Supervisory Control of Modular Discrete-Event Systems under Partial Observation: Normality

Jan Komenda and Tomáš Masopust

***Abstract*—Complex systems are often composed of many small communicating components called modules. We investigate the synthesis of supervisory controllers for modular systems under partial observation that, as the closed-loop system, realize the supremal normal sublanguage of the specification. Such controllers are called maximally permissive normal supervisors. The challenge in modular systems is to find conditions under which the global nonblocking and maximally permissive normal supervisor can be achieved locally as the parallel composition of local normal supervisors. We show that a structural concept of hierarchical supervisory control called modified observation consistency (MOC) is such a condition. However, the algorithmic verification of MOC is an open problem, and therefore it is necessary to find easily-verifiable conditions that ensure MOC. We show that the condition that all shared events are observable is such a condition. Considering specifications, we examine both local specifications, where each module has its own specification, and global specifications. Combining our results for normality with the existing results for controllability yields the local synthesis of the nonblocking and maximally permissive controllable and normal supervisor. Finally, we illustrate the results on an industrial case study of the patient table of an MRI scanner.**

***Index Terms*—Discrete-event system, Modular control, Observation consistency, Normality.**

I. Introduction

Organizing large-scale complex systems into interconnected communicating components is a common engineering practice, which has many applications across control theory and computer science, including manufacturing, robotics, and artificial intelligence [1]–[3].

The key concepts of supervisory control under partial observation are controllability and observability [4], [5]. Unlike controllability, however, observability is not closed under union, and hence the set of observable sublanguages of a given language does not have the supremal element. Therefore, other concepts, including normality or relative observability, are used instead of observability [6], [7]. For more results on supervisory control of partially-observed (monolithic) plants, we refer the reader to, e. g., Takai and Ushio [8], Thistle and Lamouchi [9], or Yin and Lafortune [10].

Compared with the monolithic plant, modular discrete-event systems (modular DES) consist of a concurrent composition of many small components, also known as modules. The main challenge of supervisory control for modular DES is the problem how to synthesize local supervisors in such a way that the concurrent behavior of local modules controlled by the corresponding local supervisors coincides with the behavior of the global (monolithic) plant controlled by the nonblocking and maximally permissive supervisor. The idea of a local controller synthesis is natural and has been discussed in the literature since the early days of supervisory control [11].

In this paper, we use the concept of normality. Normality is stronger than observability, and if all controllable events are observable, then the two notions coincide [4], [5].

Given a modular DES and a specification, the problem now is how to compute the supremal controllable and normal sublanguage of the specification without explicitly constructing the global plant. Since local computations of the supremal controllable sublanguage have widely been investigated in the literature, we focus on the local construction of the supremal normal sublanguage; and we show how to combine the results.

Although the supervisor synthesis for a monolithic plant with complete observation is polynomial in time, the main motivation for the computation of local supervisors is the avoidance of the construction of the monolithic plant, the size of which grows exponentially in the number of local modules. Furthermore, unlike complete observation, there are no polynomial-time algorithms for the synthesis of supervisors for systems under partial observation, and hence the local computation of supervisors for systems under partial observation is even more important.

Systems with Complete Observation

We now briefly review the main approaches to supervisory control of modular DES under complete observation. In this case, the problem is to locally compute the supremal controllable sublanguage of a global specification without explicitly constructing the global plant.

De Queiroz and Cury [12] considered modular DES, where local modules have no events in common, equipped with a set of specifications. The specifications do not correspond to the modules, and therefore, for each specification, a local plant is constructed as a parallel composition of those modules that share an event with the specification. De Queiroz and Cury showed that the parallel behavior of the maximally permissive local supervisors, computed for each specification and the corresponding local plant, coincides with the behavior of the nonblocking and maximally permissive monolithic supervisor.

Considering the same framework, a similar approach was discussed by Hill and Tilbury [13], who further employed an abstraction and a structuring into several levels.

Gaudin and Marchand [14], on the other hand, considered a global prefix-closed specification, which they localize with the

J. Komenda and T. Masopust are, respectively, with the Institute of Mathematics of the Czech Academy of Sciences, Prague, Czechia, and with the Faculty of Science, Palacky University Olomouc, Czechia. Emails: komenda@ipm.cz, tomas.masopust@upol.cz.

help of inverse projections to the alphabets of local modules. Then, they employ the concept of partial controllability to compute maximally permissive local supervisors. The parallel composition of the constructed local supervisors is then maximally permissive in the monolithic sense under the condition that all shared events are controllable.

Willner and Heymann [15] investigated global decomposable prefix-closed specifications—specifications that can be decomposed according to the alphabets of local modules. They showed that if all shared events are controllable, then the parallel composition of maximally permissive local supervisors is maximally permissive in the monolithic sense.

However, the assumption of decomposability of the specification [15] or of the solution [16], [17] is very restrictive. Therefore, we introduced the notion of *conditional decomposability* [18]; intuitively, rather than to decompose the specification with respect to the alphabets of local modules, we search for a coordinator that covers the shared communication among the modules in such a way that the specification can be decomposed with respect to the alphabets of local modules composed with the coordinator. In this way, we reduce a modular DES with a global specification to a modular DES with local specifications [19]; see also Section V.

Flordal et al. [20] developed an incremental compositional technique that avoids the construction of the monolithic supervisor and preserves the nonblockingness and maximal permissiveness of the closed-loop system. We further refer to Abdelwahed and Wonham [21] and Lee and Wong [22].

Another approach to supervisory control of modular systems is based on the *observer property* [23], on the property of *output control consistency* (OCC) [24], and on the property of *local control consistency* (LCC) [25]; these properties are concepts of hierarchical supervisory control. Namely, given a modular DES with a decomposable prefix-closed specification, if each projection from the overall alphabet to the alphabet of a local module satisfies the observer property and the LCC (or OCC) condition, then the parallel composition of maximally permissive local supervisors coincides with the monolithic nonblocking and maximally permissive supervisor, see Section VI for more details. Feng [26] has lifted this result to non-prefix-closed specifications by the assumption that the local supervisors are nonconflicting.

Combining this result with the result of Willner and Heymann [15] then gives that, for a modular DES under complete observation with a decomposable specification, if all shared events are controllable, then the concurrent behavior of maximally permissive local closed-loop systems coincides with the behavior of the nonblocking and maximally permissive monolithic closed-loop system.

Systems with Partial Observation

We investigate the synthesis of supervisory controllers for modular DES under partial observation. In particular, we focus on the synthesis of maximally permissive *normal supervisors*, which are supervisors realizing the supremal normal sublanguage of the specification.

Whereas the synthesis of maximally permissive local supervisors is well understood for modular DES under complete observation, the situation is significantly different for partial observation, and only a few partial results can be found in the literature.

Komenda and Van Schuppen [27] showed that the computation of local normal supervisors is maximally permissive for prefix-closed specifications if the modules share no events. They further suggested a condition of mutual normality, which is, however, too restrictive [28]. Komenda et al. [18] extended the coordination control framework to partial observation, where additional conditions are required to achieve maximal permissiveness of the composition of local supervisors.

Among other works related to supervisory control of modular DES, Rohloff and Lafortune [17], Su and Lennartson [29], or Liu et al. [30] modelled multi-agent systems as a modular DES composed of isomorphic modules. The isomorphism of modules allows us to handle the number of agents that can dynamically decrease or increase without an upper bound.

In this paper, we show that a concept of hierarchical supervisory control called *modified observation consistency* (MOC) is of interest for modular DES; namely, if a modular system satisfies the MOC condition, then the global nonblocking and maximally permissive normal supervisor can be achieved locally as the parallel composition of nonblocking and maximally permissive local normal supervisors (Theorem 3).

However, the algorithmic verification of MOC is a challenging open problem; the problem is known to be PSPACE-hard, but unknown to be decidable [31]. Therefore, we need other, easily-verifiable conditions that ensure MOC. We show that the assumption that *all shared events are observable* is such a condition (Theorem 8).

The reader may notice that several of the above-discussed approaches to supervisory control of modular DES under complete observation make a similar assumption; namely, that *all shared events are controllable*. Consequently, our condition that all shared events are observable naturally extends and complements the existing results.

Concerning the structure of a specification, we discuss two fundamental cases: (i) the specification is given as a set of local specifications, where each local module has its own local specification, and (ii) the specification is global, given as a sublanguage of the global plant language. For the latter case, we employ our coordination control framework [19] and show that under the assumption that *all coordinated events are observable*, global and local computations of nonblocking and maximally permissive normal supervisors coincide (Theorem 11).

Last but not least, our results provide further evidence that the modular supervisory control framework benefits from the results and concepts of hierarchical supervisory control. In particular, the concepts of the observer property, OCC, LCC, and MOC conditions, developed for supervisory control of hierarchical systems, find applications in supervisory control of modular DES.

II. Preliminaries and Definitions

We assume that the reader is familiar with the basic concepts of supervisory control [4]. For a set A, we denote by $|A|$ the

cardinality of A. For an alphabet (finite nonempty set) Σ, we denote by Σ^* the set of all finite strings over Σ; the empty string is denoted by ε. A language L over Σ is a subset of Σ^*. The prefix closure of a language L is the set $\overline{L} = \{w \in \Sigma^* \mid \text{there exists } v \in \Sigma^* \text{ such that } wv \in L\}$. A language L is prefix-closed if $L = \overline{L}$.

