Modular and Decentralized Supervisory Control of Concurrent Discrete Event Systems Using Reduced System Models

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

Hervé Marchand  
IRISA-INRIA Rennes  
Campus univ de Rennes 1  
35042 Rennes, France  
herve.marchand@irisa.fr

Benoit Gaudin  
FOKUS Fraunhofer  
Kaiserin Augusta Allee 31  
10589 Berlin, Germany  
bg@fokus.fraunhofer.de

Abstract—This work investigates the supervisor synthesis for concurrent systems based on reduced system models with the intention of complexity reduction. It is assumed that the expected behavior (specification) is given on a subset of the system alphabet, and the system behavior is reduced to this alphabet. Supervisors are computed for each reduced subsystem employing the modular approach in [5] and the decentralized approach in [8]. Depending on the chosen architecture, we provide sufficient conditions for the consistent implementation of the reduced supervisors for the original system.

Keywords—Concurrent discrete event systems, hierarchical control, modular and decentralized architecture.

I. INTRODUCTION

The main issue in supervisor synthesis for discrete event systems (DES) is the state-space explosion for large-scale systems. Addressing this problem, recent approaches study hierarchical, decentralized and modular methods to reduce the complexity of supervisor synthesis algorithms. In hierarchical architectures [17], [3], [7], [14], [9], [13], 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 structure of concurrent systems (systems modeled by several components) is exploited for decentralized and modular control. In most of the decentralized architectures [15], [10], [1], [11], [6], [8], the methodology is characterized by the fact that the specification (i.e. the expected behavior) can be decomposed according to the structure of the plant. In that case, local modular supervisors operating each concurrent system component individually are implemented, and necessary and sufficient conditions under which the behavior of the controlled plant corresponds to the supremal one are given. In contrast, the authors of [4], [5] consider a modular architecture. The specification does not need to be separable (but locally consistent and prefix-closed, which is not the case for most of the previously mentioned works). Modular supervisors can be computed based on the specification and abstractions of the subsystems so that they solve the supervisory control problem without having to build the whole system.

In this paper, we elaborate two approaches for concurrent systems that both avoid the computation of the overall system and are based on a reduced system model. We assume that the specification is given on a subset of the system alphabet and the behavior of the concurrent systems is reduced to this alphabet. Supervisors are synthesized for the reduced system models using the modular approach in [5] and the decentralized approach in [8]. We provide sufficient conditions for the consistent implementation of the reduced supervisors for the original system.

The outline of the paper is as follows. After providing basic definitions in supervisory control in Section II, we present the setting of the paper in Section III. Section IV and V discuss modular and structural decentralized control for reduced system models, respectively. Conclusions are given in Section VI.

II. PRELIMINARIES

We recall basics from supervisory control theory [18], [2].

For a finite alphabet $\Sigma$, the set of all finite strings over $\Sigma$ is denoted $\Sigma^*$. We write $s_1, s_2 \in \Sigma^*$ for the concatenation of two strings $s_1, s_2 \in \Sigma^*$. We write $s_1 \leq s$ when $s_1$ is a prefix of $s$, i.e. if there exists a string $s_2 \in \Sigma^*$ with $s = s_1 s_2$. The empty string is denoted $\epsilon \in \Sigma^*$, i.e. $s = \epsilon s = s$ for all $s \in \Sigma^*$.

A language over $\Sigma$ is a subset $H \subseteq \Sigma^*$. The prefix closure of $H$ is defined by $\overline{H} := \{s \in \Sigma^* | \exists t \in H \text{ s.t. } s \leq t \}$. A language $H$ is prefix closed if $H = \overline{H}$. Let $H, F \subseteq \Sigma^*$, then $H$ is nonblocking w.r.t. $F$ if $\overline{H} \cap F = \emptyset$.

The natural projection $p_i : \Sigma^* \rightarrow \Sigma_i^*$, $i = 1, 2$, for the (not necessarily disjoint) union $\Sigma = \Sigma_1 \cup \Sigma_2$ is defined iteratively: (i) let $p_i(\epsilon) := \epsilon$; (ii) 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.

A finite automaton is a tuple $G = (X, \Sigma, \delta, x_0, X_m)$, with the finite set of states $X$; the finite alphabet of events $\Sigma$; the partial transition function $\delta : X \times \Sigma \rightarrow X$; the initial state $x_0 \in X$; and the set of marked states $X_m \subseteq X$. We write $\delta(x, \sigma)$! if $\delta$ is defined at $(x, \sigma)$. In order to extend $\delta$ to a partial function on $X \times \Sigma^*$, recursively let $\delta(x, \epsilon) := x$ and $\delta(x, s\sigma) := \delta(\delta(x, s), \sigma)$, whenever both $x' = \delta(x, s)$ and $\delta(x', \sigma)$!.

