

Control of Modular and Distributed Discrete-Event Systems

Jan Komenda¹ and Jan H. van Schuppen²

¹ Institute of Mathematics - Brno Branch, Czech Academy of Sciences,
Žitkova 22, 616 62 Brno, Czech Republic

`komenda@ipm.cz`

² CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

`J.H.van.Schuppen@cwi.nl`

Abstract. Control of modular and distributed discrete-event systems appears as an approach to handle computational complexity of synthesizing supervisory controllers for large scale systems. For both modular and distributed discrete-event systems sufficient and necessary conditions are derived for modular control synthesis to equal global control synthesis for the supremal controllable sublanguage, for the supremal normal sublanguage, and for the supremal controllable and normal sublanguage. The modular control synthesis has a much lower computational complexity than the corresponding global control synthesis for the respective sublanguages.

1 Introduction

The purpose of the paper is to present an overview of recent results together with new results on control of modular (concurrent) and distributed discrete-event systems. Discrete-event systems (DES) are dynamical systems which are studied in computer science with applications in manufacturing, communication networks, but also in software engineering (automated system design). In particular the various types of state transition systems (automata, Petri Nets, process algebras) are typical instances of DES. The topic of modular DES arises because of an increasing complexity of engineering systems, in particular of computer and communication networks. There is a strong need for system theoretical treatment of modular DES motivated by these emerging application fields. Control of discrete-event systems is a natural generalization of their verification that is now very well established for both finite and infinite state transition systems.

In computer science the problems of supervisory control synthesis are studied as automated synthesis. In control theory for DES the goal is not to verify the specification, but to impose it by means of a supervisor that runs in parallel with the original system. The supervisor is chosen such that the composed system meets the specification. In the Ramadge-Wonham framework it is an automaton which runs in parallel with the original system and the parallel composition of the original system with the supervisor (called closed-loop or controlled system)

meets the specification given mostly by a language or a logical formula. In this way the specification is imposed on the controlled system.

Although most of the verification and control problems for finite-state transition systems are decidable, the high complexity of most control and verification problems makes them practically difficult. Moreover, there are undecidable control problems for decentralized DES [10], [16]. In order to limit the high computational complexity of (global) control synthesis efficient methods for component-based control synthesis are developed. Synthesis of modular and distributed systems has also been treated by computer scientists, see for example [13].

A discrete-event system is said to have *complete observations* if all events are observed and are available for the supervisory control. A discrete-event system is said to have *partial observations* if only a strict subset of the events are observed and are available for the supervisory control. A *modular discrete-event system* is a system consisting of a composition of two or more subsystems where each subsystem or module has complete observations of its (local) events. A *distributed discrete-event system* is a system consisting of a composition of two or more subsystems where at least one subsystem has only partial observations of its events.

The novelty of the paper is in the following results: The necessary conditions for commutativity between parallel composition and supremal sublanguages (Theorems 3, 4, and 5; supremal controllable sublanguages, supremal normal sublanguages, and supremal controllable and normal sublanguages), necessary conditions in case of indecomposable specifications (Theorems 7, 8, and 9), and necessary and sufficient conditions for closed-loop languages with respect to the antipermissive control policy in case of global specification (Theorems 6 and 10).

Because this paper appears in a proceedings of a theoretical computer science symposium, there is an additional tutorial text in Section 2 for computer scientists on the concepts and results of control theory for discrete-event systems.

The paper has the following structure. The next section is an introduction to supervisory control. Section 3 is devoted to modular control with complete observations and with decomposable specification languages. In Section 4 the case of a distributed DES and a decomposable specification is treated. In Section 5 the case is discussed of a modular DES with an indecomposable specification. Finally in Section 6 the case of a distributed system with an indecomposable specification is treated. In Section 7 concluding remarks are proposed.

2 Control of discrete-event systems - Introduction

In this section basic notation and terminology of supervisory control is recalled. The notation used in this paper is mostly taken from the lecture notes of W.M. Wonham [19] and the book [1].