A *projection* $R\colon \Sigma^* \to \Gamma^*$, where $\Gamma \subseteq \Sigma$ are alphabets, is a morphism for concatenation defined by $R(a) = \varepsilon$ if $a \in \Sigma - \Gamma$, and $R(a) = a$ if $a \in \Gamma$. The action of R on a string $a_1 a_2 \cdots a_n$ is to remove events that are not in Γ, that is, $R(a_1 a_2 \cdots a_n) = R(a_1)R(a_2)\cdots R(a_n)$. The inverse image of a string $w \in \Gamma^*$ under the projection R is the set $R^{-1}(w) = \{s \in \Sigma^* \mid R(s) = w\}$. The definitions can readily be extended to languages.

A *deterministic finite automaton* (DFA) is a quintuple $G = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is an alphabet, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of marked states, and $\delta\colon Q \times \Sigma \to Q$ is the transition function that can be extended to the domain $Q \times \Sigma^*$ in the usual way. The language $L(G) = \{w \in \Sigma^* \mid \delta(q_0, w) \in Q\}$ is *generated* by G, while the language $L_m(G) = \{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$ is *marked* or *accepted* by G. By definition, $L_m(G) \subseteq L(G)$, and $L(G)$ is prefix-closed. If $\overline{L_m(G)} = L(G)$, then G is called *nonblocking*.

A discrete-event system (DES) over Σ is a DFA over Σ together with the determination of *controllable events* $\Sigma_c \subseteq \Sigma$ and *uncontrollable events* $\Sigma_{uc} = \Sigma - \Sigma_c$, and of *observable events* $\Sigma_o \subseteq \Sigma$ and *unobservable events* $\Sigma_{uo} = \Sigma - \Sigma_o$.

For a DES G over Σ, we denote the set of control patterns by $\Gamma = \{\gamma \subseteq \Sigma \mid \Sigma_{uc} \subseteq \gamma\}$, and the projection removing unobservable events by $P\colon \Sigma^* \to \Sigma_o^*$. A *supervisor* of G with respect to a set of control patterns Γ is a map $S\colon P(L(G)) \to \Gamma$ returning, for every observed string, a set of enabled events that always includes all uncontrollable events. The *closed-loop system* of G under the supervision of S is the minimal language $L(S/G)$ such that $\varepsilon \in L(S/G)$ and for every $s \in L(S/G)$, if $sa \in L(G)$ and $a \in S(P(s))$, then $sa \in L(S/G)$. If the closed-loop system is nonblocking, that is, $\overline{L_m(S/G)} = L(S/G)$, the supervisor S is called *nonblocking*. Intuitively, the supervisor disables some of the controllable transitions based on the partial observation of the system.

There are two views on the marked language of the closed-loop system: (i) the marking is adopted from the plant G, that is, $L_m(S/G) = L(S/G) \cap L_m(G)$, and (ii) the supervisor marks according to a given specification $M \subseteq L(G)$, that is, $L_m(S/G) = L(S/G) \cap M$. In the latter case, the existence of a supervisor achieving the specification is equivalent to controllability and observability of the specification, whereas, in the former case, an additional assumption of $L_m(G)$-closedness is needed [5, Section 6.3].

However, unlike controllability, observability is not closed under union, and hence the set of observable sublanguages of a given language does not have the supremal element. For this reason, other concepts are used instead of observability. In this paper, we use the concept of normality. Normality is stronger than observability, and coincides with observability if all controllable events are observable [4], [5].

For a DES G over Σ, a language $K \subseteq L_m(G)$ is *controllable* with respect to the language $L(G)$ and the set Σ_{uc} of uncontrollable events if $\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}$. The language K is *normal* with respect to the language $L(G)$ and the projection $P\colon \Sigma^* \to \Sigma_o^*$ if $\overline{K} = P^{-1}[P(\overline{K})] \cap L(G)$.

For a prefix-closed language L and a (not necessarily prefix-closed) specification $K \subseteq L$, we denote by

$$\mathrm{supN}(K, L, P)$$

the supremal normal sublanguage of the specification K with respect to the language L and projection P [32].

The parallel composition of languages $L_i \subseteq \Sigma_i^*$ is the language $\|_{i=1}^n L_i = \cap_{i=1}^n P_i^{-1}(L_i)$, where $P_i\colon (\cup_{i=1}^n \Sigma_i)^* \to \Sigma_i^*$ is the projection, $i = 1, \ldots, n$. A definition of the parallel composition for automata can be found in the literature [4]. In particular, for DFAs G_i, we have $L(\|_{i=1}^n G_i) = \|_{i=1}^n L(G_i)$ and $L_m(\|_{i=1}^n G_i) = \|_{i=1}^n L_m(G_i)$. The languages L_i are *(synchronously) nonconflicting* if $\overline{\|_{i=1}^n L_i} = \|_{i=1}^n \overline{L_i}$.

A modular DES $G = \|_{i=1}^n G_i$ consists of the parallel composition of $n \geq 2$ systems or modules G_i over local alphabets Σ_i, for $i = 1, \ldots, n$.

A. Modular Supervisory Control

We now briefly review the principles and concepts of supervisory control of modular DES.

The modular supervisory control problem consists of a modular system modeled by a set of $n \geq 2$ automata

$$G_1, \ldots, G_n$$

generating languages $L_1 = L(G_1), \ldots, L_n = L(G_n)$, respectively, with the global (monolithic) behavior $L = \|_{i=1}^n L_i$, and of a specification K given either

1) as a parallel composition $K = \|_{i=1}^n K_i$ of a set of local specifications $K_i \subseteq L_i$, or
2) as a global specification $K \subseteq \|_{i=1}^n L_i$.[1]

The aim is to synthesize local controllers S_i such that

$$\|_{i=1}^n L_m(S_i/G_i) = L_m(S/\|_{i=1}^n G_i) \tag{1}$$

where S denotes the nonblocking and maximally permissive supervisor for the global specification K and the global plant language L. In particular, the fundamental question is under which conditions (1) holds.

This problem is well understood for modular systems under complete observation, where we essentially have two types of sufficient conditions ensuring the maximal permissiveness of the local (modular) control synthesis:

1) conditions adopted from hierarchical supervisory control; namely, the observer property and OCC/LCC conditions [25], [33], and
2) the condition of mutual controllability and the variants thereof [27].

For partially observed DES, however, only the second type of conditions was discussed in the literature [34].

In this paper, we discuss a condition of the first type called modified observation consistency.

[1]The specification $K \subseteq \|_{i=1}^n L_i$ is called *decomposable* with respect to $\|_{i=1}^n L_i$ if there are $K_i \subseteq L_i$, for $i = 1, , \ldots, n$, such that $K = \|_{i=1}^n K_i$.

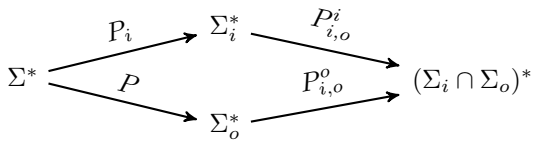

Figure 1. Commutative diagram of abstractions and projections.

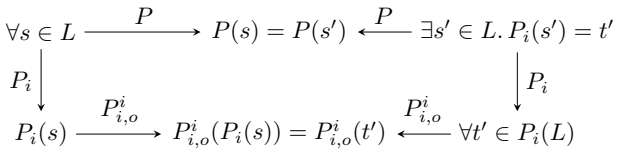

Figure 2. Illustration of the MOC condition.

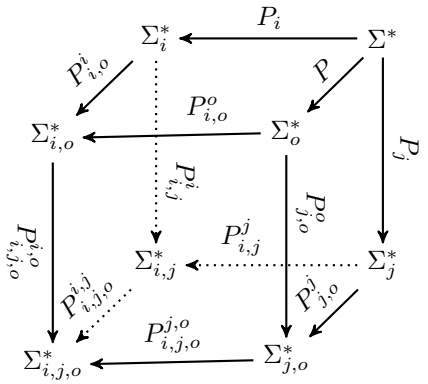

Figure 3. Our notation for the projections used in modular systems.

B. Modified Observation Consistency

Modified observation consistency (MOC) is a concept of hierarchical supervisory control under partial observation developed to ensure hierarchical consistency [31].

Before we recall the definition of MOC, we fix the notation for projections. Namely, we denote system's partial observation by the projection $P\colon \Sigma^* \to \Sigma_o^*$, the local projection to a module by the projection $P_i\colon \Sigma^* \to \Sigma_i^*$, and the corresponding restricted observations and projections by $P_{i,o}^i\colon \Sigma_i^* \to (\Sigma_i \cap \Sigma_o)^*$ and $P_{i,o}^o\colon \Sigma_o^* \to (\Sigma_i \cap \Sigma_o)^*$, see Figure 1.

Definition 1. *A prefix-closed language $L \subseteq \Sigma^*$ is* modified observation consistent *(MOC) with respect to projections P_i, P, and $P_{i,o}^i$ if for every string $s \in L$ and every string $t' \in P_i(L)$ such that $P_{i,o}^i(P_i(s)) = P_{i,o}^i(t')$, there exists a string $s' \in L$ such that $P(s) = P(s')$ and $P_i(s') = t'$.*

Intuitively, every string of a localized plant that locally looks the same as a global string s has a locally equivalent global string s' that looks the same as s, cf. Figure 2 for an illustration.

