Marked-String Accepting Observers for the Hierarchical and Decentralized Control of Discrete Event Systems

Klaus Schmidt and Thomas Moor
Universität Erlangen-Nürnberg
Lehrstuhl für Regelungstechnik
Cauerstraße 7 - 91058 Erlangen, Germany
{klaus.schmidt,thomas.moor}@rt.eei.uni-erlangen.de

***Abstract*— The paper extends previous work, where we develop a control theory for nonblocking hierarchical control of decentralized discrete event systems (DES). The results are based on two technical conditions for the hierarchical abstraction: it has to be (i) *locally nonblocking* and (ii) *marked string accepting*.**

In this paper, we investigate the systematic construction of the hierarchical abstraction. Starting from an initial natural projection which need not fulfill (i) and (ii), we provide an algorithm to compute the hierarchical abstraction with the coarsest equivalence kernel finer than that of the initial natural projection, and such that (i) and (ii) hold. Our approach extends the work in [10], where the authors compute observers for the hierarchical control of DES.

I. Introduction

Recent approaches for the control of large-scale discrete event systems employ hierarchical control architectures for reducing the computational complexity of supervisor synthesis [1], [3], [4], [6], [7], [9], [11]. In hierarchical architectures, controller synthesis is based on a plant abstraction (high-level model), which is supposed to be less complex than the original plant model (low-level model). The main question is how to derive the plant abstraction and the low-level supervisor implementation of a high-level controller such that the low-level closed-loop system is nonblocking and satisfies the expected high-level behavior.

All of the above approaches assume that the high-level observation is given. The method in [1] employs a two-level control hierarchy such that hierarchically consistent and nonblocking control are guaranteed by construction. In [3], [4], [6], [9], certain conditions for nonblocking and hierarchically consistent control are required. However, little is known about the systematic choice of high-level observations such that these conditions are fulfilled.

A first result in this direction is elaborated in [10] based on the theory of observers in [11]. An observer with the coarsest possible equivalence kernel that is finer than that of an initial *causal reporter map* is computed. Nevertheless, the choice of the initial reporter map is not obvious.

In this paper, we consider the hierarchical and decentralized architecture presented in [7], where the overall system is modeled by the synchronous product of decentralized subsystems. A natural projection, where the *shared events* of the decentralized subsystems must be contained in the high-level alphabet, is used for hierarchical abstraction. For nonblocking and hierarchically consistent control, it is required that this natural projection is (i) *locally nonblocking* and (ii) *marked string accepting*. Similar to the observer algorithm in [10], we develop a procedure to modify an initial natural projection such that the resulting natural projection satisfies (i) and (ii), and has the coarsest equivalence kernel possible. However, in our case, the choice of the initial natural projection is straightforward; it is the projection on the shared events of the decentralized subsystems.

The outline of the paper is as follows. Basic definitions of supervisory control theory are recalled in Section II. Section III discusses the features of the hierarchical and decentralized approach in [7] and formalizes the problem statement. Our algorithm is developed and illustrated with an example in Section IV. Section V elaborates how the algorithm can be applied to build an architecture for nonblocking hierarchical and decentralized supervisory control.

II. Preliminaries

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

The *natural projection* $p_i : \Sigma^* \to \Sigma_i^*$, $i = 1,2$, for the 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 $\sigma \in \Sigma_i$, and $p_i(s\sigma) := p_i(s)$ otherwise. The set-valued inverse of p_i is denoted $p_i^{-1} : \Sigma_i^* \to 2^{\Sigma^*}$. The *synchronous product* $M_1 || M_2 \subseteq \Sigma^*$ of $M_i \subseteq \Sigma_i^*$, $i = 1,2$ is $M_1 || M_2 = p_1^{-1}(M_1) \cap p_2^{-1}(M_2) \subseteq \Sigma^*$.

A *finite automaton* is a tuple $G = (X, \Sigma, \delta, x_0, X_\mathrm{m})$, with the finite set of *states* X; the finite alphabet of *events* Σ; the partial *transition function* $\delta : X \times \Sigma \to X$; the *initial state* $x_0 \in X$; and the set of *marked states* $X_\mathrm{m} \subseteq X$. We write $\delta(x,\sigma)!$ if δ is defined at (x,σ). In order to extend δ to

a partial function on $X \times \Sigma^*$, recursively let $\delta(x,\varepsilon) := x$ and $\delta(x,s\sigma) := \delta(\delta(x,s),\sigma)$, whenever both $x' = \delta(x,s)$ and $\delta(x',\sigma)!$. $L(G) := \{s \in \Sigma^* : \delta(x_0,s)!\}$ and $L_m(G) := \{s \in L(G) : \delta(x_0,s) \in X_m\}$ are the *closed* and *marked language* generated by the finite automaton G, respectively. G is *nonblocking* if $\overline{L_m(G)} = L(G)$, i.e. if each string in $L(G)$ is the prefix of a marked string in $L_m(G)$. For any string $s \in L(G)$, $\Sigma(s) := \{\sigma | s\sigma \in L(G)\}$ is the set of eligible events after s. For a definition of the synchronous composition $G_1||G_2$, see e.g. [12]; in particular $L_m(G_1||G_2) = L_m(G_1)||L_m(G_2)$.

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

III. Hierarchical Control Approach

In [7], we develop a hierarchical approach to the control of decentralized DES, as illustrated in Figure 1.