$\delta(x, \sigma)$! Let $L(G) := \{s \in \Sigma^* | \exists t \in H \subseteq \Sigma^* \text{ s.t. } s \in L(H) \}$ 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 $L_m(G) = L(G)$, i.e. if each string in $L(G)$ is the prefix of a marked string in $L_m(G)$. A formal definition of the synchronous composition of two automata $G_1$ and $G_2$ is given in e.g. [2]. Note that $L(G_1 || G_2) = L(G_1) || L(G_2)$ and $L_m(G_1 || G_2) = L_m(G_1) || L_m(G_2)$.

In a supervisory control context, we write $\Sigma = \Sigma \cup \Sigma_{uc}$, $\Sigma \cap \Sigma_{uc} = 0$, to distinguish controllable ($\Sigma$) and uncontrollable ($\Sigma_{uc}$) events. A control pattern is a set $\gamma, \Sigma_{uc} \subseteq \gamma \subseteq \Sigma$, and the set of all control patterns is denoted $\Gamma \subseteq 2^\Sigma$. A supervisor is a map $S: L(G) \rightarrow \Gamma$, 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 (i) $e \in L(S/G)$ and (ii) $s\sigma \in L(S/G)$ iff $s \in L(S/G), \sigma \in S(s)$ and $\sigma \in L(G)$. Thus, $L(S/G)$ represents the behavior of the closed-loop system. To take into account the marking of $G$, let $L_m(S/G) := L(S/G) \cap L_m(G)$.

A language $H$ is said to be controllable w.r.t. $L(G)$ and $\Sigma_{uc}$ if there exists a supervisor $S$ such that $H = L(S/G)$. The set of all languages that are controllable w.r.t. $L(G)$ is denoted $C(L(G))$ and can be characterized by $C(L(G)) = \{ H \subseteq L(G) \mid \exists S \text{ s.t. } H = L(S/G) \}$. Furthermore, the set $C(L(G))$ is closed under arbitrary union. Hence, for every specification language $E$ there uniquely exists a supremal controllable sublanguage of $E$ w.r.t. $L(G)$ and $\Sigma_{uc}$, which is formally defined as $\kappa_{L(G)}(E, \Sigma_{uc}) := \{ K \in C(L(G)) \mid K \subseteq E \}$. A supervisor $S$ that leads to a closed-loop behavior $\kappa_{L(G)}(E, \Sigma_{uc})$ is said to be maximally permissive. A maximally permissive supervisor $S$ can be implemented as an automaton $S$ that generates $\kappa_{L(G)}(E, \Sigma_{uc})$ such that $L(S/G) = L(S) || L(G)$. The latter can be computed from $G$ and a generator of $E$. The notion of controllability is extended by the notion of partial controllability [4]. Let $M \subseteq L(G)$ be a prefix-closed language and let $\Sigma'_{uc} \subseteq \Sigma_{uc}$. The language $M' \subseteq M$ is partially controllable w.r.t. $M, L(G), \Sigma_{uc}$ and $\Sigma'_{uc}$ if (i) $M'$ is controllable w.r.t. $\Sigma'_{uc}$ and $L(G)$ and (ii) $M'$ is controllable w.r.t. $\Sigma_{uc}$ and $M$. The unique supremal partially controllable sublanguage w.r.t. $M, L(G), \Sigma_{uc}$ and $\Sigma_{uc}$ is defined as $M'^{pc} = \kappa_{M}(\kappa_{L(G)}(M, \Sigma'_{uc}), \Sigma_{uc})$.

A language $E$ is $L_m(G)$-closed if $E \cap L_m(G) = E$ and the set of $L_m(G)$-closed languages is denoted $f_L(G)$. The closed-loop system $S/G$ is nonblocking under maximally permissive supervision for specifications $E \in f_L(G)$.

III. SETTING

As a system model, we consider concurrent DES represented by finite automata $G_i$ $\Sigma \subseteq \Sigma \cup \Sigma_{uc}$ over the corresponding alphabets $\Sigma \subseteq \Sigma \cup \Sigma_{uc}$. Here, $\Sigma_{uc}$ and $\Sigma_{uc}$ denote the uncontrollable and the controllable events, respectively. We assume that all subsystems are directly or indirectly connected to all other subsystems via events from the set $\Sigma_{es} := \cup_{i \neq k} (\Sigma \cap \Sigma_{k})$ of shared events. The global set of shared events is thus given by $\Sigma := \cup_{i \neq k} \Sigma_{es}$.

The overall system model is $\hat{G} := ||G_i$ over the alphabet $\hat{\Sigma} := \cup_{i \neq k} \Sigma_i$. Moreover, we assume that the components that share an event agree on the control status of this event, i.e. $\forall i, k, \Sigma_{i, uc} \cap \Sigma_{k, uc} = 0$. Under this hypothesis, we have that $\Sigma_{uc} = \cup_{i \neq k} \Sigma_{i, uc}$ and $\hat{\Sigma} = \cup_{i \neq k} \Sigma_{i, uc}$.