We define the set of local observable events as $\Sigma_i \cap \Sigma_o$. For simplicity, in the sequel, the intersection of two alphabets is denoted by the corresponding subscripts separated by commas; for instance, $\Sigma_{i,o} = \Sigma_i \cap \Sigma_o$. The same notation is used for the intersection of other alphabets, and for the intersection of more than two alphabets. The projections used in this paper are summarized in Figure 3.

Finally, we define the set of all shared events of the modular DES $G = \|_{i=1}^n G_i$, where the alphabet of G_i is Σ_i, as

$$\Sigma_s = \bigcup_{i \neq j} (\Sigma_i \cap \Sigma_j) \tag{2}$$

and we assume that

modular components agree on the observability status of shared events.

In other words, if an event is observable in one component, it is observable in all components where it appears; formally,

$$\Sigma_{i,o} \cap \Sigma_j = \Sigma_i \cap \Sigma_{j,o} = \Sigma_{i,o} \cap \Sigma_{j,o}\,.$$

III. Synthesis of Normal Supervisors for Modular DES with Local Specifications

In this section, we consider the case of local specifications, where each module L_i has its own specification K_i. The problem is to find conditions under which the parallel composition of the supremal normal sublanguages computed locally for each pair (K_i, L_i) of a specification and a local plant coincides with the supremal normal sublanguage computed for the global specification $K = \|_{i=1}^n K_i$ and the monolithic plant $L = \|_{i=1}^n L_i$. In other words, the question is under which conditions

$$\mathrm{supN}(K, L, P) = \|_{i=1}^n \mathrm{supN}(K_i, L_i, P_{i,o}^i)\,.$$

We show that the MOC condition plays a key role to answer this question. To prove the main result of this section, we make use of the following lemma.

Lemma 2. *For a modular DES $G = \|_{i=1}^n G_i$ with $L = L(G)$ and $L_m = L_m(G)$. If L is MOC with respect to projections P_i, P, and $P_{i.o}^i$, then normality of $S \subseteq L_m$ with respect to L and P implies normality of $P_i(S)$ with respect to projections $P_i(L)$ and $P_{i,o}^i$, for $i \in \{1, \ldots, n\}$.*

Proof: We show that for any language $S \subseteq L_m$ that is normal with respect to L and P, the language $P_i(S)$ is normal with respect to $P_i(L)$ and $P_{i,o}^i$, that is, we show that $P_i(\overline{S}) = (P_{i,o}^i)^{-1} P_{i,o}^i(P_i(\overline{S})) \cap P_i(L)$.

However, since $P_i(\overline{S}) \subseteq (P_{i,o}^i)^{-1} P_{i,o}^i(P_i(\overline{S})) \cap P_i(L)$, we need to show the opposite inclusion. To this end, we consider a string $t' \in (P_{i,o}^i)^{-1} P_{i,o}^i(P_i(\overline{S})) \cap P_i(L)$. Then, there is a string $s \in \overline{S}$ such that $t' \in (P_{i,o}^i)^{-1} P_{i,o}^i(P_i(s))$, and therefore $P_{i,o}^i(P_i(s)) = P_{i,o}^i(t')$. By the MOC property, there is a string $s' \in L$ such that $P_i(s') = t'$ and $P(s) = P(s')$, and hence

$$s' \in P^{-1}P(s) \cap L \subseteq P^{-1}P(\overline{S}) \cap L = \overline{S}$$

where the last equality is by normality of the language S. Altogether, we have shown that $t' = P_i(s') \in P_i(\overline{S})$, which was to be shown. ∎

To avoid the well-known conflicting issues and to focus on the role of MOC in maximal permissiveness, we consider prefix-closed languages or assume that the languages are non-conflicting; the case of conflicting languages can be handled

by the existing approaches, such as those based on abstractions or coordinators for nonblockingness [19], [35], [36].

We now formulate the main result of this section.

Theorem 3. *Consider languages $K_i \subseteq L_i$ over Σ_i, where L_i is prefix-closed, for $i = 1, \ldots, n$ and $n \geq 2$. We define the global languages $K = \|_{i=1}^{n} K_i$ and $L = \|_{i=1}^{n} L_i$.*

1) *If the languages $\mathrm{supN}(K_i, L_i, P_{i,o}^i)$ are nonconflicting, then $\|_{i=1}^{n} \mathrm{supN}(K_i, L_i, P_{i,o}^i) \subseteq \mathrm{supN}(K, L, P)$.*
2) *If, in addition, for all $i = 1, \ldots, n$, $P_i(L) = L_i$ and the language L is MOC with respect to projections P_i, P, and $P_{i,o}^i$, then $\mathrm{supN}(K, L, P) = \|_{i=1}^{n} \mathrm{supN}(K_i, L_i, P_{i,o}^i)$.*

Proof: To simplify the notation, we denote the language $\mathrm{supN}(K_i, L_i, P_{i,o}^i)$ by supN_i.

To prove 1), we show that $\|_{i=1}^{n} \mathrm{supN}_i$ is normal with respect to L and P, which implies that $\|_{i=1}^{n} \mathrm{supN}_i$ is included in the supremal element $\mathrm{supN}(K, L, P)$. To this end, we use the nonconflictingness of supN_i, and we obtain that

$$
\begin{aligned}
\overline{\|_{i=1}^{n} \mathrm{supN}_i} &\subseteq P^{-1}P(\overline{\|_{i=1}^{n} \mathrm{supN}_i}) \cap L \\
&\qquad \text{(by the properties of projections)} \\
&= P^{-1}P(\|_{i=1}^{n} \overline{\mathrm{supN}_i}) \cap L \\
&\qquad \text{(by the nonconflictingness of } \mathrm{supN}_i\text{)} \\
&\subseteq P^{-1}(\|_{i=1}^{n} P_{i,o}^i(\overline{\mathrm{supN}_i})) \cap L \\
&\qquad \text{(by the projection of a parallel composition)} \\
&= \|_{i=1}^{n} (P_{i,o}^i)^{-1} P_{i,o}^i(\overline{\mathrm{supN}_i}) \cap L \\
&\qquad \text{(by the properties of inverse projections)} \\
&= \|_{i=1}^{n} (P_{i,o}^i)^{-1} P_{i,o}^i(\overline{\mathrm{supN}_i}) \cap \|_{i=1}^{n} L_i \\
&\qquad \text{(by replacing } L \text{ with } \|_{i=1}^{n} L_i\text{)} \\
&= \|_{i=1}^{n} [(P_{i,o}^i)^{-1} P_{i,o}^i(\overline{\mathrm{supN}_i}) \cap L_i] \\
&\qquad \text{(by reformulating the composition)} \\
&= \|_{i=1}^{n} \overline{\mathrm{supN}_i} \qquad \text{(by the normality of } \mathrm{supN}_i\text{)} \\
&= \overline{\|_{i=1}^{n} \mathrm{supN}_i} \\
&\qquad \text{(by the nonconflictingness of } \mathrm{supN}_i\text{)}
\end{aligned}
$$

which shows that $\|_{i=1}^{n} \mathrm{supN}_i$ is normal, as claimed.

To prove 2), the assumptions that $P_i(L) = L_i$ and that L is MOC with respect to projections P_i, P, and $P_{i,o}^i$, together with Lemma 2, imply that the language $P_i(\mathrm{supN}(K, L, P))$ is normal with respect to $P_i(L) = L_i$ and $P_{i,o}^i$. As a result, we have that $P_i(\mathrm{supN}(K, L, P)) \subseteq \mathrm{supN}_i$, for all $i = 1, \ldots, n$, which proves the claim. ∎

IV. Ensuring MOC

Looking at Theorem 3, the reader may see two key, restrictive assumptions: (i) $P_i(L) = L_i$ and (ii) L is MOC with respect to projections P_i, P, and $P_{i,o}^i$, for all $i = 1, \ldots, n$.

Whereas the condition $P_i(L) = L_i$ can be algorithmically verified, though in exponential time, it is in fact a modelling decision whether the condition $P_i(L) = L_i$ is satisfied. Indeed,

$$L = \|_{i=1}^{n} L_i = \|_{i=1}^{n} P_i(L)\,,$$

and hence $P_i(L)$ can be considered instead of L_i; in addition, the technique described in Section V handling the case of global specifications satisfies this condition by construction.

The verification of MOC, on the other hand, is a task that we are currently unable to perform algorithmically. More specifically, the verification of MOC is a PSPACE-hard problem and it is open whether the problem is decidable. Therefore, the main algorithmic issue is how to (easily) verify or ensure that a given plant language satisfies MOC. In other words, the existence of sufficient conditions under which the MOC condition is satisfied is a challenging open problem.

We now discuss this problem and show that the assumption that all shared events are observable is such a condition. To prove this result, the following well-known fact [5] is used, and the notation is simplified by denoting the languages $L(G_i)$ of local plants by L_i, for $i = 1, \ldots, n$.

Lemma 4. *Let $R\colon (\cup_{i=1}^{n} \Sigma_i)^* \to \Gamma^*$ be a projection, and let $R_i\colon \Sigma_i^* \to (\Gamma \cap \Sigma_i)^*$ be its restriction to local modules. If all shared events of the languages L_i over Σ_i, for $i = 1, \ldots, n$, are in Γ, that is, $\Sigma_s \subseteq \Gamma$, then $R(\|_{i=1}^{n} L_i) = \|_{i=1}^{n} R_i(L_i)$.* ∎

Now, we formulate a key lemma showing that the easily-verifiable assumption that all shared events are observable is of interest in supervisory control of modular systems. It is worth noticing that this assumption is similar to the frequently used assumption that all shared events are controllable, used in supervisory control of modular systems under complete observation [14], [15].