A (deterministic) *generator*

$$G = (Q, A, f, q_0, Q_m),$$

is an algebraic structure consisting of a *state set* Q , an *event set* A , a *partial transition function* $f : Q \times A \rightarrow Q$, an *initial state* $q_0 \in Q$, and a *subset of marked states* $Q_m \subseteq Q$. A transition is also denoted as $q \xrightarrow{a} q^+ = f(q, a)$. If a transition is defined then this is denoted by $f(q, a)!$. Denote by A^* the set of all finite strings of elements of the alphabet A and the empty string. Extend the transition function f to $f : Q \times A^* \rightarrow Q$ by induction. Define respectively the *language* and the *marked language* of the generator as,

$$L(G) = \{s \in A^* | f(q_0, s)!\}, \quad L_m(G) = \{s \in L(G) | f(q_0, s) \in Q_m\}.$$

Note that unlike $L_m(G)$, $L(G)$ is always prefix-closed. The prefix closure of a language $K \subseteq A^*$ is denoted by $\text{prefix}(K)$. We often abuse notation and write L instead of $L(G)$. The tuple of languages $(L_m(G), L(G))$ will be called the *behavior* of the generator. The system is said to be *nonblocking* if the the prefix closure of the marked language $L_m(G)$ equals the language $L(G)$. This is equivalent to the property that every string of the system, $s \in L(G)$, can be extended to a marked string, thus there exists a string $v \in A^*$ such that $sv \in L_m(G)$.

A *controlled generator* is a structure

$$(G, A_c, \Gamma_c), \text{ where,}$$

- G is a generator,
- $A_c \subseteq A$ is the subset of *controllable events*,
- $A_{uc} = A \setminus A_c$ is the subset of *uncontrollable events*, and
- $\Gamma_c = \{S \subseteq A | A_{uc} \subseteq S\}$, is called the *set of control patterns*.

The set of control patterns $S(s)$ is the subset of events that the supervisor enables after string s has been generated by G . A *supervisory control* or a *supervisor* for the controlled generator is map $S : L(G) \rightarrow \Gamma_c$. The *closed-loop system* associated with a controllable generator and a supervisory control as denoted above is defined as the language $L(S/G) \subseteq A^*$ and the marked language $L_m(S/G) \subseteq L(S/G)$ which satisfy respectively,

- (1) $\epsilon \in L(S/G)$,
- (2) if $s \in L(S/G)$ and if $a \in S(s)$ such that $sa \in L(G)$ then $sa \in L(S/G)$;

$$L_m(S/G) = L(S/G) \cap L_m(G).$$

Note that at the automata level the supervision is implemented by a parallel composition of the generator and the supervisor. This composition is a special form of the synchronous product (with priorities), in order to ensure that a supervisor never disables uncontrollable events.

Problem 1. Supervisory control problem. Consider a controlled generator (G, A_c, Γ_c) and a specification sublanguage $K \subset L_m(G)$. Does there exist a supervisor S such that the closed-loop system satisfies (1) $L_m(S/G) \subseteq K$ and (2) $L(S/G)$ is nonblocking?

Because often the subset of the specification sublanguage K contains only the safe strings, thus the unsafe strings are excluded, the control objective (1) of the above problem is called the *safety control objective*. Not every language can be exactly achieved by a supervisor. The property called controllability is needed.

Definition 1. A language $K \subseteq A^*$ is said to be controllable with respect to plant language $L = L(G)$ and alphabet A_{uc} if $\forall s \in \text{prefix}(K)$ and $\forall a \in A_{uc}$ such that $sa \in L$ we have that $sa \in \text{prefix}(K)$. Equivalently,

$$\text{prefix}(K)A_{uc} \cap L \subseteq \text{prefix}(K). \quad (1)$$

Theorem 1. (Due to Ramadge, Wonham, see [14].) There exists a nonblocking supervisory control S for a generator G such that $L_m(S/G) = K$ and $L(S/G) = \text{prefix}(K)$ if and only if

1. K is controllable with respect to $L(G)$ and A_{uc} and
2. $K = \text{prefix}(K) \cap L_m(G)$ (then one says that K is $L_m(G)$ -closed.).

As a corollary for prefix-closed specifications:

Corollary 1. Let $\emptyset \neq K \subseteq L(G)$ be prefix closed. There exists a supervisory control S for G such that $L_m(S/G) = K$ and $L(S/G) = K$ if and only if K is controllable with respect to $L(G)$ and A_{uc} .