A *decentralized DES* is a synchronous product system $G := ||_{i=1}^n G_i$ where each G_i, for $i = 1,\ldots,n$, is a finite automaton with event alphabet Σ_i, and the event alphabet of G is $\Sigma := \bigcup_{i=1}^n \Sigma_i$. High-level abstractions G_i^{hi} of the low-level subsystems G_i are computed by evaluating the natural projections $p_i^{dec} : \Sigma_i^* \to (\Sigma_i^{hi})^*$ of the low-level languages $L(G_i)$ and $L_m(G_i)$ such that $L(G_i^{hi}) = p_i^{dec}(L(G_i))$ and $L_m(G_i^{hi}) = p_i^{dec}(L_m(G_i))$. We require that

a. the high-level alphabets Σ_i^{hi} are chosen such that $\bigcup_{j\neq i}^n (\Sigma_i \cap \Sigma_j) \subseteq \Sigma_i^{hi} \subseteq \Sigma_i$, i.e. Σ_i^{hi} contains all events shared with other components.

The overall high-level model G^{hi} is defined such that $L(G^{hi}) := p^{hi}(L(G))$ and $L_m(G^{hi}) = p^{hi}(L_m(G))$ with the natural projection $p^{hi} : \Sigma^* \to (\bigcup_{i=1}^n \Sigma_i^{hi})$. Using assumption a., it can be shown [9] that $G^{hi} = ||_{i=1}^n G_i^{hi}$. This means that instead of deriving the high-level model G^{hi} from the overall low-level model G, a parallel composition of the decentralized high-level models G_i^{hi} can be evaluated. The tuple $(||_{i=1}^n G_i, ||_{i=1}^n G_i^{hi})$ is denoted a *decentralized projected DES*. A nonblocking high-level supervisor S^{hi} for G^{hi} and a high-level specification $E^{hi} \subseteq L_m(G^{hi})$ is implemented by decentralized low-level supervisors S_i^{lo} that exist if

b. the high-level languages $L(G_i^{hi})$ are mutually controllable (see [5]).

The hierarchical and decentralized control architecture guarantees nonblocking and hierarchically consistent control if

c. the decentralized low-level/high-level pairs (G_i, G_i^{hi}) are locally nonblocking and marked string accepting, as formulated in Definitions 3.1 and 3.2 below.

The approach is computationally efficient as the overall system need not be computed for both the abstraction and the supervisor implementation.

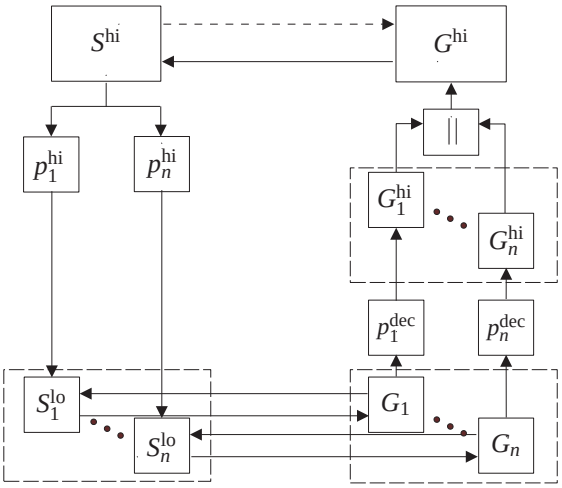

Fig. 1. Hierarchical architecture

From the perspective of each individual subsystem G_i, nonblocking control is based on two different types of conditions. Verifying mutual controllability of the high-level languages $L(G_i^{hi})$ (b.) involves the other subsystems. In contrast, the locally nonblocking and the marked string accepting condition (c.) exclusively depend on the behavior of each individual tuple (G_i, G_i^{hi}), denoted *projected system* (PS), and the choice of the high-level alphabet (a.).

The latter *structural* conditions, which only depend on the system structure of each PS, are the focus of this paper. Throughout Sections III and IV, we will make a notational simplification (an avoidance of subscripts) by replacing the pair (G_i, G_i^{hi}) with (H, H^{hi}), having event alphabets Σ and $\Sigma^{hi} \subseteq \Sigma$ and the natural projection $p^{hi} : \Sigma^* \to (\Sigma^{hi})^*$.

A PS (H, H^{hi}) is locally nonblocking if for all low-level strings $s \in L(H)$ and for all high-level events $\sigma \in \Sigma^{hi}$ which are feasible after the high-level string $p^{hi}(s)$, there exists a local path starting from s on which event σ can occur.

Definition 3.1 (Locally Nonblocking Condition): Let (H, H^{hi}) be a PS. The string $s^{hi} \in L(H^{hi})$ is locally nonblocking if for all $s \in L(H)$ with $p^{hi}(s) = s^{hi}$ and $\forall \sigma \in \Sigma^{hi}(s^{hi})$, $\exists u \in (\Sigma - \Sigma^{hi})^*$ s.t. $su\sigma \in L(H)$. (H, H^{hi}) is locally nonblocking if this is true for all $s^{hi} \in L(H^{hi})$.

For formulating the marked string accepting condition, the set of *exit strings* is needed. For a given PS (H, H^{hi}) and a high-level string $s^{hi} \in L(H^{hi})$, the set of exit strings $L_{ex}(s^{hi})$ is the set of corresponding low-level strings which have a

high-level successor event, i.e. $L_{\mathrm{ex}}(s^{\mathrm{hi}}) := \{s \in L(H) | p^{\mathrm{hi}}(s) = s^{\mathrm{hi}} \wedge (\exists \sigma \in \Sigma^{\mathrm{hi}} \text{ s.t. } s\sigma \in L(H))\} \subseteq \Sigma^*$.

Marked string acceptance guarantees that if the high-level system passes a marked string, the low-level system also has to pass a marked string.

Definition 3.2 (Marked String Acceptance): Let (H, H^{hi}) be a PS. The string $s^{\mathrm{hi}} \in L_{\mathrm{m}}(H^{\mathrm{hi}})$ is marked string accepting[1] if for all $s \in L_{\mathrm{ex}}(s^{\mathrm{hi}})$

$$\exists s' \leq s \text{ with } p^{\mathrm{hi}}(s') = s^{\mathrm{hi}} \text{ and } s' \in L_{\mathrm{m}}(H). \qquad (1)$$

(H, H^{hi}) is marked string accepting if s^{hi} is marked string accepting for all $s^{\mathrm{hi}} \in L_{\mathrm{m}}(H^{\mathrm{hi}})$.