Lemma 5. *Given a modular plant $L = \|_{i=1}^{n} L_i$ that is formed by prefix-closed languages L_i over Σ_i, for $i = 1, \ldots, n$. If all shared events are observable, then the language L is MOC with respect to projections P_i, P, and $P_{i,o}^i$, for all $i = 1, \ldots, n$.*

Proof: To show that L is MOC with respect to P_i, P, and $P_{i,o}^i$, for all $i = 1, \ldots, n$, let $s \in L$ and $t_i \in P_i(L) \subseteq L_i$ be two arbitrary strings such that $P_{i,o}^i P_i(s) = P_{i,o}^i(t_i)$. We need to show that there is a string $s' \in L$ such that $P(s') = P(s)$ and $P_i(s') = t_i$. To this end, we first notice that

$$P(s) \parallel t_i \parallel (\|_{j \neq i} P_j(s)) \neq \emptyset$$

if and only if

$$P\big(P(s) \parallel t_i \parallel (\|_{j \neq i} P_j(s))\big) \neq \emptyset\,.$$

However, because all shared events are observable, that is, $\Sigma_s \subseteq \Sigma_o$, we have that

$$
\begin{aligned}
&P(P(s) \parallel t_i \parallel (\|_{j \neq i} P_j(s))) \\
&= P(s) \parallel P_{i,o}^i(t_i) \parallel (\|_{j \neq i} P_{j,o}^j P_j(s)) \qquad \text{(by Lemma 4)} \\
&= P(s) \parallel P_{i,o}^i P_i(s) \parallel (\|_{j \neq i} P_{j,o}^j P_j(s)) \quad \text{(by the assumption)} \\
&= P(s) \parallel (\|_{i=1}^{n} P_{i,o}^i P_i(s)) \\
&= P(s) \parallel (\|_{i=1}^{n} P_{i,o}^o P(s)) \quad \text{(by commutativity of Figure 1)}
\end{aligned}
$$

Since $P(s) \in P(s) \parallel (\|_{i=1}^{n} P_{i,o}^o P(s))$, we have $P(P(s) \parallel t_i \parallel (\|_{j \neq i} P_j(s))) \neq \emptyset$, and hence $P(s) \parallel t_i \parallel (\|_{j \neq i} P_j(s)) \neq \emptyset$. Therefore, considering any $s' \in P(s) \parallel t_i \parallel (\|_{j \neq i} P_j(s)) \subseteq P(L) \parallel L_i \parallel (\|_{j \neq i} L_j) = P(L) \parallel L = L$, we have that the string $s' \in L$, $P(s') = P(s)$, and $P_i(s') = t_i$, as required. ∎

A corollary for prefix-closed specifications is stated below.

Corollary 6. *Given prefix-closed languages L_i over Σ_i, for $i = 1, \ldots, n$ with $n \geq 2$. If the alphabets $\Sigma_1, \ldots, \Sigma_n$ are*

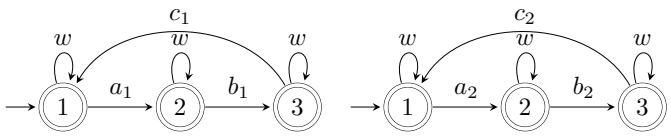

Figure 4. Generators G_1 and G_2 of Example 9.

pairwise disjoint, then, for all $i = 1, \ldots, n$, $P_i(L) = L_i$ and the language $L = \|_{j=1}^{n} L_j$ is MOC with respect to projections P_i, P, and $P_{i,o}^{i}$.

Proof: Since the alphabets are pairwise disjoint, the set of shared events $\Sigma_s = \emptyset \subseteq \Sigma_i$, for all i, and Lemma 4 implies that $P_i(L) = P_i(\|_{j=1}^{n} L_j) = \|_{j=1}^{n} P_i(L_j) = L_i$, as claimed.

The rest follows from Lemma 5. ∎

Remark 7. *It is worth pointing out that the result of Komenda and Van Schuppen [27, Theorem 5.13], which shows that*

$$\mathrm{supN}(K, L, P) = \|_{i=1}^{n} \mathrm{supN}(K_i, L_i, P_{i,o}^{i})$$

for prefix-closed local specifications over disjoint alphabets of local modules, is a consequence of Corollary 6 and Theorem 3.

We are now ready to present the main result of this section.

Theorem 8. *Let $n \geq 2$, and let $L = \|_{i=1}^{n} L_i$ be a modular DES formed by prefix-closed languages L_i over Σ_i. Let $K = \|_{i=1}^{n} K_i$ with $K_i \subseteq L_i$ be a specification. If the languages $\mathrm{supN}(K_i, L_i, P_{i,o}^{i})$ are nonconflicting, $P_i(L) = L_i$, and all shared events are observable, then*

$$\mathrm{supN}(K, L, P) = \|_{i=1}^{n} \mathrm{supN}(K_i, L_i, P_{i,o}^{i})\,.$$

Proof: Since $\Sigma_s \subseteq \Sigma_o$, Lemma 5 implies that L is MOC with respect to projections P_i, P, and $P_{i,o}^{i}$, for all $i = 1, \ldots, n$. The result then follows by the application of Lemma 5 and Theorem 3. ∎

To illustrate our results, we discuss the following example.

Example 9. *We consider a modular system consisting of two modules G_1 and G_2 depicted in Figure 4, where each module models a process consisting of three steps, a_i, b_i, c_i, $i = 1, 2$, and the event w serves as a possible synchronisation of the processes. The observable events are $\Sigma_o = \{w, a_1, a_2, b_1, b_2\}$, and all events are controllable. We set $L_i = L(G_i)$, for $i = 1, 2$. Obviously, all shared events are observable, and it can be verified that $P_i(L) = L_i$, for $i = 1, 2$. These observations are the assumptions of Theorem 8, and hence Theorem 8 is applicable to any local specifications K_i, $i = 1, 2$, for which the supremal normal languages $\mathrm{supN}(K_i, L_i, P_{i,o}^{i})$ are nonconflicting.*

For an illustration, we consider prefix-closed safety specifications $K_1 = \overline{\{a_1 w b_1 c_1\}}^*$ and $K_2 = \overline{\{a_2 b_2 w c_2\}}^*$ that require to synchronize the processes at the positions of the event w. The local supervisors realize the supremal normal languages $\mathrm{supN}(K_1, L_1, P_{1,o}^{1}) = K_1$ and $\mathrm{supN}(K_2, L_2, P_{2,o}^{2}) = \overline{\{a_2\}}$. The reader may verify that $\mathrm{supN}(K_1 \| K_2, L_1 \| L_2, P) = K_1 \parallel \overline{\{a_2\}} = \overline{\{a_1 a_2, a_2 a_1\}}$.

V. Synthesis of Normal Supervisors for Modular DES with Global Specifications

In this section, we discuss the case of global specifications. The main idea is to use the concept of conditional decomposability of our coordination control framework, where we use it to handle the interaction among various local modules [18].

Specifically, for a modular system with local languages L_i over Σ_i, and a global specification $K \subseteq L = \|_{i=1}^{n} L_i$, we find an alphabet

$$\Sigma_\kappa \supseteq \Sigma_s$$

containing all shared events in such a way that the specification K is *conditionally decomposable* with respect to the alphabets of local modules extended with the events of Σ_κ; formally,

$$K = \|_{i=1}^{n} P_{i+\kappa}(K)$$

where $P_{i+\kappa}\colon \Sigma^* \to \Sigma_{i+\kappa}^*$ is the projection from the global alphabet $\Sigma = \cup_{i=1}^{n} \Sigma_i$ to the local alphabet $\Sigma_{i+\kappa} = \Sigma_i \cup \Sigma_\kappa$ of the local module L_i extended with the events of Σ_κ, see Komenda et al. [19] for more details.

The idea of the coordination approach is to decompose the global specification K into local specifications $P_{i+\kappa}(K)$ over the local alphabets $\Sigma_{i+\kappa}$, for $i = 1, \ldots, n$; note that there is always a suitable alphabet Σ_κ, for which the specification K is conditionally decomposable—if there were no better choice, then the choice of $\Sigma_\kappa = \Sigma$ would work. The verification of conditional decomposability as well as the computation of a suitable alphabet Σ_κ is of polynomial-time complexity, although the computation of the minimal Σ_κ with respect to set inclusion is NP-hard [37].

Having determined an alphabet Σ_κ, we compute a coordinator G_κ as a DFA satisfying

$$L_\kappa = L(G_\kappa) = P_\kappa(L) = \|_{i=1}^{n} P_{\kappa,i}^{i}(L_i)$$

where $P_\kappa\colon \Sigma^* \to \Sigma_\kappa^*$ and $P_{\kappa,i}^{i}\colon \Sigma_i^* \to \Sigma_{i,\kappa}^*$ are projections.

