc}$. It remains to verify that $\hat{x}' \in \Delta_{\sigma,\rm lcc}(x')$. Since $s \equiv s' \mathrm{mod}[\equiv_{p_0(L(H))} \circ p_0]$ and $p_0(s\tilde{u}\sigma) = p_0(s)\sigma \in p_0(L(H))$, also $p_0(s')\sigma \in p_0(L(H))$. Since p_0 is lcc w.r.t. $L(H)$ and $\Sigma_{\rm uc}$, there must be $\tilde{u}' \in (\Sigma_{\rm uc} - \Sigma_{0,\rm uc})^*$ s.t. $\delta(x', \tilde{u}\sigma)$ exists. But then, $\hat{x}' \in \Delta_{\sigma,\rm lcc}(x')$, i.e., $\mu_{\rm lcc}$ is also a quasi-congruence for $(X, \Delta_{\sigma,\rm lcc})$. As $\sigma \in \Sigma_{0,\rm uc}$ was arbitrary, $\mu_{\rm lcc}$ is a quasi-congruence for $(X, \{\Delta_{\sigma,\rm lcc} | \sigma \in \Sigma_{0,\rm uc}\})$. ∎

Lemma B.3: Assume that 1) and 2) in Lemma B.1 are fulfilled for $x, x' \in X$, and p_0 is an $L(H)$-observer and lcc w.r.t. $L(H)$ and $\Sigma_{\rm uc}$. Then $x \equiv x' \mathrm{mod}\mu_{\rm lcc}$.

Proof: Let $s, s' \in L(H)$ s.t. $x = \delta(x_0, s)$, $x' = \delta(x_0, s')$, and assume that $p_0(s)t \in p_0(L(H))$. We have to show that $p_0(s')t \in p_0(L(H))$ s.t. $s \equiv s' \mathrm{mod}[\equiv_{p_0(L(H))} \circ p_0]$.

Since p_0 is an $L(H)$-observer, there is $u \in \Sigma^*$ s.t. $su \in L(H)$ and $p_0(su) = p_0(s)t$. Hence, $\Delta_t(x) \neq \emptyset$, and Lemma B.1 1) implies that $\Delta_t(x') \neq \emptyset$. Thus, there is $u' \in \Sigma^*$ s.t. $s'u' \in L(H)$ and $p_0(s'u') = p_0(s')t \in p_0(L(H))$. That is, $s \equiv s' \mathrm{mod}[\equiv_{p_0(L(H))} \circ p_0]$, and hence $x \equiv x' \mathrm{mod}\mu_{\rm lcc}$. ∎

We now prove Proposition 5.1.

Proof: Following Lemma B.2, we have to show that $\mu_{\rm lcc} = \mu_{\rm lcc}^*$. Let $\mu' \in \mathcal{E}(X)$ be a quasi-congruence for $\tilde{H}_{\rm lcc}$ and $x \equiv x' \mathrm{mod}\mu'$ for $x, x' \in X$. Lemma B.1 suggests that 1) and 2) hold and Lemma B.3 implies that $x \equiv x' \mathrm{mod}\mu_{\rm lcc}$, i.e., $\mu' \leq \mu_{\rm lcc}$. Hence, $\mu_{\rm lcc}$ is the coarsest quasi-congruence $\mu_{\rm lcc}^*$ for $\tilde{H}_{\rm lcc}$. ∎

REFERENCES

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

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

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

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

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

[6] A. de Cunha and J. Cury, "Hierarchical supervisory control based on discrete event systems with flexible marking," *IEEE Trans. Autom. Control*, vol. 52, no. 12, pp. 2242–2253, Dec. 2007.

[7] H. Flordal, R. Malik, M. Fabian, and K. Åkesson, "Compositional synthesis of maximally permissive supervisors using supervision equivalence," *Discrete Event Dyn. Syst.: Theor. Appl.*, vol. 17, no. 4, pp. 475–504, 2007.

[8] K. Schmidt, T. Moor, and S. Perk, "Nonblocking hierarchical control of decentralized discrete event systems," *IEEE Trans. Autom. Control*, vol. 53, no. 10, pp. 2252–2265, Oct. 2008.