The main objective of this paper is to study control architectures which reduce the computational complexity of supervisor synthesis for a given specification $K \subseteq \Sigma^*$ by avoiding the computation of $\hat{G}$. To this end, we are interested in the case where the complexity of the specification $K$ is lower than that of the plant $\hat{G}$. In the literature, there are different approaches tackling this problem.

An approach for the modular control of concurrent systems is proposed in [4, 5]. Modular supervisors are computed using abstractions of the decentralized subsystems and corresponding local specifications. The supremal partially controllable sublanguages of the local specifications solve the supervisory control problem if the specification $K$ is locally consistent and prefix-closed, and the languages of the subsystems are mutually controllable.

The method in [8] suggests structural decentralized control. It requires the specification $K$ to be separable, i.e. $K = ||p_{\hat{S}}(\hat{K})$, where $p_{\hat{S}}: \hat{\Sigma} \rightarrow \Sigma^*$ is the natural projection. If the languages of the subsystems are mutually controllable and shared event marking,1 then using nonblocking local controllers for the specifications $p_{\hat{S}}(\hat{K})$ is equivalent to the nonblocking overall supervisor.

This approach is supplemented with hierarchical control in [14]. Monolithic control is applied to a reduced (hierarchical) system model which is derived by projecting the behavior of the original model to the set of shared events $\Sigma$. However, this approach requires the computation of an overall reduced system model which is not always feasible.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{architecture.png}
\caption{Modular/Decentralized Architecture}
\end{figure}

Motivated by these considerations, we elaborate two methods that employ reduced concurrent system models, but avoid computing an overall reduced system model. To this end, we investigate the case where the specification $K \subseteq (\Sigma)^*$ for the supervisory control problem is given on a reduced

1Definitions of these notions are given in Section IV and V.
alphabet $\Sigma \subseteq \tilde{\Sigma}$ with $\Sigma_1 \subseteq \Sigma$.\footnote{This assumption is no restriction. If $\Sigma_1 - \Sigma \neq \emptyset$, $K' = K\setminus (\Sigma_1 - \Sigma)$ fulfills the requirement.} Hence, the reduction is based on projecting out events that occur in only one subsystem. With the reduced decentralized alphabets $\Sigma_i := \Sigma \cap \tilde{\Sigma}_i$ and the decentralized natural projections $p_i^{\text{dec}} : \tilde{\Sigma}_i \to \Sigma_i$, the decentralized reduced system models are $(G_i)_{1 \leq i \leq n}$, where $L(G_i) = p_i^{\text{dec}}(L(\tilde{G}_i))$ and $L_m(G_i) = p_i^{\text{dec}}(L_m(\tilde{G}_i))$.

In the following sections, we utilize the approaches in [5] and [8] to design supervisors for the reduced system models. Based on these supervisors, we provide conditions for the decentralized supervisor implementation $\tilde{S}_i$ for the original systems. The first approach results in an estimation of the supremal controllable and nonblocking sublanguages of modular supervisors implementing the supremal partially non-separable specification. The second method provides an estimation of the supremal controllable and nonblocking sublanguage of a not necessarily prefix-closed but separable specification. Figure 1 illustrates the control scheme.

IV. MODULAR CONTROL

According to [5], modular supervisors $S_i^{-1} : \Sigma^* \to \Gamma_i$ with the set of control patterns $\Gamma_i := \{ \gamma \subseteq \Sigma \Sigma_{i,\text{dec}} \subseteq \gamma \}$ are computed for the abstractions $G_i^{-1}$ of the reduced system models and the local specifications $K_i^{-1} := K \cap L(G_i^{-1})$, where $L(G_i^{-1}) = p_i^{-1}(L(G_i)) \subseteq \Sigma^*$, with the natural projection $p_i : \Sigma^* \to \Sigma_i$. The main result of [5] is based on the following definitions.

**Definition 4.1:** $G_i$ and $G_k$ are mutually controllable if
1) $L(G_i)(\Sigma_{i,\text{dec}} \cap \Sigma_k) \cap p_i^{-1}(p_k^{-1}(L(G_k))) \subseteq L(G_i)$
2) $L(G_k)(\Sigma_{k,\text{dec}} \cap \Sigma_k) \cap p_k^{-1}(p_i^{-1}(L(G_i))) \subseteq L(G_k)$

where $p_i : \Sigma^* \to \Sigma_i$ and $p_k : \Sigma^* \to \Sigma_k$.

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

**Definition 4.2 (Local consistency):** A specification $K = \mathbb{K}$ is said to be locally consistent w.r.t. $\Sigma_{i,\text{dec}}$ and $(L(G_i))_{1 \leq i \leq n}$, if for any $i$ we have: $\forall s \in K_i^{-1}$ and $\forall u \in \Sigma_{i,\text{dec}}$ such that $su \in K_i^{-1}$ and $\forall v \in \Sigma_i^*$ it holds that $sp_i(u)v \in K_i^{-1} \Rightarrow suv \in K_i^{-1}$. Based on the above definitions, it holds that the computation of modular supervisors implementing the supremal partially controllable sublanguages of $K_i^{-1}$ is equivalent to the monolithic supervisor for the specification $K$.