This way, we have transformed the original modular system consisting of modules L_i over Σ_i and a global specification K to a modular system consisting of modules

$$L_{i+\kappa} = P_{i+\kappa}(L) = L_i \parallel L_\kappa \tag{3}$$

and the local specifications

$$K_{i+\kappa} = P_{i+\kappa}(K) \tag{4}$$

for which the plant language is $L = \|_{i=1}^{n} L_i = \|_{i=1}^{n} L_{i+\kappa}$ and the specification is $K = \|_{i=1}^{n} K_{i+\kappa}$, see Komenda et al. [19] for more details.

Now, we obtain the following corollary of Lemma 5; see Figure 5 for the notation of the used projections.

Lemma 10. *Consider a modular DES $G = \|_{i=1}^{n} G_i$, where Σ_s denotes the set of shared events. Let $L_i = L(G_i)$. Then, for every alphabet $\Sigma_\kappa \supseteq \Sigma_s$, whenever $\Sigma_\kappa \subseteq \Sigma_o$, the global plant $L = \|_{i=1}^{n} L_i$ is MOC with respect to projections $P_{j+\kappa}$, P, and $P_{j+\kappa,o}^{j+\kappa}$, for all $j = 1, \ldots, n$, where $P_{j+\kappa,o}^{j+\kappa}\colon \Sigma_{j+\kappa}^* \to \Sigma_{j+\kappa,o}^*$.*

Proof: The proof follows from Lemma 5 applied to the language $L = \|_{i=1}^{n} L_{i+\kappa} = \|_{i=1}^{n} L_i$, because the shared events of $L_{i+\kappa}$, for $i = 1, \ldots, n$, are observable—the set of shared

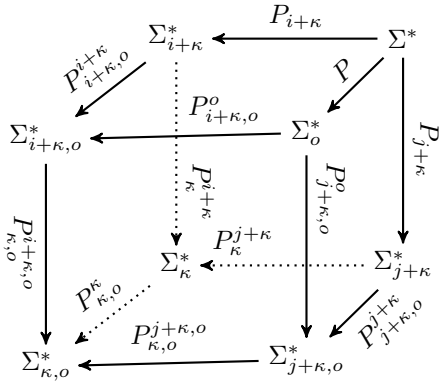

Figure 5. Projections used in the coordination control framework.

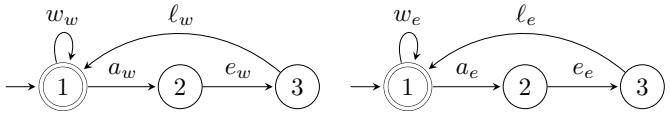

Figure 6. Generators G_1 and G_2 modelling the trains T_1 and T_2, resp.

events is the alphabet $\Sigma_\kappa \subseteq \Sigma_o$. See also Figure 5 for the notation of projections. ■

Now, we can state the main result of this section.

Theorem 11. *Consider prefix-closed languages L_i over Σ_i, for $i = 1, \ldots, n$, and define the global language $L = \|_{i=1}^n L_i$. Let $K \subseteq L$ be a global specification. Compute an alphabet Σ_κ containing all shared events such that Σ_κ makes the specification K conditionally decomposable with respect to alphabets $\Sigma_{i+\kappa}$, $i = 1, \ldots, n$. If, for the languages $L_{i+\kappa}$ and $K_{i+\kappa}$ defined in* (3) *and* (4)*, respectively,*

1) *languages* $\mathrm{supN}(K_{i+\kappa}, L_{i+\kappa}, P^{i+\kappa}_{i+\kappa,o})$ *are nonconflicting, and*
2) $\Sigma_\kappa \subseteq \Sigma_o$ *consists only of observable events,*

then $\mathrm{supN}(K, L, P) = \|_{i=1}^n \mathrm{supN}(K_{i+\kappa}, L_{i+\kappa}, P^{i+\kappa}_{i+\kappa,o})$.

Proof: The result follows from Lemma 10 and Theorem 8, because $L_{i+\kappa} = P_{i+\kappa}(L)$, for $i = 1, \ldots, n$, by definition, and the alphabet Σ_κ forms the set of shared events of local modules $L_{i+\kappa}$, for $i = 1, \ldots, n$. ■

To illustrate the results, we use a simple railroad example.

Example 12. *We consider the synthesis of a bridge controller of a railroad with two tracks and a bridge where the tracks merge motivated by Alur [38]. Two trains operate in the system—the western train T_1 and the eastern train T_2. To control the access to the bridge, trains communicate with the bridge controller. If the western train arrives at the bridge, it sends the arrive signal a_w. If the bridge controller accepts the signal, the train can enter the bridge (e_w); otherwise, it waits (w_w) and keeps sending the arrive signal a_w until it is accepted. When leaving the bridge, the train sends the leave signal ℓ_w. The eastern train behaves the same, using the signals a_e, e_e, w_e, and ℓ_e, respectively. The models G_1 and G_2 of the two trains are depicted in Figure 6.*

We post the safety requirement that a train may enter the bridge if its arrive signal is accepted. The signal may be accepted if the other train waits at or is away from the bridge and there is no train on the bridge. Our specification depicted in Figure 7 takes care of this requirement as well as of a kind of fairness; namely, both trains wait before the arrive signal of one of them is accepted, and no train that wants to enter the bridge waits for ever.

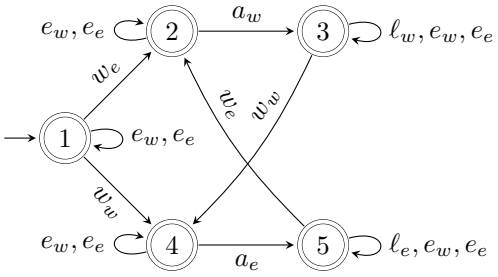

Figure 7. The global specification K.

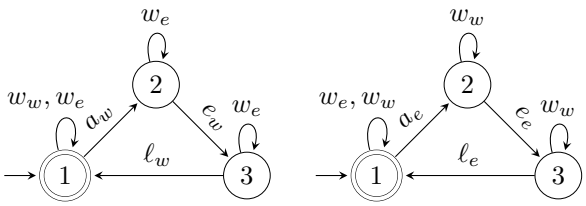

Figure 8. New modules $L_{1+\kappa}$ and $L_{2+\kappa}$.

Our focus is primarily on partial observation, and therefore we assume that all events are controllable. The observable events are $\Sigma_o = \{w_w, w_e, a_w, a_w, e_e, e_w\}$, that is, unobservable events are $\Sigma_{uo} = \{\ell_w, \ell_e\}$. To simplify the notation, we define $L_i = L(G_i)$, for $i = 1, 2$, and $L = L_1 \parallel L_2$.

Since the specification is global, we look for an alphabet Σ_κ that contains all shared events and that makes our specification, which we denote by K, conditionally decomposable. The alphabets of local modules are $\Sigma_1 = \{a_w, e_w, \ell_w, w_w\}$ and $\Sigma_2 = \{a_e, e_e, \ell_e, w_e\}$, and hence the set of shared events is $\Sigma_s = \emptyset$. The reader may verify that for the alphabet

$$\Sigma_\kappa = \{w_w, w_e\}$$

we have $K = P_{1+\kappa}(K) \parallel P_{2+\kappa}(K)$; in other words, the alphabet Σ_κ makes the specification K of Figure 7 conditionally decomposable. Now, we compute the language

$$L_\kappa = \|_{i=1}^n P^i_{\kappa,i}(L_i) = \{r_1, r_2\}^*$$

and the new modules $L_{i+\kappa} = L_i \parallel L_\kappa$, see Figure 8.

Having decomposed the global specification K into local specifications $K_{i+\kappa} = P_{i+\kappa}(K)$, as depicted in Figure 9, we proceed by computing the local normal supervisors realizing the supremal normal languages $\mathrm{supN}(K_{i+\kappa}, L_{i+\kappa}, P^{i+\kappa}_{i+\kappa,o})$, for $i = 1, 2$, depicted in Figure 10.

Since the alphabet $\Sigma_\kappa \subseteq \Sigma_o$ consists only of observable events, and the languages $\mathrm{supN}(K_{i+\kappa}, L_{i+\kappa}, P^{i+\kappa}_{i+\kappa,o})$ are prefix-closed, and hence nonconflicting, Theorem 11 gives that

$$\mathrm{supN}(K, L, P) = \|_{i=1}^n \mathrm{supN}(K_{i+\kappa}, L_{i+\kappa}, P^{i+\kappa}_{i+\kappa,o})\,.$$

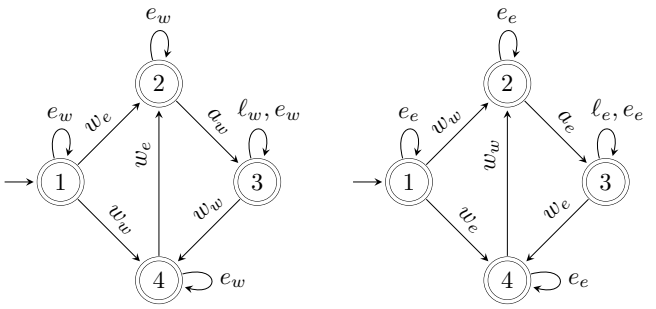

Figure 9. Decomposition of the global specification K into local specifications $K_{i+\kappa} = P_{i+\kappa}(K)$ of the new modules $L_{i+\kappa}$.