According to condition a., the choice of the high-level alphabets Σ_i^{hi} is restricted by $\bigcup_{j \neq i}(\Sigma_i \cap \Sigma_j) \subseteq \Sigma_i^{hi} \subseteq \Sigma_i$. To keep the high-level model H_i^{hi} small, a natural candidate is $\Sigma_i^{\mathrm{hi}} = \bigcup_{j \neq i}(\Sigma_i \cap \Sigma_j)$. However, choosing this Σ_i^{hi}, the locally nonblocking and the marked string accepting condition need not be fulfilled. An intuitive solution to this problem is presented in the following example.

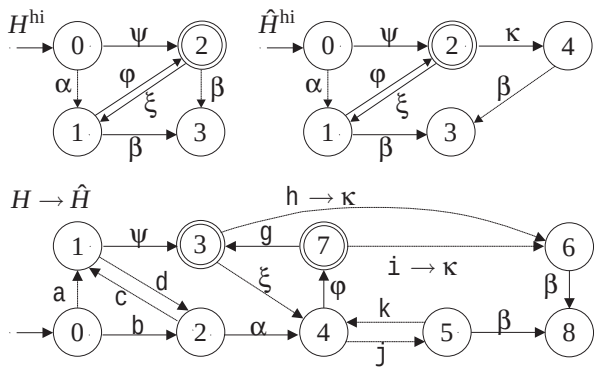

Fig. 2. Automaton with relabeling

Example 3.1: Consider the PS (H, H^{hi}) for the automata H and H^{hi} in Figure 2, where $\Sigma^{\mathrm{hi}} := \{\alpha, \beta, \gamma, \delta, \varphi, \psi\}$. (H, H^{hi}) is marked string accepting but not locally nonblocking.

An ad hoc solution to the problem is obtained if the low-level transitions from state 3 and 7 to state 6 are relabeled κ (as indicated in Figure 2) and $\hat{\Sigma}^{\mathrm{hi}} = \Sigma^{\mathrm{hi}} \cup \{\kappa\}$ is used as the high-level alphabet.[2]

Thus, the question arises if there is a systematic way to determine $\hat{\Sigma}^{\mathrm{hi}}$ such that condition c. holds by adding high-level observations. The next section provides an algorithm for computing the minimal $\hat{\Sigma}^{\mathrm{hi}}$ meeting condition c. The corresponding natural projection is called an *msa-observer*.

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

[2]relabeling in H just changes the observation sent to the high level.

IV. Computation of MSA-Observers

A. Basic Notation

We first present basic results from set theory. 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 $m, m' \in M$, $m \equiv m' \mod \ker f$ iff $f(m) = f(m')$.

Given two equivalence relations η and μ on M, $\eta \leq \mu$, i.e. η refines μ, if $m \equiv m' \mod \eta \Rightarrow m \equiv m' \mod \mu$ for all $m, m' \in M$. With the partial order $\leq$, we use $\vee$ and $\wedge$ for the join and the meet operations of the lattice $\mathcal{E}(M)$.

Let M and N be sets and $f : M \to 2^N$ be a function. Also assume $\varphi \in \mathcal{E}(N)$. The equivalence relation $\varphi \circ f$ on M is defined for all $m, m' \in M$:[3]

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

Now let $f_i : M \to 2^M$ be a function, where i ranges over an index set I. Then $S := (M, \{f_i | i \in I\})$ is called a *dynamic system* [10]. $\varphi \in \mathcal{E}(M)$ is called a *quasi-congruence* for S if $\varphi \leq \bigwedge_{i \in I}(\varphi \circ f_i)$. The quasi-congruences for S form a complete upper semilattice of the lattice $\mathcal{E}(M)$ [12].

B. Existence

In this section, the problem discussed in Section III is formally stated and solved for the PS (H, H^{hi}). The set of transitions of the automaton H is denoted $T_H := \{(x, \sigma, x') \in X \times \Sigma \times X | x' = \delta(x, \sigma)\}$. A *relabeling* of H is another automaton $\hat{H}$ with the same states as H, together with a surjective function $r : T_H \to T_{\hat{H}}$ such that for all $x, x' \in X = \hat{X}$ and for all $\sigma \in \Sigma$, we have $r((x, \sigma, x')) = (x, \hat{\sigma}, x')$ for some $\hat{\sigma} \in \hat{\Sigma}$. We refer to r as the *relabeling function*.

We adapt the following result on the prefix-closure function $\mathrm{pre} : \Sigma^* \to 2^{\Sigma^*}$ with $\mathrm{pre}(s) = \overline{\{s\}}$ for $s \in \Sigma^*$ [10].[4] The kernel $\ker p^{\mathrm{hi}}$ of p^{hi} for $L(H)$ is a quasi-congruence for $(L(H), \mathrm{pre})$. If $s, s' \in L(H)$, then $p^{\mathrm{hi}}(s) = p^{\mathrm{hi}}(s') \Rightarrow p^{\mathrm{hi}}(\mathrm{pre}(s)) = p^{\mathrm{hi}}(\mathrm{pre}(s'))$. Also, for any quasi-congruence μ on $(L(H), \mathrm{pre})$, there is a relabeling $r : T_H \to T_{\hat{H}}$ with a natural projection $\hat{p}^{\mathrm{hi}} : \hat{\Sigma}^* \to (\hat{\Sigma}^{\mathrm{hi}})^*$ for $L(\hat{H})$ s.t. $\ker \hat{p}^{\mathrm{hi}} = \mu$.

We can now formalize the problem in Section III.