The corresponding supervisory control $S : L(G) \rightarrow \Gamma_c$ is:

$$S(s) = A_{uc} \cup \{a \in A_c : sa \in \text{prefix}(K)\}.$$

Note the abuse of notation, where the same symbol is used to denote a supervisor and a control law. This is justified by the fact that, considering a supervisor, the control law is described by the transition function of the supervisor in the form of the set of active events after string s has been processed by the supervisor. Most often one is concerned only with the safety issue, i.e. the controlled behavior must be included in the specification language. This is why for specifications which are not controllable, supremal controllable sublanguages are considered. The notation $\text{supC}(K, L, A_u)$ is chosen for the supremal controllable sublanguage of K with respect to L and A_u . This language always exists, it is the union of all controllable sublanguages because controllability is preserved by language unions.

In the presence of partial observations additional issues appear. A *generator with partial observations* is a structure (G, A_o) where G is a generator, $A_o \subseteq A$ is called the subset of *observable events*, and $A_{uo} = A \setminus A_o$ is called the subset of *unobservable events*. In this case define the *natural projection* $P : A^* \rightarrow A_o^*$ such that $P(\epsilon) = \epsilon$ and P erases only the unobservable events. Note that the supervisor cannot distinguish between two strings with the same projections, i.e. after two such strings the same control law must be applied. Therefore, a supervisor with partial observations is a map $S : P(L(G)) \rightarrow \Gamma_c$. Define also the inverse projection $P_i^{-1} : \text{Pwr}(A_i^*) \rightarrow \text{Pwr}(A^*)$ on subsets of strings or languages.

Let K be a specification language. The supervisory control with partial observations is:

$$S(s) = A_{uc} \cup \{a \in A_c : \exists s' \in \text{prefix}(K) \text{ with } P(s') = P(s) \text{ and } s'a \in \text{prefix}(K)\}. \quad (2)$$

The additional property needed to exactly achieve a specification language by a supervisor with partial observations is called observability.

Definition 2. *The sublanguage $K \subseteq P(L)$ is said to be observable with respect to the plant language L and the projection P if*

$$\forall s \in \text{prefix}(K), \forall a \in A_c, sa \in L, s'a \in \text{prefix}(K), \text{ and } P(s) = P(s') \Rightarrow sa \in \text{prefix}(K). \quad (3)$$

Theorem 2. *(Due to F. Lin and W.M. Wonham, see [11].) Consider a generator with partial observations. There exists a nonblocking supervisory control S with partial observations such that $L_m(S/G) = K$ and $L(S/G) = \text{prefix}(K)$ if and only if*

1. K is controllable with respect to $L(G)$ and A_{uc} ,
2. K is observable with respect to $L(G)$ and P , and
3. $K = \text{prefix}(K) \cap L_m(G)$. (K is $L_m(G)$ -closed.)

Unfortunately, unlike controllability, observability is not preserved by language unions. This is why a stronger property, called normality, has been introduced.

Definition 3. [1] *Consider a controlled generator with partial observations and a specification sublanguage $K \subseteq L_m(G)$. Call the specification sublanguage K (L, P) -normal if*

$$\text{prefix}(K) = P^{-1}P(\text{prefix}(K)) \cap L. \quad (4)$$

It is known that normal languages are closed with respect to unions, hence the supremal normal sublanguage of K always exists, it is the union of all normal sublanguages of K and it is denoted by $\text{supN}(K, L, P)$.

Recall finally that in the case $A_c \subseteq A_o$ normality coincides with observability. This assumption is widely accepted in the computer science community, where only uncontrollable actions might be internal (silent).

For control problems with partial observations and a safety control objective, *supremal controllable and normal sublanguages* are important.

Recall that the synchronous product (also called the parallel composition) of languages $L_1 \subseteq A_1^*$ and $L_2 \subseteq A_2^*$ is defined by $L = L_1 \parallel L_2 = \bigcap_{i=1}^2 P_i^{-1}(L_i)$, where $P_i : A^* \rightarrow A_i^*$, $i = 1, 2$ are natural projections to local event sets.