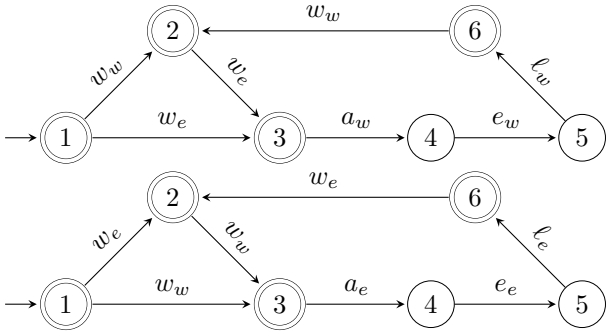

Figure 10. The constructed local normal supervisors.

VI. Modular Computation of Supremal Controllable and Normal Sublanguages

Under complete observation, Zhong and Wonham [24] formulated the notion of *output control consistency* (OCC) that together with the *observer property* guarantee that controllability is preserved under projections.

For two alphabets $\Gamma \subseteq \Sigma$, the projection $R\colon \Sigma^* \to \Gamma^*$ is *output control consistent* (OCC) for a prefix-closed language L over Σ if for every string $s \in L$ of the form $s = \sigma_1 \cdots \sigma_k$ or $s = s'\sigma_1 \cdots \sigma_k$, $k \geq 1$, satisfying that (i) the prefix s' terminates with an event from Γ, (ii) the event $\sigma_i \in \Sigma - \Gamma$, for $i = 1, \ldots, k-1$, and (iii) the last event $\sigma_k \in \Gamma$, we have that if the event σ_k is uncontrollable, then so are uncontrollable events σ_i, for $i = 1, \ldots, k-1$.

The projection $R\colon \Sigma^* \to \Gamma^*$ is an *$L_m(G)$-observer* for a nonblocking plant G over Σ if for all strings $t \in R(L_m(G))$ and $s \in \overline{L_m(G)}$, whenever the projection $R(s)$ is a prefix of t, then there is $u \in \Sigma^*$ such that $su \in L_m(G)$ and $R(su) = t$.

Now, we recall the result of Feng [26] that OCC and the observer property guarantee that controllability is preserved by projections. This result is a complete-observation counterpart of Lemma 2.

Lemma 13 ([26, Lemma 4.3])**.** *Let L and X be prefix-closed languages over an alphabet Σ, such that $X \subseteq L$. Suppose that a language $S \subseteq X$ is controllable with respect to L and Σ_{uc}. If the projection $R\colon \Sigma^* \to \Gamma^*$ is an X-observer and OCC for L, then $R(S)$ is controllable with respect to $R(X)$ and $\Sigma_{uc} \cap \Gamma$.* ■

Schmidt and Breindl [25] further investigated the problem of Lemma 13 and defined a weaker version of the OCC condition, called *local control consistency* (LCC). The projection $R\colon \Sigma^* \to \Gamma^*$ is *locally control consistent* (LCC) for a string $s \in L(G)$ if for every $e \in \Gamma \cap \Sigma_{uc}$ with $R(s)e \in R(L(G))$ either there is no $u \in (\Sigma - \Gamma)^*$ such that $sue \in L(G)$ or there is $u \in (\Sigma_{uc} - \Gamma)^*$ such that $sue \in L(G)$. We say that the projection R is LCC for a language if it is LCC for all strings of the language.

Lemma 14 ([25, Lemma 4.1])**.** *Let G over Σ be a nonblocking plant, let T over $\Gamma \subseteq \Sigma$ be a specification, and let $R\colon \Sigma^* \to \Gamma^*$ be the corresponding projection. If* supC *is the supremal controllable sublanguage of $T \parallel L_m(G)$ with respect to $L(G)$ and Σ_{uc}, and the projection R is an $L_m(G)$-observer and LCC for $L(G)$, then the language $R(\text{supC})$ is controllable with respect to $R(L(G))$ and $\Sigma_{uc} \cap \Gamma$.* ■

Because the proof of Lemma 14 does not depend on the supremality of the considered controllable language, we can reformulate it as follows.

Lemma 15. *Let L over Σ be a prefix-closed language, and suppose that a language $S \subseteq L$ is controllable with respect to L and Σ_{uc}. If the projection $R\colon \Sigma^* \to \Gamma^*$ is an L-observer and LCC for L, then $R(S)$ is controllable with respect to $R(L)$ and $\Sigma_{uc} \cap \Gamma$.* ■

Now, we combine our results for normality with the existing results for controllability. We use the notation

$$\text{supCN}(K, L, \Sigma_{uc}, P)$$

to denote the supremal controllable and normal sublanguage of the specification K with respect to the plant language L, the set of uncontrollable events Σ_{uc}, and the partial-observation projection $P\colon \Sigma^* \to \Sigma_o^*$.

Theorem 16. *For an integer $n \geq 2$, we consider a modular DES $L = \|_{i=1}^n L_i$ formed by prefix-closed languages L_i over Σ_i, and a specification $K = \|_{i=1}^n K_i$, where $K_i \subseteq L_i$. If, for all $i = 1, \ldots, n$,*

1) $\text{supCN}(K_i, L_i, \Sigma_{i,uc}, P_{i,o}^i)$ *are nonconflicting,*
2) $P_i(L) = L_i$,
3) *the projection $P_i\colon \Sigma^* \to \Sigma_i^*$ is an L-observer,*
4) *$P_i\colon \Sigma^* \to \Sigma_i^*$ is LCC (or OCC) for L, and*
5) *L is MOC with respect to projections P_i, P, and $P_{i,o}^i$,*

then

$$\text{supCN}(K, L, \Sigma_{uc}, P) = \|_{i=1}^n \text{supCN}(K_i, L_i, \Sigma_{i,uc}, P_{i,o}^i)\,.$$

Proof: The preservation of normality can be shown analogously as in the proof of Theorem 3.

For controllability, the right-to-left inclusion "$\supseteq$" follows from Proposition 4.6 of Feng [26] showing that the parallel composition of nonconflicting controllable languages is controllable. The left-to-right inclusion follows from Lemma 15 (resp. Lemma 13) by substituting P_i for R, for $i = 1, \ldots, n$, which shows that the language $P_i(\text{supCN}(K, L, \Sigma_{uc}, P)) \subseteq K_i$ is controllable with respect to $P_i(L) = L_i$ and $\Sigma_{i,uc}$, and hence it is included in $\text{supCN}(K_i, L_i, \Sigma_{i,uc}, P_{i,o}^i)$. ■

For completely observed modular DES with prefix-closed specifications, Willner and Heymann [15, Theorem 4.4] have shown that if all shared events are controllable, then the nonblocking and maximally permissive supervisor can be computed in a modular way. This result can be lifted to non-prefix-closed specifications with the help of Proposition 4.6 of Feng [26] as discussed in the proof of Theorem 16. We further lift it to partially observed modular systems as follows.

Corollary 17. *For an integer $n \geq 2$, we consider a modular DES $L = \|_{i=1}^{n} L_i$, formed by prefix-closed languages L_i over Σ_i, and a specification $K = \|_{i=1}^{n} K_i$, where $K_i \subseteq L_i$. If, for all $i = 1, \ldots, n$,*

1) $\mathrm{supCN}(K_i, L_i, \Sigma_{i,uc}, P_{i,o}^{i})$ *are nonconflicting,*
2) $P_i(L) = L_i$, *and*
3) *all shared events are controllable and observable,*

then

$$\mathrm{supCN}(K, L, \Sigma_{uc}, P) = \|_{i=1}^{n} \mathrm{supCN}(K_i, L_i, \Sigma_{i,uc}, P_{i,o}^{i})\,.$$

■

Combining Theorem 16 and Corollary 17 with the reduction of Section V, transforming a modular DES with a global specification to a modular DES with local specifications, we obtain the following results.

Theorem 18. *For an integer $n \geq 2$, we consider a modular DES $L = \|_{i=1}^{n} L_i$ formed by prefix-closed languages L_i over Σ_i, and a global specification $K \subseteq L$. Compute an alphabet Σ_κ containing all shared events such that Σ_κ makes the specification K conditionally decomposable with respect to alphabets $\Sigma_{i+\kappa}$, for $i = 1, \ldots, n$. If, for the languages $L_{i+\kappa}$ and $K_{i+\kappa}$ defined in* (3) *and* (4)*, respectively,*

1) *the languages* $\mathrm{supCN}(K_{i+\kappa}, L_{i+\kappa}, \Sigma_{i+\kappa,uc}, P_{i+\kappa,o}^{i+\kappa})$ *are nonconflicting,*
2) *the projection $P_{i+\kappa}\colon \Sigma^* \to \Sigma_{i+\kappa}^*$ is an L-observer,*
3) *$P_{i+\kappa}\colon \Sigma^* \to \Sigma_{i+\kappa}^*$ is LCC (or OCC) for L, and*
4) *the language L is MOC with respect to projections $P_{i+\kappa}$, P, and $P_{i+\kappa,o}^{i+\kappa}$,*

for all $i = 1, \ldots, n$, then

$$\mathrm{supCN}(K, L, \Sigma_{uc}, P) = \|_{i=1}^{n} \mathrm{supCN}(K_{i+\kappa}, L_{i+\kappa}, \Sigma_{i+\kappa,uc}, P_{i+\kappa,o}^{i+\kappa}).$$

Proof: A proof follows by the application of Theorem 16 to a modular DES with local specifications constructed from a modular DES with a global specification in Section V. ■

We can now formulate the following corollary.