Problem 4.1: Let H be an automaton with the event alphabet Σ, let $\Sigma^{\mathrm{hi}} \subseteq \Sigma$ be a sub-alphabet, and let $p^{\mathrm{hi}} : \Sigma^* \to (\Sigma^{\mathrm{hi}})^*$ be the natural projection. The problem is to find (i) the coarsest quasi-congruence μ for $(L(H), \mathrm{pre})$ that is finer than p^{hi}, and (ii) a relabeling $\hat{H}$ of H with relabeling function $r : T_H \to T_{\hat{H}}$, and a sub-alphabet $\hat{\Sigma}^{\mathrm{hi}} \subseteq \hat{\Sigma}$ with natural projection $\hat{p}^{\mathrm{hi}} :$

[3]The natural extension of cp_φ to sets is used.

[4]In [10], the result is established for a causal reporter map.

$\hat{\Sigma}^* \to (\hat{\Sigma}^{\mathrm{hi}})^*$ with $\ker \hat{p}^{\mathrm{hi}} = \mu$ for μ from (i), such that the pair $(\hat{H}, \hat{H}^{\mathrm{hi}})$ satisfies condition c.; i.e. it is locally nonblocking and marked string accepting.

Regarding Definition 3.1 and 3.2, two post-sets for languages are needed to find the quasi-congruence in Problem 4.1. The *M-local post-set* contains all extensions of s with at most one event in Σ^{hi}. The *M-msa post-set* maps strings that violate Definition 3.2 to the local post-set of s. The remaining strings are mapped to the empty set.

Definition 4.1 (post-sets): Let H and p^{hi} be as above and let $M \subseteq L(H)$. The *M-local post-set* of $s \in L(H)$ is $\mathrm{lpos}_M(s) := \{u \in (\Sigma - \Sigma^{\mathrm{hi}})^* \Sigma (\Sigma - \Sigma^{\mathrm{hi}})^* | su \in M\}$. The *M-msa post-set* of $s \in L(H)$ is defined as

$$\mathrm{lpos}^{\mathrm{msa}}_M(s) := \begin{cases} \emptyset & \text{if equation (1) holds} \\ & \forall s_{\mathrm{ex}} \in L_{\mathrm{ex}}(p^{\mathrm{hi}}(s)) \text{ s.t. } s \leq s_{\mathrm{ex}} \\ \mathrm{lpos}_M(s) & \text{otherwise} \end{cases}$$

The *marked string accepting (msa)-observer* is introduced for formulating Lemma 4.1. If the map p^{hi} is a $L(H)-$msa-observer for $L(H)$, then the corresponding PS (H, H^{hi}) is locally nonblocking and marked string accepting.

Definition 4.2 (M-MSA-Observer): The natural projection $p_0 : \Sigma^* \to \Sigma_0^*$ with $\Sigma_0 \subseteq \Sigma$ is an *M-msa-observer* for the automaton H with $M \subseteq L(H)$ if $\ker p_0$ is a quasi-congruence for $(L(H), \mathrm{pre})$, $(L(H), \mathrm{lpos}_M)$ and $(L(H), \mathrm{lpos}^{\mathrm{msa}}_M)$.

Lemma 4.1 (MSA and LNB): Let H, Σ^{hi} and p^{hi} be as in Problem 4.1. The natural projection p^{hi} is a $L(H)$-msa-observer for H if and only if (H, H^{hi}) is locally nonblocking and marked string accepting.

In the light of Lemma 4.1 and Problem 4.1, we want to determine the coarsest quasi-congruence which is finer than the kernel $\ker p^{\mathrm{hi}}_{\mathrm{in}}$ of an initial natural projection $p^{\mathrm{hi}}_{\mathrm{in}}$.

$$\begin{aligned} \pi^{\star}_{\mathrm{msa}} \;:=\; & \sup\{\pi \in \mathcal{E}(L(H)) | \pi \leq (\ker p^{\mathrm{hi}}_{\mathrm{in}}) \wedge \\ & (\pi \circ \mathrm{pre}) \wedge (\pi \circ \mathrm{lpos}_{L(H)}) \wedge (\pi \circ \mathrm{lpos}^{\mathrm{msa}}_{L(H)})\}. \end{aligned} \tag{2}$$

The supremal element $\pi^{\star}_{\mathrm{msa}}$ exists as the quasi-congruences form a complete upper semilattice of the lattice $\mathcal{E}(L(H))$.

Theorem 4.1: $\mu = \pi^{\star}_{\mathrm{msa}}$ in Equation (2) is the quasi-congruence which solves Problem 4.1 (i).

The above theorem extends the theory of observers discussed in [10]. It can be shown that if the natural projection p^{hi} is an $L_{\mathrm{m}}(H)$-observer, then it is also an $L(H)$-msa-observer, while the converse implication does not hold.

C. Algorithmic Computation

We now turn to the construction of an msa-observer to fulfill part (ii) of Problem 4.1. The algorithm below is adapted from an iterative procedure in [10].

Let μ be an equivalence relation on the state set X of H with the quotient set $Y := X/\mu$ and the associated canonical projection $\mathrm{cp}_\mu : X \to Y$. The initial state and the marked states in the quotient are $y_0 = \mathrm{cp}_\mu(x_0)$ and $Y_{\mathrm{m}} = \mathrm{cp}_\mu(X_{\mathrm{m}})$, respectively. Also let $\Sigma^{\mathrm{hi}} \subseteq \Sigma$ and $\sigma_0 \notin \Sigma$ be an additional label. We call $H_{\mu,\Sigma^{\mathrm{hi}}} := (Y, \Sigma^{\mathrm{hi}} \cup \{\sigma_0\}, \nu, y_0, Y_{\mathrm{m}})$ the *quotient automaton* of H for Σ^{hi} and μ, where the induced transition function $\nu : Y \times (\Sigma^{\mathrm{hi}} \cup \{\sigma_0\}) \to 2^Y$ on the quotient is

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

In order to determine the msa-observer and similar to the post-sets in Definition 4.1, the *successor event transition function* and the *nonmarked transition function* are used.