**Theorem 4.1 (Supervisor Computation [5]):** Let $(G_i)_{1 \leq i \leq n}$ be mutually controllable\footnote{It can be shown that the condition of global mutual controllability required in [5] is equivalent to mutual controllability.} and assume that $\forall i,k,\Sigma_{i,\text{dec}} \cap \Sigma_{k,\text{dec}} = \emptyset$. If the specification $K = \mathbb{K} \subseteq \Sigma^*$ is locally consistent w.r.t. $\Sigma_{i,\text{dec}}$ and $(L(G_i))_{1 \leq i \leq n}$, then

$$\bigcap (K_i^{-1})^{\text{pc}} = K \cap L(G_i)(\Sigma_{i,\text{dec}}).$$

Using the concept of modularity [4], the overall supervisor $S^{-1} : \Sigma^* \to \Gamma$ with $\Gamma := \{ \gamma \subseteq \Sigma \Sigma_{i,\text{dec}} \subseteq \gamma \}$ and $L(S^{-1})/G = \bigcap (K_i^{-1})^{\text{pc}}$ can now be implemented as the intersection of the control actions of the modular supervisors $S_i^{-1}$ with $L(S_i^{-1}/G_i^{-1}) = (K_i^{-1})^{\text{pc}}$ (see Figure 2).

However, the supervisors $S_i^{-1}$ are computed based on the reduced system model. In what follows, we provide an implementation of these supervisors for the original systems. For this purpose, $p_i(L(S_i^{-1}/G_i^{-1}))$ is used as an approximation of the modular closed-loop behavior $L(S^{-1}/G) = \|L(S_i^{-1}/G_i^{-1})\|$ projected on the reduced subalphabets $\Sigma_i$. As shown in the next lemma, $p_i(L(S_i^{-1}/G_i^{-1}))$ is controllable w.r.t. $L(G_i)$ and $\Sigma_{i,\text{dec}}$. Thus it can be enforced by a supervisor for $G_i$.

![Modular architecture](image)

**Lemma 4.1:** With the preceding notations, $p_i(L(S_i^{-1}/G_i^{-1}))$ is controllable w.r.t. $L(G_i)$ and $\Sigma_{i,\text{dec}}$.

**Proof:** Let us consider

$$s\sigma \in p_i(L(S_i^{-1}/G_i^{-1})) \Sigma_{i,\text{dec}} \cap L(G_i)$$

and $s' \in L(S_i^{-1}/G_i^{-1})$ such that $p_i(s') = s$. Since $s\sigma \in L(G_i)$ and $\sigma \in \Sigma_i$, it is also true that $p_i(s\sigma) \in L(G_i)$. This entails that $s'\sigma \in L(G_i^{-1})$ and

$$s'\sigma \in L(S_i^{-1}/G_i^{-1}) \Sigma_{i,\text{dec}} \cap L(G_i^{-1}).$$

But $L(S_i^{-1}/G_i^{-1})$ is controllable w.r.t. $\Sigma_{i,\text{dec}}$ and $L(G_i^{-1})$ since it is partially controllable. Therefore, $s'\sigma \in L(S_i^{-1}/G_i^{-1})$ and $p_i(s'\sigma) = s\sigma \in p_i(L(S_i^{-1}/G_i^{-1}))$, which concludes the proof.

Now as $\forall i$, $p_i(L(S_i^{-1}/G_i^{-1})) \subseteq L(G_i)$ is controllable w.r.t. $L(G_i)$ and $\Sigma_{i,\text{dec}}$, there exist $n$ supervisors $(S_i)_{1 \leq i \leq n}$ such that

$$L(S_i/G_i) = p_i(L(S_i^{-1}/G_i^{-1})).$$

Based on the results in [12], an admissible supervisor for the original system is given with the consistent implementations $\tilde{S}_i : \tilde{\Sigma}_i \to \tilde{\Gamma}_i$ (see [12]) of the decentralized reduced supervisors $S_i$. It is defined for $s \in L(\tilde{G}_i)$ as $\tilde{S}_i(s) := S_i(p_i(s)) \cup (\tilde{\Sigma}_i - \Sigma_i)$. Note the equality $p_i^{\text{dec}}(L(\tilde{S}_i/G_i)) = L(S_i/G_i)$.

Combining the steps described above, the main result of this section can be stated.