Corollary 19. *For an integer $n \geq 2$, we consider a modular DES $L = \|_{i=1}^{n} L_i$, formed by prefix-closed languages L_i over Σ_i, and a global specification $K \subseteq L$. We compute an alphabet Σ_κ containing all shared events such that Σ_κ makes K conditionally decomposable with respect to $\Sigma_{i+\kappa}$, for $i = 1, \ldots, n$. If, for the languages $L_{i+\kappa}$ and $K_{i+\kappa}$ defined in* (3) *and* (4)*, respectively,* $\mathrm{supCN}(K_{i+\kappa}, L_{i+\kappa}, \Sigma_{i+\kappa,uc}, P_{i+\kappa,o}^{i+\kappa})$ *are nonconflicting, and either*

- *for $i = 1, \ldots, n$, $P_{i+\kappa}\colon \Sigma^* \to \Sigma_{i+\kappa}^*$ is an L-observer and LCC (or OCC) for L, and the alphabet Σ_κ contains only observable events,*

or

- *Σ_κ contains only controllable and observable events,*

then

$$\mathrm{supCN}(K, L, \Sigma_{uc}, P) = \|_{i=1}^{n} \mathrm{supCN}(K_{i+\kappa}, L_{i+\kappa}, \Sigma_{i+\kappa,uc}, P_{i+\kappa,o}^{i+\kappa})\,.$$

■

In a similar way, we could combine our results with the results of De Queiroz and Cury [12], Hill and Tilbury [13], or Gaudin and Marchand [14].

VII. Case Study

To evaluate our results on an industrial example, we consider the model and specification of a patient table of an MRI scanner designed by Theunissen [39]. The plant consists of four components

$$\mathrm{VAxis} \parallel \mathrm{HAxis} \parallel \mathrm{HVN} \parallel \mathrm{UI}\,.$$

Although each component is again a composition of other components, we do not go into more details and consider these four modules as the modular DES. Similarly, the specification consists of four parts

$$\mathrm{VReq} \parallel \mathrm{HReq} \parallel \mathrm{HVReq} \parallel \mathrm{UIReq}\,,$$

which do not exactly correspond to the four modules. In fact, each of the four parts of the specification concerns some of the four modules, and therefore each part of the specification can be seen as a global specification for the concerned modules.

The model of Theunissen [39] is constructed under complete observation. To lift it to partial observation, we define the events occurring in the specification as observable, and the other as unobservable. However, since we are currently unable to algorithmically verify MOC, and the models are too large for a manual verification, we need to ensure the MOC condition by making all shared (resp. coordinated) events observable, as required in Theorems 8 and 11.

For the computations, we used the C++ library `libFAUDES` in version 2.31d [40]. The computations were performed on an Intel-Core i7 processor laptop with 15 GB memory running Ubuntu 22.04.

We computed the automata representations of the supervisors using the `libFAUDES` function `SupConNormNB`, which implements the standard algorithm for the computation of the supremal controllable and normal sublanguage. The automata are further minimized with respect to the number of states using the function `StateMin`.

A. Procedure

We now describe the procedure how we handle the computation of local supervisors. Since we consider every part of the specification as a global specification, our procedure is based on the coordination approach described in Section V.

- For every $K \in \{\mathrm{VReq}, \mathrm{HReq}, \mathrm{HVReq}, \mathrm{UIReq}\}$, we take the set H of all plants from $\{\mathrm{VAxis}, \mathrm{HAxis}, \mathrm{HVN}, \mathrm{UI}\}$ that share an event with K.

- Let the alphabet of K be denoted by Σ_K. If Σ_K is strictly included in the set of events occurring in the plants of H, which we denote by Σ_H, we lift the specification K to the alphabet Σ_H by the inverse of projection $R\colon \Sigma_H^* \to \Sigma_K^*$.
  - Now, the specification is $R^{-1}(K)$.
- If needed, we make the specification $R^{-1}(K)$ conditionally decomposable with respect to the alphabets of the modules of H by computing an alphabet Σ_κ using the `libFAUDES` function `ConDecExtension` from the coordination control plug-in.
  - The set of observable events is $\Sigma_o = \Sigma_K \cup \Sigma_\kappa$.
  - Controllable events are defined by Theunissen [39].
- We consider the pair $(H, R^{-1}(K))$ as an instance of the modular DES consisting of the plans of H and of the global specification $R^{-1}(K)$.
  - We use Corollary 19 to construct local supervisors. The constructed supervisors are then maximally permissive in the monolithic sense.
  - Although we do not discuss the conditions of Corollary 19 for controllability below, we mention here that they are also satisfied.

B. Results

In this section, we briefly discuss the obtained results, which we summarize in the corresponding tables.

The specification VReq contains nine events that are shared only with the plant VAxis. Therefore, $H = \{\text{VAxis}\}$. The events of VReq are the only observable events. The automaton representation of VReq has 12 states and 44 transitions, while the representation of VAxis has 15 states and 50 transitions. A monolithic approach was used to compute a supervisor S with 15 states and 36 transitions. The results for the specifications VReq are summarized in Table I, where S denotes the automaton representing the controllable and normal supervisor realizing the supremal controllable and normal sublanguage of the specification.

Table I
THE SPECIFICATION VREQ.

	VReq	VAxis	S
States	12	15	15
Trans.	44	50	36
Events	9	11	11

Similarly, the specification HReq contains 19 events that are shared only with the plant HAxis. The events of HReq are the only observable events. The results for specification HReq are summarized in Table II.

The specification HVReq contains ten events that are shared with the plants VAxis, HAxis, and HVN. Therefore, the set $H = \{\text{VAxis}, \text{HAxis}, \text{HVN}\}$. Since the plant H is modular, we use the technique of Section V and the algorithms of Komenda and Masopust [37] to compute a set Σ_κ, with 14 events, that makes the specification, which is obtained by lifting HVReq to the set of events occurring in VAxis, HAxis, or HVN, conditionally decomposable. To apply Corollary 19, we set the events of Σ_κ and the events occurring in the specification HVReq observable, which results in the set of observable events Σ_o with 16 events. We construct the coordinator G_κ with 160 states, 1287 transitions, and 14 events, and three local nonblocking and maximally permissive controllable and normal supervisors S_1, S_2, S_3. The results are summarized in Table III.

Table II
THE SPECIFICATION HREQ.

	HReq	HAxis	S
States	112	128	80
Trans.	736	1002	320
Events	19	20	20

Table III
THE SPECIFICATION HVREQ.

	K	H					
	HVReq	VAxis	HAxis	HVN	S_1	S_2	S_3
States	7	15	128	1	516	1132	283
Trans.	35	50	1002	1	3395	10298	1692
Events	10	11	20	1	21	25	14

For comparison, the global plant VAxis $\|$ HAxis $\|$ HVN has 1920 states, 23350 transitions, and 32 events, and the nonblocking and maximally permissive controllable and normal supervisor of the specification obtained by lifting HVReq to the alphabet of VAxis $\|$ HAxis $\|$ HVN has 2064 states and 20120 transitions; see Section VII-D for a summary.

Finally, the specification UIReq contains 21 events that are shared with all four plants VAxis, HAxis, HVN, and UI. Again, the plant $H = \{\text{VAxis}, \text{HAxis}, \text{HVN}, \text{UI}\}$ is modular with the global specification UIReq, and hence we compute a set Σ_κ consisting of 10 events, all of which occur in UIReq. Consequently, the set of observable events is formed by the alphabet of UIReq. We now compute the coordinator G_κ with 4 states and 30 transitions, and four local nonblocking and maximally permissive controllable and normal supervisors S_1, S_2, S_3, S_4. The results are summarized in Table IV.

Table IV
THE SPECIFICATION UIREQ.

	K	H							
	UIReq	VAxis	HAxis	HVN	UI	S_1	S_2	S_3	S_4
States	256	15	128	1	2	432	768	12	96
Trans.	2336	50	1002	1	15	3488	6652	74	808
Events	21	11	20	1	9	21	24	10	16

For comparison, VAxis $\|$ HAxis $\|$ HVN $\|$ UI has 3840 states, 75500 transitions, and 41 events, and the minimal automaton realizing a nonblocking and maximally permissive controllable

and normal supervisor of the specification obtained by lifting UIReq to the alphabet of VAxis ‖ HAxis ‖ HVN ‖ UI has 211200 states and 2751680 transitions.

C. Experimental Time Complexity

From the experimental time-complexity viewpoint, the required times in seconds to perform all computations, including the input/output operations and the minimization of the constructed automata, are summarized in Table V.

In addition, for specifications HVReq and UIReq, we further include, in parentheses, the time of the computation of the global supervisor constructed for the modular systems $H = \{\text{VAxis}, \text{HAxis}, \text{HVN}\}$ and $H = \{\text{VAxis}, \text{HAxis}, \text{HVN}, \text{UI}\}$, respectively. In particular, the computation for UIReq and the corresponding modular plant $H = \{\text{VAxis}, \text{HAxis}, \text{HVN}, \text{UI}\}$ allocated more than 10 GB of memory in ca. 10 minutes, and ran out of memory (oom) in one hour and five minutes.

Table V
EXPERIMENTAL TIME COMPLEXITY IN SECONDS.