Definition 4.3: Let H and $\Sigma^{\mathrm{hi}} \subseteq \Sigma$ be as above. Let $x = \delta(x_0, s)$ for $s \in L(H)$. The *successor event transition function* $\Delta_\sigma : X \to 2^X$ is defined for $\sigma \in \Sigma^{\mathrm{hi}}$ as

$$\Delta_\sigma(x) := \{\delta(x,u) | u \in \mathrm{lpos}_{L(H)}(s) \cap (\Sigma - \Sigma^{\mathrm{hi}})^* \sigma (\Sigma - \Sigma^{\mathrm{hi}})^*\}.$$

The *nonmarked transition function* $\Delta_{\mathrm{nm}} : X \to 2^X$ is

$$\Delta_{\mathrm{nm}}(x) := \begin{cases} \bigcup\limits_{\sigma \in \Sigma^{\mathrm{hi}}} \Delta_\sigma(x) & \text{if } \mathrm{lpos}^{\mathrm{msa}}_{L(H)}(s) \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

We define the dynamic system $\tilde{H} := (X, \{\Delta_\sigma | \sigma \in \Sigma^{\mathrm{hi}}\} \cup \Delta_{\mathrm{nm}})$. The coarsest quasi-congruence $\mu_{\tilde{H}}$ for $\tilde{H}$ is

$$\mu_{\tilde{H}} := \sup\{\mu \in \mathcal{E}(X) | \mu \leq \bigwedge_{\sigma \in \Sigma^{\mathrm{hi}} \cup \{\mathrm{nm}\}} (\mu \circ \Delta_\sigma)\}. \tag{3}$$

An efficient algorithm for computing $\mu_{\tilde{H}}$ is given in [2]. Based on $\mu_{\tilde{H}}$, Theorem 4.2 establishes the relation between the quotient $H_{\mu_{\tilde{H}},\Sigma^{\mathrm{hi}}}$ and an $L(H)$-msa-observer.[5]

Theorem 4.2: Let H and p^{hi} be given as above and let $\mu_{\tilde{H}}$ be the quasi-congruence in Equation (3). p^{hi} is an $L(H)$-msa-observer iff $H_{\mu_{\tilde{H}},\Sigma^{\mathrm{hi}}}$ is deterministic and contains no σ_0 transitions. In this case, $H_{\mu_{\tilde{H}},\Sigma^{\mathrm{hi}}}$ is a minimal state recognizer of $p^{\mathrm{hi}}(L_{\mathrm{m}}(H))$ and can be computed in polynomial time.

The remaining question is how to proceed if $H_{\mu_{\tilde{H}},\Sigma^{\mathrm{hi}}}$ is nondeterministic or has σ_0 transitions. Algorithm 4.1 solves this problem by relabeling transitions in H using $H_{\mu_{\tilde{H}},\Sigma^{\mathrm{hi}}}$.

Algorithm 4.1 (MSA-Observer): **Input:** H, Σ^{hi}.

1. compute $\mu_{\tilde{H}}$ according to Equation (3).
2. compute $H_{\mu_{\tilde{H}},\Sigma^{\mathrm{hi}}}$.
3. **if** $H_{\mu_{\tilde{H}},\Sigma^{\mathrm{hi}}}$ is deterministic and has no σ_0-transitions
   - $\hat{H} = H$, $\hat{\Sigma}^{\mathrm{hi}} = \Sigma^{\mathrm{hi}}$; **terminate**.

   else
   - $(\hat{H}, \hat{\Sigma}^{\mathrm{hi}}) = \mathrm{relabel}_{\mu_{\tilde{H}}}(H, H_{\mu_{\tilde{H}},\Sigma^{\mathrm{hi}}}, \Sigma^{\mathrm{hi}})$
   - $H = \hat{H}$, $\Sigma^{\mathrm{hi}} = \hat{\Sigma}^{\mathrm{hi}}$; **go to** Step 1.

Output: $\hat{H}, \hat{\Sigma}^{\mathrm{hi}}$.

The relabeling function $\mathrm{relabel}_{\mu_{\tilde{H}}}(H, H_{\mu_{\tilde{H}},\Sigma^{\mathrm{hi}}}, \Sigma^{\mathrm{hi}})$ is implemented by the following algorithm.

[5] The results of this section are proven in [8].

Algorithm 4.2 (relabeling): **Input:** H, $H_{\mu_{\tilde{H}},\Sigma^{\mathrm{hi}}}$, Σ^{hi}.

1. $\overline{r}: T_{H_{\mu_{\tilde{H}},\Sigma^{\mathrm{hi}}}} \to T_{\hat{H}_{\mu_{\tilde{H}},\hat{\Sigma}^{\mathrm{hi}}}}$ relabels $H_{\mu_{\tilde{H}},\Sigma^{\mathrm{hi}}}$ to $\hat{H}_{\mu_{\tilde{H}},\hat{\Sigma}^{\mathrm{hi}}}$ over $\hat{\Sigma}^{\mathrm{hi}}$ with the following restrictions:
   - $\overline{r}((y,\sigma,y')) = (y,\hat{\sigma},y')$ and $\sigma \neq \hat{\sigma} \Rightarrow \hat{\sigma} \notin (\Sigma \cup \{\sigma_0\})$, i.e. always relabel with new labels.
   - if $(y,\hat{\sigma},y') = r(y,\sigma,y')$ and $(z,\hat{\gamma},z') = r(z,\gamma,z')$ with $\sigma \neq \gamma$, then $\hat{\sigma} \neq \hat{\gamma}$, i.e. transitions with different original event labels have different new labels.
2. $r: T_H \to T_{\hat{H}}$ relabels H to $\hat{H}$ according to $\overline{r}$. Assume $(x,\sigma,x') \in T_H$.
   - if $\sigma \in \Sigma^{\mathrm{hi}}$ and $\overline{r}((\mathrm{cp}_{\mu_{\tilde{H}}}(x),\sigma,\mathrm{cp}_{\mu_{\tilde{H}}}(x'))) = (\mathrm{cp}_{\mu_{\tilde{H}}}(x),\hat{\sigma},\mathrm{cp}_{\mu_{\tilde{H}}}(x'))$ with $\sigma \neq \hat{\sigma}$
   $$\Rightarrow r((x,\sigma,x')) = (x,\hat{\sigma},x').$$
   - if $\sigma \notin \Sigma^{\mathrm{hi}}$ and $\overline{r}((\mathrm{cp}_{\mu_{\tilde{H}}}(x),\sigma_0,\mathrm{cp}_{\mu_{\tilde{H}}}(x'))) = (\mathrm{cp}_{\mu_{\tilde{H}}}(x),\hat{\sigma},\mathrm{cp}_{\mu_{\tilde{H}}}(x'))$
   $$\Rightarrow r((x,\sigma,x')) = (x,\hat{\sigma},x').$$