**Theorem 4.2:** Recalling that $\Sigma_i \subseteq \Sigma$ and with the notation from above, the supervisor implementation

$$L(\tilde{S}/\tilde{G}) = (\|L(S_i^{-1}/G_i^{-1})\|, \|L(\tilde{S}_i/G_i)\|)$$

leads to consistent control of the original system, i.e.

$$p(L(\tilde{S}/\tilde{G})) = L(S^{-1}/G),$$

$$L(\tilde{S}/\tilde{G}) \subseteq p^{-1}(K),$$

where $p : \tilde{\Sigma}^* \to \Sigma^*$.

The following Lemma aids the proof of Theorem 4.2.
Lemma 4.2 ([14]): Let \((L_i)_{1 \leq i \leq n}\) be languages over the respective alphabets \(\Sigma_i\). Assume that \(\Sigma_0 \subseteq \bigcup \Sigma_i\) and \(\bigcup_{i \neq k} (\Sigma_i \cap \Sigma_k) \subseteq \Sigma_0\) with the natural projections \(p_0 : (\bigcup \Sigma_i)^* \to \Sigma_0\) and \(p_i : \Sigma_i \to (\Sigma_i \cap \Sigma_0)^*\) for \(i = 1, \ldots, n\). Then
\[
p_0(||L_i||) = ||p_i(L_i)||.
\]

Proof of Theorem 4.2:
Now Theorem 4.2 can be proven. Because of Lemma 4.2,
\[
p(L(\tilde{S}/\tilde{G})) = p(L(S_i^{-1}/G_i)) ||p(L(S_i/G_i)) = (L(S_i^{-1}/G_i)) ||p(L(S_i/G_i)) = (||L(S_i^{-1}/G_i'||) ||p_{dec}(L(S_i/G_i))).
\]
This can be written as \(\forall (\bigcup \Sigma_i) = \bigcup (\Sigma_i \cap \Sigma_0)^*\) with \(p_{dec}(L(S_i/G_i)) = L(S_i/G_i)\).

Now according to Lemma 4.1, the previous equation can be rearranged as
\[
p(L(\tilde{S}/\tilde{G})) = (||L(S_i^{-1}/G_i')||) ||p_1(L(S_i/G_i)) = (||L(S_i^{-1}/G_i')||).
\]

Moreover, the supervisor computation implies that \(\|L(S_i^{-1}/G_i')\| \subseteq K\). This means that \(L(\tilde{S}/\tilde{G}) \subseteq p^{-1}(K)\). □

The reduced modular architecture is shown in Figure 3. The control actions of the decentralized supervisors for the original systems evaluate to \(\tilde{S}_i(s_i) = p_i(S_i^{-1}(s_i)) \cup (\tilde{S}_i - \Sigma_i)\).

![Fig. 3. Reduced modular architecture](image)

So far, our method allows to perform local computations instead of building a single finite automaton for solving the supervisory control problem. However, only the case of prefix-closed specifications was considered. Based on a decentralized architecture, we now give sufficient conditions under which a nonblocking solution can be computed whenever the reduced specification is separable.

V. STRUCTURAL DECENTRALIZED CONTROL

Consider a concurrent system given by a set of nonblocking decentralized systems \((\tilde{G}_i)_{1 \leq i \leq n}\) (i.e., \(\forall i, \tilde{L}_m(\tilde{G}_i) = L(\tilde{G}_i)\)). It follows that the reduced system models \(\tilde{G}_i\) are also nonblocking. We assume that the specification \(K \subseteq \Sigma^*\) over the subalphabet \(\Sigma\) is separable, i.e., \(K = ||p_i(K)||\), where \(p_i : \Sigma^* \to \Sigma_i^*\) and each local specification \(K_i := p_i(K)\) is \(L_m(G_i)\)-closed, i.e., \(K_i \in \mathcal{F}_{L_m(G_i)}\). Our aim is to use the methodology of [8] to compute nonblocking decentralized supervisors acting upon the subsystems \(G_i\) and to implement these supervisors for the original system \(G = ||G_i||\).

We first formally describe the approach in [8] and then provide new results that are useful in our setting.

Definition 5.1: Let \(\Sigma' \subseteq \Sigma\) and \(H \subseteq \Sigma'\), then \(H\) marks \(\Sigma'\) whenever \(\Sigma' \cap \Sigma \subseteq H\).

Using the above definition combined with mutual controllability, Theorem 5.1 follows. The structural decentralized architecture is illustrated in Figure 4.

![Fig. 4. Structural decentralized architecture](image)

In addition to this result due to [8], one can prove that the overall closed-loop behavior is actually nonblocking. To do so, we first show that whenever the local specifications \(K_i\) are \(L_m(G_i)\)-closed then so is the global specification \(K\) with respect to the reduced plant \(G\).

Lemma 5.1: Let \((G_i)_{1 \leq i \leq n}\) be the set of decentralized subsystems and \(K_i \in \mathcal{F}_{L_m(G_i)}\). Then \(K = ||K_i|| \in \mathcal{F}_{L_m(G)}\).