	VReq	HReq	HVReq (global)	UIReq (global)	Mono
Time	0.01	0.03	2.66 (9.37)	2.72 (oom)	4006

Finally, the computation of the nonblocking and maximally permissive controllable and normal supervisor constructed for the global specification VReq ‖ HReq ‖ HVReq ‖ UIReq and the monolithic plant VAxis ‖ HAxis ‖ HVN ‖ UI took more than one hour (≈ 4006 seconds). For completeness, we have computationally verified that the parallel composition of all the constructed local supervisors results in the nonblocking and maximally permissive monolithic supervisor.

D. Summary

We have constructed nine local supervisors. The total time of the computations and the overall size of the constructed local supervisors are summarized in Table VI (first column). All the considered local supervisors are nonblocking and maximally permissive in the sense that the resulting closed-loop system coincides with the nonblocking and maximally permissive monolithic closed-loop system. For comparison, we have included the monolithic approach (second column), and four monolithic approaches, one for each part of the specification (third column). The last column overviews the time and overall size of the high-level supervisors constructed by the hierarchical approach of Komenda and Masopust [31].

Table VI
THE SUMMARY OF RESULTS.

	$9 \times$ local	Monolithic	$4 \times$ global	$4 \times$ high
States	3334	68672	213359	3768
Trans.	26763	616000	2772156	31486
Time	5.42	4006	oom	ca. 11

VIII. CONCLUSIONS

We investigated supervisory control of modular discrete-event systems under partial observations, and showed that the concept of hierarchical supervisory control called modified observation consistency (MOC) can be used to guarantee that the global nonblocking and maximally permissive normal supervisor can be achieved locally as the parallel composition of local normal supervisors. Both the case of local specifications as well as the case of global specifications was considered.

We further showed that the global and local computations of nonblocking and maximally permissive normal supervisors coincide under the condition that all shared events are observable. This condition is stronger than MOC and nicely complements a similar condition of modular supervisory control under complete observation that all shared events are controllable.

An industrial case study of the patient table of an MRI scanner illustrated our results.

Finally, we would like to point out that it is worth combining both the modular approach and the hierarchical approach. This combination is particularly useful if the specification describes the required behavior in terms of high-level events, and it will very likely bring further improvements. However, this problem requires further investigation and we plan to discuss it in our future work.

ACKNOWLEDGMENT

This research was partially supported by the MŠMT under the INTER-EXCELLENCE project LTAUSA19098, and by the Czech Academy of Sciences under RVO 67985840.

REFERENCES

[1] C. Baier and T. Moor, "A hierarchical and modular control architecture for sequential behaviours," *Discrete Event Dynamic Systems*, vol. 25, no. 1-2, pp. 95–124, 2015.

[2] A. Sylla, M. Louvel, É. Rutten, and G. Delaval, "Modular and hierarchical discrete control for applications and middleware deployment in IoT and smart buildings," in *Conference on Control Technology and Applications (CCTA)*, 2018, pp. 1472–1479.

[3] J. Raisch and T. Moor, "Hierarchical hybrid control synthesis and its application to a multiproduct batch plant," in *Control and Observer Design for Nonlinear Finite and Infinite Dimensional Systems*. Springer Berlin Heidelberg, 2005, pp. 199–216.

[4] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*, 3rd ed. Springer, 2021.

[5] W. M. Wonham and K. Cai, *Supervisory control of discrete-event systems*. Springer, 2018.

[6] K. Cai, R. Zhang, and W. M. Wonham, "Relative observability of discrete-event systems and its supremal sublanguages," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 659–670, 2015, and its correction.

[7] M. V. S. Alves, L. K. Carvalho, and J. C. Basilio, "New algorithms for verification of relative observability and computation of supremal relatively observable sublanguage," *IEEE Transactions on Automatic Control*, vol. 62, no. 11, pp. 5902–5908, 2017.

[8] S. Takai and T. Ushio, "Effective computation of an $L_m(G)$-closed, controllable, and observable sublanguage arising in supervisory control," *Systems & Control Letters*, vol. 49, no. 3, pp. 191–200, 2003.

[9] J. G. Thistle and H. M. Lamouchi, "Effective control synthesis for partially observed discrete-event systems," *SIAM Journal on Control and Optimization*, vol. 48, no. 3, pp. 1858–1887, 2009.

[10] X. Yin and S. Lafortune, "Synthesis of maximally permissive supervisors for partially-observed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 5, pp. 1239–1254, 2016.

[11] W. Wonham and P. Ramadge, "Modular supervisory control of discrete-event systems," *Mathematics of Control, Signals, and Systems*, vol. 1, no. 1, pp. 13–30, 1988.

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

[13] R. C. Hill and D. M. Tilbury, "Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction," in *Workshop on Discrete Event Systems (WODES)*, 2006.

[14] B. Gaudin and H. Marchand, "An efficient modular method for the control of concurrent discrete event systems: A language-based approach," *Discrete Event Dynamic Systems*, vol. 17, no. 2, pp. 179–209, 2007.

[15] Y. Willner and M. Heymann, "Supervisory control of concurrent discrete-event systems," *International Journal of Control*, vol. 54, no. 5, pp. 1143–1169, 1991.

[16] S. Jiang and R. Kumar, "Decentralized control of discrete event systems with specializations to local control and concurrent systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 30, no. 5, pp. 653–660, 2000.

[17] K. Rohloff and S. Lafortune, "The verification and control of interacting similar discrete-event systems," *SIAM Journal on Control and Optimization*, vol. 45, no. 2, pp. 634–667, 2006.

[18] J. Komenda, T. Masopust, and J. H. van Schuppen, "Synthesis of controllable and normal sublanguages for discrete-event systems using a coordinator," *Systems & Control Letters*, vol. 60, no. 7, pp. 492–502, 2011.

[19] ——, "Coordination control of discrete-event systems revisited," *Discrete Event Dynamic Systems*, vol. 25, no. 1-2, pp. 65–94, 2015.

[20] H. Flordal, R. Malik, M. Fabian, and K. Åkesson, "Compositional synthesis of maximally permissive supervisors using supervision equivalence," *Discrete Event Dynamic Systems*, vol. 17, no. 4, pp. 475–504, 2007.

[21] S. Abdelwahed and W. Wonham, "Supervisory control of interacting discrete event systems," in *IEEE Conference on Decision and Control (CDC)*, vol. 2, 2002, pp. 1175–1180.

[22] S.-H. Lee and K. C. Wong, "Structural decentralised control of concurrent discrete-event systems," *European Journal of Control*, vol. 8, no. 5, pp. 477–491, 2002.

[23] K. Wong and W. Wonham, "Hierarchical control of discrete-event systems," *Discrete Event Dynamic Systems*, vol. 6, no. 3, pp. 241–273, 1996.

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

[25] K. Schmidt and C. Breindl, "Maximally permissive hierarchical control of decentralized discrete event systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 4, pp. 723–737, 2011.

[26] L. Feng, "Computationally efficient supervisor design for discrete-event systems," Ph.D. dissertation, University of Toronto, 2007. [Online]. Available: https://hdl.handle.net/1807/113031

[27] J. Komenda and J. H. van Schuppen, "Control of discrete-event systems with modular or distributed structure," *Theoretical Computer Science*, vol. 388, no. 3, pp. 199–226, 2007.

[28] ——, "Modular control of discrete-event systems with coalgebra," *IEEE Transactions on Automatic Control*, vol. 53, no. 2, pp. 447–460, 2008.

[29] R. Su and B. Lennartson, "Control protocol synthesis for multi-agent systems with similar actions instantiated from agent and requirement templates," *Automatica*, vol. 79, pp. 244–255, 2017.

[30] Y. Liu, J. Komenda, T. Masopust, and Z. Li, "Modular control of discrete-event systems using similarity," *Automatica*, vol. 142, p. 110431, 2022.

[31] J. Komenda and T. Masopust, "Hierarchical supervisory control under partial observation: Normality," *IEEE Transactions on Automatic Control*, pp. 1–12, 2023.

[32] F. Lin and W. M. Wonham, "On observability of discrete-event systems," *Information Sciences*, vol. 44, no. 3, pp. 173–198, 1988.

[33] L. Feng and W. Wonham, "Supervisory control architecture for discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1449–1461, 2008.

[34] J. Komenda, F. Lin, and J. H. van Schuppen, "A unifying approach to maximal permissiveness in modular control of discrete-event systems," in *Conference on Decision and Control (CDC)*, 2019, pp. 2009–2014.

[35] P. Pena, J. Cury, and S. Lafortune, "Verification of nonconflict of supervisors using abstractions," *IEEE Transactions on Automatic Control*, vol. 54, no. 12, pp. 2803–2815, 2009.

[36] R. Malik and S. Ware, "On the computation of counterexamples in compositional nonblocking verification," *Discrete Event Dynamic Systems*, vol. 30, no. 2, pp. 301–334, 2020.

[37] J. Komenda, T. Masopust, and J. H. van Schuppen, "On conditional decomposability," *Systems & Control Letters*, vol. 61, no. 12, pp. 1260–1268, 2012.

[38] R. Alur, *Principles of Cyber-Physical Systems*. The MIT Press, 2015.

[39] R. Theunissen, "Supervisory control in health care systems," Ph.D. dissertation, Technische Universiteit Eindhoven, 2015.

[40] "`libFAUDES` – a software library for supervisory control." [Online]. Available: https://fgdes.tf.fau.de/faudes/index.html