Output: $\hat{H}$, $\hat{\Sigma}^{\mathrm{hi}}$.

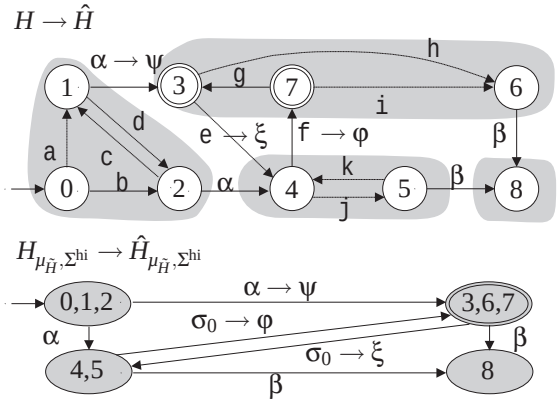

Fig. 3. Illustration of the msa-observer algorithm

The application of Algorithm 4.1 results in the main theorem of this section. Given an automaton H and a high-level alphabet Σ^{hi}, the observer algorithm returns a natural projection $\hat{p}^{\mathrm{hi}}$ for the relabeled automaton $\hat{H}$ such that $(\hat{H},\hat{H}^{\mathrm{hi}})$ is locally nonblocking and marked string accepting.

Theorem 4.3: Algorithm 4.1 with H and Σ^{hi} terminates in at most $|X|$ steps. If the algorithm stops with the automaton $\hat{H}$ and the alphabet $\hat{\Sigma}^{\mathrm{hi}}$, then the kernel of the natural projection $\hat{p}^{\mathrm{hi}}$ for $L(\hat{H})$ satisfies $\ker \hat{p}^{\mathrm{hi}} = \pi^{\star}_{\mathrm{msa}}$.

Example 4.1: Let H be as in Figure 3 with the high-level alphabet $\Sigma^{\mathrm{hi}} = \{\alpha,\beta\}$. We follow the procedure in Algorithm 4.1. The quasi-congruence $\mu_{\tilde{H}}$ in (3) evaluates to $\mu_{\tilde{H}} = \{\{0,1,2\},\{3,6,7\},\{4,5\},\{8\}\}$ (for example compare $\Delta_{\mathrm{nm}}(3) = \Delta_{\mathrm{nm}}(6) = \Delta_{\mathrm{nm}}(7) = \emptyset$ and $\Delta_{\mathrm{nm}}(4) = \Delta_{\mathrm{nm}}(5) = \{8\}$). The quotient automaton $H_{\mu_{\tilde{H}},\Sigma^{\mathrm{hi}}}$ is shown in Figure 3. It has a nondeterministic transition α in state (0,1,2) and two σ_0-transitions. Thus, the corresponding transitions must be relabled in $H_{\mu_{\tilde{H}},\Sigma^{\mathrm{hi}}}$ and in H according to Algorithm 4.2. As an example, we choose $\overline{r}\big(((0,1,2),\alpha,(3,6,7))\big) = ((0,1,2),\psi,(3,6,7))$ and thus $r\big((1,\alpha,3)\big) = (1,\psi,3)$. The resulting PS $(\hat{H},\hat{H}^{\mathrm{hi}})$ with the high-level alphabet $\hat{\Sigma} = \{\alpha,\beta,\varphi,\xi,\psi\}$ is equal to the PS (H,H^{hi}) in Example 3.1. Thus, after one more iteration, the observer algorithm terminates with the solution $(\hat{H},\hat{\Sigma}^{\mathrm{hi}})$ in Example 3.1.

V. Consistent relabeling of Decentralized DES

The algorithms in Section IV-C provide a method to compute a relabeling and a locally nonblocking and marked string accepting natural projection for a single PS (H,H^{hi}). As the control architecture introduced in Section III involves *decentralized projected systems* (DPS) $(||_{i=1}^{n} G_i, ||_{i=1}^{n} G_i^{\mathrm{hi}})$, the effect of relabeling one automaton G_k, $1 \leq k \leq n$, on the overall synchronous behavior has to be investigated. To this end, consider a transition $q_k = (x_1,\sigma,x_2) \in T_{G_k}$ which is relabeled to (x_1,τ,x_2) in $T_{\hat{G}_k}$, i.e. $r_k((x_1,\tau,x_2)) = q_k$. If σ is not contained in any of the other alphabets, that is $\sigma \notin \Sigma_i$ for all $i \neq k$, there is no effect on the other subsystems as σ occurs asynchronously. In case that $\sigma \in \Sigma_i$ for some $i \neq k$, a relabeling of σ in T_{G_k} changes the synchronous behavior of the decentralized subsystems. We can bypass this problem by adding a new transition containing the event τ for any transition containing σ in the subsystems G_i, $i \neq k$. The following definitions formalize this idea.