Proof: First we clearly have that \(K \subseteq \mathcal{K}\). Now since \(\forall i, K_i \subseteq L_m(G_i)\), it holds that \(||K_i|| \subseteq ||L_m(G_i)||\) which entails that \(K \subseteq L_m(G)\). Therefore \(K \subseteq \mathcal{K} \cap L_m(G)\).

Reciprocally, consider \(s \in \mathcal{K} \cap L_m(G)\). We thus have that \(p_i(s) \in p_i(K)\) and \(p_i(s) \in p_i(L_m(G))\).

- As \(L_m(G) = \bigcap \{p_i^{-1}(L_m(G_i))\}\), this entails that \(p_i(s) \in p_i(p_i^{-1}(L_m(G_i))) = L_m(G_i)\).
- Let us now show that \(p_i(s) \in \mathcal{K}\). First, we have that \(K \subseteq \mathcal{K} = \bigcap p_i^{-1}(K_i) \subseteq \cap p_i^{-1}(K_i) \subseteq p_i^{-1}(K)\).

Also \(p_i(s) \in p_i(K) \subseteq p_i(p_i^{-1}(K)) \subseteq \cap p_i^{-1}(K_i)\), and hence \(p_i(s) \in p_i(p_i^{-1}(K)) = K\).

Overall, \(\forall i, p_i(s) \in \mathcal{K} \cap L_m(G_i) = K_i\) as \(K_i \in \mathcal{F}_{L_m(G_i)}\). Thus \(\forall i, s \in p_i^{-1}(K_i)\) and \(s \in \cap p_i^{-1}(K_i) = K\).

\[\text{Note that it is equivalent to say that } \forall i, L_m(G_i) \text{ marks } \Sigma_i.\]
We now need to show that the behavior of the closed-loop reduced system can be actually obtained by a collection of supervisors each of them acting upon a local decentralized subsystem $G_i$. This is the aim of the next lemma:

**Lemma 5.2:** With the preceding notations, we have that

1) $||\kappa_{L(G)}(K_i)\cap L(G)\parallel = ||\kappa_{L(G)}(K_i)\cap L(G)\parallel$ 
2) $||\kappa_{L(G)}(K_i)\cap L(G)\parallel = ||\kappa_{L(G)}(K_i)\cap L(G)\parallel$ 
3) $||\kappa_{L(G)}(K_i)\cap L(G)\parallel = ||\kappa_{L(G)}(K_i)\cap L(G)\parallel$ 
4) $||\kappa_{L(G)}(K_i)\cap L(G)\parallel = ||\kappa_{L(G)}(K_i)\cap L(G)\parallel$

**Proof:**

1) Due to Theorem 5.1, we have that 

\[ \kappa_{L(G)}(K) = \parallel \kappa_{L(G)}(K_i)\cap L(G)\parallel = \parallel \kappa_{L(G)}(K_i)\cap L(G)\parallel = \parallel \kappa_{L(G)}(K_i)\cap L(G)\parallel = \parallel \kappa_{L(G)}(K_i)\cap L(G)\parallel \]

2) Because of Lemma 5.1, we have 

\[ \kappa_{L(G)}(K) = \kappa_{L(G)}(K_i)\cap L_m(G) = \parallel \kappa_{L(G)}(K_i)\cap L_m(G)\parallel = \parallel \kappa_{L(G)}(K_i)\cap L_m(G)\parallel = \parallel \kappa_{L(G)}(K_i)\cap L_m(G)\parallel \]

With the above lemmas, the existence of a nonblocking supervisor for $G$ can be shown.

**Theorem 5.2:** Under the assumptions of Theorem 5.1, the supervisor $S$ such that $L(S^\parallel G) = \kappa_{L(G)}(K)$ is nonblocking.

**Proof:** Based on Lemma 5.2, we consider nonblocking supervisors $S_i$ such that $L(S_i^\parallel G_i) = \kappa_{L(G)}(K_i)$. It holds that 

\[ \kappa_{L(G)}(K) = \parallel L(S_i^\parallel G_i)\parallel = \parallel L(S_i^\parallel G_i)\parallel \]

Let us now consider $S$ such that $S = ||S_i|\parallel$, where each supervisor $S_i$ is seen as a finite automaton. We have that 

\[ S^\parallel G = S^\parallel G = ||S_i^\parallel ||G_i|| = ||S_i^\parallel ||G_i|| = ||S_i^\parallel ||G_i|| = ||S_i^\parallel ||G_i|| \]

We thus have that $L_m(S^\parallel G) = \kappa_{L(G)}(K)$ and $L(S^\parallel G) = \kappa_{L(G)}(K)$.

Now, based on Lemma 5.1, we know that $K$ is $L_m(G)$-closed. Moreover, $L_m(G)$-closed is preserved under control, which ensures us that the overall closed-loop decentralized system $||S_i^\parallel ||G_i|| = ||S_i^\parallel ||G_i||$ is nonblocking.