[9] S. Perk, T. Moor, and K. Schmidt, "Controller synthesis for an I/O-based hierarchical system architecture," in *Workshop Discrete Event Syst.*, 2008, pp. 474–479.

[10] L. Feng and W. Wonham, "Supervisory control architecture for discrete-event systems," *IEEE Trans. Autom. Control*, vol. 53, no. 6, pp. 1449–1461, Jun. 2008.

[11] R. Hill, D. Tilbury, and S. Lafortune, "Modular supervisory control with equivalence-based conflict resolution," in *Proc. Amer. Control Conf.*, 2008, pp. 491–498.

[12] J. Komenda and J. van Schuppen, "Coordination control of discrete-event systems," in *Workshop Discrete Event Syst.*, 2008, pp. 9–15.

[13] R. Su, J. van Schuppen, and J. Rooda, "Supervisor synthesis based on abstractions of nondeterministic automata," in *Workshop Discrete Event Syst.*, 2008, pp. 412–418.

[14] K. Schmidt and T. Moor, "Marked-string accepting observers for the hierarchical and decentralized control of discrete event systems," in *Workshop Discrete Event Syst.*, 2006, pp. 413–418.

[15] L. Feng and W. Wonham, "On the computation of natural observers in discrete-event systems," *Discrete Event Dyn. Syst.: Theor. Appl.*, vol. 20, no. 1, pp. 63–102, Mar. 2010.

[16] K. C. Wong and W. M. Wonham, "On the computation of observers in discrete-event systems," *Discrete Event Dyn. Syst.: Theor. Appl.*, vol. 14, no. 1, pp. 55–107, 2004.

[17] K. Schmidt and C. Breindl, "On maximal permissiveness of hierarchical and modular supervisory control approaches for discrete event systems," in *Workshop Discrete Event Syst.*, 2008, pp. 462–467.

[18] P. J. Ramadge and W. M. Wonham, "The control of discrete event systems," in *Proc. IEEE, Special Issue Discrete Event Dyn. Syst.*, 1989, vol. 77, pp. 81–98.

[19] W. M. Wonham, "Supervisory Control of Discrete-Event Systems," Dept. Elect. Comp. Eng., Univ. Toronto, Toronto, ON, Canada, 2008 [Online]. Available: http://www.control.utoronto.ca/DES

[20] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Secaucus, NJ: Springer-Verlag, 2006.
[21] K. Schmidt, J. Reger, and T. Moor, "Hierarchical control of structural decentralized DES," in *Workshop Discrete Event Syst.*, 2004, pp. 289–294.
[22] R. Hill and D. Tilbury, "Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction," in *Workshop Discrete Event Syst.*, 2006, pp. 399–406.
[23] K. Schmidt, "Hierarchical Control of Decentralized Discrete Event Systems: Theory and Application," Ph.D. dissertation, Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg, Nürnberg, Germany, 2005.
[24] J.-C. Fernandez, "An implementation of an efficient algorithm for bisimulation equivalence," *Sci. Comp. Programming*, vol. 13, pp. 219–236, 1990.
[25] S.-H. Lee and K. C. Wong, "Structural decentralised control of concurrent DES," *Eur. J. Control*, vol. 35, pp. 1125–1134, 2002.
[26] "libFAUDES Software Library for Discrete Event Systems," libFAUDES., Tech. Rep., 2006–2010 [Online]. Available: www.rt.eei.uni-erlangen.de/FGdes/faudes

Klaus Schmidt (M'11) received the Diploma and Ph.D. degrees in electrical, electronic, and communication engineering from the University of Erlangen-Nürnberg, Germany, in 2002 and 2005, respectively.

He is currently an Assistant Professor with the Department of Electronic and Communication Engineering, Cankaya University, Ankara, Turkey. His research interests include controller synthesis and failure diagnosis for discrete event systems, industrial automation systems, vehicular communication networks, and industrial project control.

Christian Breindl received the Diploma degree in mechatronics from the University of Erlangen-Nürnberg, Germany, in 2007.

He is currently a Research Assistant at the Institute for Systems Theory and Automatic Control, University of Stuttgart, Germany. His current research interests are in the field of systems biology with focus on modeling and analysis of gene regulatory networks.