Definition 5.1: Let G_k be an automaton with the relabeled automaton $\hat{G}_k$. The map $R_k: \hat{\Sigma}_k \to \Sigma_k$ is defined as

$$R_k(\tau) \quad = \quad \begin{cases} \sigma & \text{if } \exists q_k = (x_1,\tau,x_2) \in T_{\hat{G}_k} \text{ s.t.} \\ & r_k(q) = (x_1,\sigma,x_2) \neq q, \\ \tau & \text{otherwise.} \end{cases}$$

The function R_k denotes the map from the relabeled events to their original events. $\overline{R}_k: \hat{\Sigma}_k^{*} \to \Sigma_k^{*}$ is the extension of R_k to strings with $\overline{R}_k(\varepsilon) = \varepsilon$ and $\overline{R}_k(\hat{s}\tau) = \overline{R}_k(\hat{s})R_k(\tau)$ for $\hat{s} \in \hat{\Sigma}_k^{*}$ and $\tau \in \hat{\Sigma}_k$.

Definition 5.2 (Consistent relabeling): Let $(||_{i=1}^{n} G_i, ||_{i=1}^{n} G_i^{\mathrm{hi}})$ be a DPS and let $\hat{G}_k$ be a relabeling of G_k with R_k according to Definition 5.1 and the high-level alphabet $\hat{\Sigma}_k^{\mathrm{hi}}$.[6] The tuple $(\hat{G}_i,\hat{\Sigma}_i^{\mathrm{hi}})$, $i \neq k$ is a *consistent relabeling* of $(G_i,\Sigma_i^{\mathrm{hi}})$ w.r.t. $(\hat{G}_k,\hat{\Sigma}_k^{\mathrm{hi}})$ if (i) $\hat{\Sigma}_i^{\mathrm{hi}} = \Sigma_i^{\mathrm{hi}} \cup \{\tau \in \hat{\Sigma}_k^{\mathrm{hi}} | R_k(\tau) \in \Sigma_i\}$ and (ii) for all $\tau \in \hat{\Sigma}_k$ and $\forall q_i \in T_{G_i}$ such that $q_i = (x_1,R_k(\tau),x_2)$, it holds that $(x_1,\tau,x_2) \in T_{\hat{G}_i}$. The DPS $(||_{i=1}^{n} \hat{G}_i, ||_{i=1}^{n} \hat{G}_i^{\mathrm{hi}})$ is a consistent relabeling of $(||_{i=1}^{n} G_i, ||_{i=1}^{n} G_i^{\mathrm{hi}})$ w.r.t. $(\hat{G}_k,\hat{\Sigma}_k^{\mathrm{hi}})$ if each tuple $(\hat{G}_i,\hat{\Sigma}_i^{\mathrm{hi}})$, $i \neq k$ is a consistent relabeling of $(G_i,\Sigma_i^{\mathrm{hi}})$ w.r.t. $(\hat{G}_k,\hat{\Sigma}_k^{\mathrm{hi}})$.

It is readily observed, that for all $i = 1,\ldots,n$, it is true that $\overline{R}_k(L(\hat{G}_i)) = L(G_i)$. Yet, it has to be shown that the

[6] The corresponding natural projection is $\hat{p}_i^{\mathrm{dec}}: \hat{\Sigma}_i^{*} \to (\hat{\Sigma}_i^{\mathrm{hi}})^{*}$.

synchronous behavior of the decentralized systems is not changed by the consistent relabeling. Lemma 5.1 provides this result.

Lemma 5.1 (Consistent relabeling): Let $(||_{i=1}^{n}\hat{G}_i, ||_{i=1}^{n}\hat{G}_i^{\mathrm{hi}})$ be a consistent relabeling of $(||_{i=1}^{n}G_i, ||_{i=1}^{n}G_i^{\mathrm{hi}})$ w.r.t. $(\hat{G}_k, \hat{\Sigma}_k^{\mathrm{hi}})$ and define the natural projections $p_i : \Sigma^* \to \Sigma_i^*$ and $\hat{p}_i : \hat{\Sigma}^* \to \hat{\Sigma}_i^*$. Then

$$\overline{R}_k(L(\hat{G})) = \overline{R}_k(||_{i=1}^{n}L(\hat{G}_i)) = ||_{i=1}^{n}L(G_i) = L(G), \quad (4)$$

$$\overline{R}_k(L(\hat{G}^{\mathrm{hi}})) = \overline{R}_k(||_{i=1}^{n}L(\hat{G}_i^{\mathrm{hi}})) = ||_{i=1}^{n}L(G_i^{\mathrm{hi}}) = L(H^{\mathrm{hi}}). \quad (5)$$

The same equivalence holds for the respective marked languages.

A further beneficial property of the consistent relabeling is stated in Lemma 5.2. Besides the language equivalence, also the locally nonblocking and marked string accepting condition are preserved.

Lemma 5.2: Let $(||_{i=1}^{n}\hat{G}_i, ||_{i=1}^{n}\hat{G}_i^{\mathrm{hi}})$ be a consistent relabeling of $(||_{i=1}^{n}G_i, ||_{i=1}^{n}G_i^{\mathrm{hi}})$ w.r.t. $(\hat{G}_k, \hat{\Sigma}_k^{\mathrm{hi}})$. If the projected system (G_i, G_i^{hi}) is marked string accepting and locally nonblocking, then the projected system $(\hat{G}_i, \hat{G}_i^{\mathrm{hi}})$ is also marked string accepting and locally nonblocking.

Using Lemma 5.1 and Lemma 5.2, we develop an iterative relabeling algorithm. As stated in Theorem 5.1, it results in a DPS which is suitable for hierarchical and decentralized control according to [7].