**Remark 1:** It is interesting to note that this result gives sufficient conditions under which a concurrent system is nonblocking. Indeed, based on Theorem 5.1 and 5.2, given a concurrent system $G = ||G_i||$, if $L_m(G)$ marks $\Sigma_s$ and $\forall i \neq j$, $G_i$ and $G_j$ are mutually controllable, then $G$ is nonblocking. This gives access to an efficient way to test if a concurrent system is nonblocking.

Next, the implementation of the supervisors computed with respect to the reduced system models for the original system is discussed. We again support the consistent implementation, and investigate two different sets of conditions which guarantee nonblocking and consistent control.

In the first case, the original subsystems have to mark the reduced alphabets in addition to the conditions which are required for nonblocking supervisor synthesis for the reduced system model.

**Theorem 5.3:** Let $K$ be a separable specification and let $S_i$ be such that $L_m(S_i) = \kappa_{L(G)}(p_i(K), \Sigma_{se})$. Assume that $G_i$ and $G_k$ are mutually controllable for $i \neq k$ and $L_m(G_i)$ marks $\Sigma_s$ for all $i = 1, \ldots, n$. If the supervisors $S_i$ are consistent implementations of $S_i$, then the overall supervisor $S$ such that 

\[ L(S^\parallel G) := ||L(S_i)\parallel \]

is nonblocking and consistent.

First we need the following lemma.

**Lemma 5.3 ([12]):** The consistent implementation implies that if $s_i \in L(S_i) = L(G_i)$ and $s_i u_i \in L(G_i)$ for $u_i\in (\Sigma_i - \Sigma)^*$, then $s_i u_i \in L(S_i^\parallel G_i)$. If additionally $s_i u_i \sigma \in L(G_i)$ for $\sigma \in \Sigma_s$ and $p_i^{\text{dec}}(s_i u_i^\sigma) \in L(S_i^\parallel G_i)$, then $s_i u_i \sigma \in L(S_i) = L(G_i)$.

Based on this lemma, the proof of Theorem 5.3 is as follows:

**Proof:** For showing consistency, we observe that 

\[ p(S^\parallel G) = p(L(S_i)\parallel G_i) = p(L(S_i)\parallel G_i) = L(S_i^\parallel G_i) = L(S/G) \]

Also, because of $L(S/G) \subseteq K$, it holds that $L(S/G) \subseteq p^{-1}(K)$.

For proving nonblocking control, it has to be shown that if $s \in L(S/G)$, then $s \in L(S/G)$. Now assume that $s \in L(S/G)$. Then $\bar{p}_i(s) \in L(S_i^\parallel G_i)$ for all $i = 1, \ldots, n$. Suppose that there is no $u_i \in (\Sigma_i - \Sigma)^*$ s.t. $\bar{p}_i(s) u_i \in L(G_i)$. As $G_i$ is nonblocking, there must be a string $v_i = v_i \sigma v_i \in (\Sigma_i - \Sigma)^* \Sigma_i^* \Sigma_s^*$ s.t. $\bar{p}_i(s) v_i \in L(G_i)$.

Based on Lemma 5.3, we also have that $p_i(s) u_i \in L(S_i^\parallel G_i)$, which entails that $s_i u_i \in ||p_i(s) u_i|| \subseteq ||L(S_i^\parallel G_i)|| = L(S/G)$. Thus $s_i u_i \in L(S/G) \parallel G_i = L(S/G) \parallel G_i = L(S/G)$ and $s \in L(S/G)$.

The second case is based on the notion of an $H$-observer.

**Definition 5.2 ($H$-observer):** Let $H \subseteq L \subseteq T \subseteq \Sigma$ be languages and $p : \Sigma^* \to \Sigma^*$ be the natural projection on the alphabet $\Sigma \subseteq \Sigma$. $p$ is called an $H$-observer if $\forall s \in L$ and $\forall \sigma \in (\Sigma \cup \{\epsilon\})$:

\[ p(s) \epsilon \in H \Rightarrow \exists \sigma \in \Sigma^* s.t. s \epsilon \in H \land p(s) \epsilon = p(s) \epsilon \]

In Theorem 5.4, the condition that all events in $\Sigma$ must mark $L_m(G)$ is reduced to the events in $\Sigma_s$. This is compensated by requiring the decentralized projection $p_i^{\text{dec}}$ to be a $L_m(G)$-observer.