Algorithm 5.1 (Decentralized relabeling):
Input: $(||_{i=1}^{n}G_i, ||_{i=1}^{n}G_i^{\mathrm{hi}})$

1. Initialize $k = 0$.
2. $k := k+1$,
   compute $L(G_k)$-msa-observer $\hat{p}_k^{\mathrm{hi}}$ for $(\hat{G}_k, \hat{G}_k^{\mathrm{hi}})$ from (G_k, G_k^{hi}) using Algorithm 4.1,
   determine $\overline{R}_k$ as in Definition 5.1.
3. compute $(||_{i=1}^{n}\hat{G}_i, ||_{i=1}^{n}\hat{G}_i^{\mathrm{hi}})$ as consistent relabeling of $(||_{i=1}^{n}G_i, ||_{i=1}^{n}G_i^{\mathrm{hi}})$ w.r.t. $(\hat{G}_k, \hat{\Sigma}_k^{\mathrm{hi}})$ according to Definition 5.2.
4. **if** $k = n$
   - **terminate**

   else
   - $(||_{i=1}^{n}G_i, ||_{i=1}^{n}G_i^{\mathrm{hi}}) := (||_{i=1}^{n}\hat{G}_i, ||_{i=1}^{n}\hat{G}_i^{\mathrm{hi}})$.
   - **go to** step 2.

Output: $(||_{i=1}^{n}\hat{G}_i, ||_{i=1}^{n}\hat{G}_i^{\mathrm{hi}})$, $\{\overline{R}_1, \ldots, \overline{R}_n\}$.

Theorem 5.1: Let $(||_{i=1}^{n}G_i, ||_{i=1}^{n}G_i^{\mathrm{hi}})$ be a DPS and let $(||_{i=1}^{n}\hat{G}_i, ||_{i=1}^{n}\hat{G}_i^{\mathrm{hi}})$ be the output of Algorithm 5.1 applied to $(||_{i=1}^{n}G_i, ||_{i=1}^{n}G_i^{\mathrm{hi}})$. Then all projected systems $(\hat{G}_i, \hat{G}_i^{\mathrm{hi}})$ are marked string accepting and locally nonblocking. Additionally, $\overline{R}_1 \circ \cdots \circ \overline{R}_n(L(\hat{G})) = L(G)$ and $\overline{R}_1 \circ \cdots \circ \overline{R}_n(L(\hat{G}^{\mathrm{hi}})) = L(G^{\mathrm{hi}})$.

Theorem 5.1 suggests the following hierarchical control design for decentralized DES $||_{i=1}^{n}G_i$. Starting from the natural projection p_i^{dec} on the set of shared events $\Sigma_i^{\mathrm{hi}} := \bigcup_{j\neq i}(\Sigma_i \cap \Sigma_j)$, Algorithm 5.1 can be applied to the DPS $(||_{i=1}^{n}G_i, ||_{i=1}^{n}G_i^{\mathrm{hi}})$. As all PSs $(\hat{G}_i, \hat{G}_i^{\mathrm{hi}})$ of the resulting DPS $(||_{i=1}^{n}\hat{G}_i, ||_{i=1}^{n}\hat{G}_i^{\mathrm{hi}})$ are locally nonblocking and marked string accepting, the hierarchical and decentralized approach in [7] can be applied.

VI. Conclusions

A hierarchical and decentralized control architecture which reduces the computational complexity of DES controller synthesis for large-scale composed systems was elaborated in [7]. Nonblocking and hierarchically consistent control can be guaranteed if the natural projection used for hierarchical sbstraction is (i) *locally nonblocking* and (ii) *marked string accepting* for each subsystem. In this paper we investigated the problem of automatically determining a natural projection such that (i) and (ii) are fulfilled. To this end, we first provided an algorithm which computes the natural projection with the coarsest equivalence kernel that is finer than that of an initial natural projection for an individual subsystem. In our case, the initial natural projection is given by the natural projection on the *shared events* of the composed system. Using this fact and applying the above method for all subsystems of a given composed system, we developed an algorithm which computes the coarsest hierarchical abstraction complying with the method for large-scale composed systems in [7].

References

[1] A.E.C. da Cunha, J.E.R. Cury, and B.H. Krogh. An assume guarantee reasoning for hierarchical coordination of discrete event systems. *Workshop on Discrete Event Systems*, 2002.

[2] J.-C. Fernandez. An implementation of an efficient algorithm for bisimulation equivalence. *Science of Computer Programming*, 13:219–236, 1990.

[3] B. Gaudin and H. Marchand. Efficient computation of supervisors for loosely synchronous discrete event systems: A state-based approach. *IFAC World Congress*, 2005.

[4] R.J. Leduc. Hierarchical interface based supervisory control. *PhD thesis, Department of Electrical and Computer Engineering, University of Toronto*, 2002.

[5] S-H. Lee and K.C. Wong. Structural decentralised control of concurrent DES. *European Journal of Control*, 35:1125–1134, October 2002.

[6] C. Ma. Nonblocking supervisory control of state tree structures. *Ph.D. Dissertation, Department of Electrical and Computer Engineering, University of Toronto*, 2004.

[7] K. Schmidt. Hierarchical control of decentralized discrete event systems: Theory and application. *Phd-thesis, Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg*, 2005.

[8] K. Schmidt. Computation of marked string accepting observers for discrete event systems. *Technical Report, Lehrstuhl für Regelungstechnik, Universität Erlangen-Nürnberg*, 2006.

[9] K. Schmidt, T. Moor, and S. Perk. A hierarchical architecture for nonblocking control of discrete event systems. *Mediterranean Conference on Control and Automation*, 2005.

[10] K. Wong and W.M. Wonham. On the computation of observers in discrete-event systems. *Discrete Event Dynamic Systems*, 14(1):55–107, 2004.

[11] K.C. Wong and W.M. Wonham. Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 1996.

[12] W.M Wonham. Notes on control of discrete event systems. *Department of Electrical Engineering, University of Toronto*, 2004.