**Theorem 5.4:** Let $K$ be a separable specification and let $S_i$ be such that $L_m(S_i) = \kappa_{L(G)}(p_i(K), \Sigma_{se})$. Assume that $G_i$ and $G_k$ are mutually controllable for $i \neq k$ and $L_m(G_i)$ marks $\Sigma_s$. If $p_i^{\text{dec}}$ is a $L_m(G_i)$-observer and the supervisors $S_i$ are consistent implementations of $S_i$, then the overall

\[ p_i^{\text{dec}} \text{ is not a } L_m(G_i)\text{-observer, then [16] provides an algorithm to compute a } L_m(G_i)\text{-observer with the coarsest equivalence kernel possible.} \]
Lemma 5.4: With the assumptions in Theorem 5.4, it holds that if \( s \in L(\tilde{S}_i/\tilde{G}_i) \) and \( p_i^{dec}(s) t \in L_m(S_i/G_i) \) for \( t \in \Sigma_i \), then \( \exists u_i \in \tilde{\Sigma}_i \) s.t. \( su_i \in L_m(\tilde{S}_i/\tilde{G}_i) \) and \( p_i^{dec}(su_i) = p_i^{dec}(s) t \).

Proof: Assume that \( s \in L(\tilde{S}_i/\tilde{G}_i) \) and \( p_i^{dec}(s) \) \( t \in L_m(S_i/G_i) \) for \( t \in \Sigma_i \). There are two cases.

1. \( t = \epsilon \). As \( p_i^{dec} \) is a \( L_m(G_i) \)-observer, there is a \( u_i \in (\tilde{\Sigma}_i - \Sigma_i)^* \) s.t. \( su_i \in L_m(G_i) \). Because of Lemma 5.3, \( su_i \in L(\tilde{S}_i/\tilde{G}_i) \). Together, \( su_i \in L(\tilde{S}_i/\tilde{G}_i) \cap L_m(G_i) = L_m(S_i/G_i) \).

2. \( t = \sigma_1 \cdots \sigma_m \). As \( p_i^{dec} \) is a \( L_m(G_i) \)-observer, there is a \( u_i = \nu_0 \sigma_1 \nu_1 \cdots \sigma_m \nu_m \in \tilde{\Sigma}_i \) s.t. \( su_i \in L_m(G_i) \) and \( p_i^{dec}(u_i) = t \), i.e. \( v_j \in (\tilde{\Sigma}_i - \Sigma_i)^* \) for all \( j = 0, \ldots, m \). Successive application of Lemma 5.3 implies \( su_i \in L(\tilde{S}_i/\tilde{G}_i) \). Thus, \( su_i \in L(\tilde{S}_i/\tilde{G}_i) \cap L_m(G_i) = L_m(S_i/G_i) \).

Proof of Theorem 5.4:

Consistency follows from the proof of Theorem 5.3.

Now assume that \( s \in L(\tilde{S}_i/\tilde{G}_i) \). Then \( s_i := \tilde{p}_i(s) \in L(\tilde{S}_i/\tilde{G}_i) \) and \( p_i^{dec}(s_i) \in L(S_i/G_i) \). As \( S_i \) is a nonblocking supervisor, there is a string \( t \tilde{\Sigma}_i \) s.t. \( p_i^{dec}(s_i) t \in L_m(S_i/G_i) \) and s.t. all its predecessors are not marked, i.e. \( \forall t' < t \) we have that \( p_i^{dec}(s_i) t' \not\in L_m(S_i/G_i) \). Then it holds that \( t \in (\tilde{\Sigma}_i - \Sigma_i)^* \) (otherwise there would be a marked predecessor string as \( L_m(G_i) \) marks \( \Sigma_i \)). Because of Lemma 5.4, there is a \( u_i \in \tilde{\Sigma}_i \) s.t. \( su_i \in L_m(\tilde{S}_i/\tilde{G}_i) \) and \( p_i^{dec}(su_i) = p_i^{dec}(s_i) t \). Furthermore, as \( p_i^{dec}(u_i) = t \subseteq (\tilde{\Sigma}_i - \Sigma_i)^* \), it turns out that \( u_i \in (p_i^{dec})^{-1}(t) \subseteq (\tilde{\Sigma}_i - \Sigma_i)^* \). As \( i \) was arbitrary, such \( u_i \) exists for all \( i = 1, \ldots, n \). Hence, for example the string \( su_i \cdots u_n \in ||\tilde{p}_i(s)||u_i \subseteq ||L_m(\tilde{S}_i/\tilde{G}_i) \cap L_m(S_i/G_i) || \) and consequently \( s \in L_m(S_i/G_i) \).

The reduced structural decentralized control architecture is depicted in Figure 5.

VI. CONCLUSIONS

We have developed two methods exploiting the structure of concurrent systems for the supervisor synthesis without composition of the overall plant. In our approach, the computational complexity is further reduced by using reduced system models for supervisor computation. Our modular approach can be applied to prefix-closed non-separable specifications and results in modular supervisors in a conjunctive architecture. Additionally, we elaborated a decentralized approach which is feasible for specifications that are separable but not necessarily prefix-closed. We provide two different sets of conditions which guarantee nonblocking control of the original system. It has to be noted that although maximally permissive supervisors could be computed for the reduced system models, the supervisors for the original system need not be maximally permissive. In further work, we want to investigate conditions which also guarantee maximally permissive supervisors for the original system.

REFERENCES