Definition 4. *Consider two generators,*

$$G_1 = (Q_1, A_1, f_1, q_{1,0}, Q_{1,m}), G_2 = (Q_2, A_2, f_2, q_{2,0}, Q_{2,m}).$$

Their synchronous product is the generator

$$\begin{aligned}
G_1 \parallel G_2 &= (Q_1 \times Q_2, A_1 \cup A_2, f, q_0, Q_m), \\
q_0 &= (q_{1,0}, q_{2,0}), \quad Q_m = Q_{1,m} \times Q_{2,m}, \\
f((q_1, q_2), a) &= \begin{cases} (f_1(q_1, a), f_2(q_2, a)) & \text{if } a \in A_1 \cap A_2, \\ & i = 1, 2, f_i(q_i, a)! \\ (f_1(q_1, a), q_2), & \text{if } a \in A_1 \setminus A_2, f_1(q_1, a)! \\ (q_1, f_2(q_2, a)), & \text{if } a \in A_2 \setminus A_1, f_2(q_2, a)! \\ \text{undefined}, & \text{otherwise.} \end{cases}
\end{aligned}$$

It can then be proven that,

$$L(G_1 \parallel G_2) = L(G_1) \parallel L(G_2), \quad L_m(G_1 \parallel G_2) = L_m(G_1) \parallel L_m(G_2).$$

Denote for $n \in \mathbb{Z}$ the set of the first n integers by $\mathbb{Z}_n = \{1, 2, \dots, n\}$.

Definition 5. A modular discrete-event system (also called a concurrent discrete-event system) is the synchronous product of two or more modules or local subsystems in which each module has complete observations of the state of its own module but does not have observations of the states of other modules unless it shares observable events with these other modules. Mathematically, a modular discrete-event system with $n \in \mathbb{Z}$ modules is a structure $\{G_i, A_{i,c}, A_{i,o}, \Gamma_{i,c}, i \in \mathbb{Z}_n\}$ consisting of n controlled generators. The associated global system is the synchronous product of the modules or the local subsystems, $\parallel_{i=1}^n G_i$. Denote the natural projections by $P_i : (\cup_{i=1}^n A_i)^* \rightarrow A_i^*$.

A distributed discrete-event system is a structure as above consisting of n controlled generators where at least one of the modules has only partial observations of that module.

3 Modular supervisory control with a decomposable specification

In this section the concurrent behavior of the modules $\{G_i, i \in \mathbb{Z}_n\}$ is considered. Consider the local alphabets of these subplants, $\{A_i, i \in \mathbb{Z}_n\}$, which are not necessarily pairwise disjoint. Denote the partition of the local alphabet into the subset of controllable events, $A_{i,c}$, and the subset of uncontrollable events, $A_{i,u}$, by $A_i = A_{i,u} \cup A_{i,c}$ for all $i \in \mathbb{Z}_n$.

Definition 6. Consider a modular DES. The local plants $\{G_i, i \in \mathbb{Z}_n\}$ agree on the controllability of their common events if

$$A_{i,u} \cap A_j = A_i \cap A_{j,u}, \quad \forall i, j \in \mathbb{Z}_n. \quad (5)$$

This definition stemming from [18] means that the events shared by two modules or local subsystems must have the same control status for both controllers associated to these subsystems. In the following it will often be assumed that the modules satisfy the condition of agreement on the controllability of their

common events. Denote $A_c = \cup_{i=1}^n A_{ic}$. The assumption on the agreement on common events implies that $A_{ic} = A_c \cap A_i$. Also, if we denote $A_u = \cup_{i=1}^n A_{iu}$ then we still have the disjoint union $A = A_c \cup A_u$ due to the assumption of agreement on the controllability of their common events. Denote by $A = \cup_{i=1}^n A_i$ the global alphabet and by $P_i : A \rightarrow A_i$ the projections to the local alphabets. The concept of inverse projection $P_i^{-1} : \text{Pwr}(A_i) \rightarrow \text{Pwr}(A)$ is also used.

The local plant languages or the languages of the modules will be denoted by $\{L_i, i \in \mathbb{Z}_n\}$ and the local specification languages by $\{K_i, i \in \mathbb{Z}_n\}$. We assume in this section that the global plant L and the specification K languages are decomposable into local plant and local specification languages: $L = \parallel_{i=1}^n L_i$ and $K = \parallel_{i=1}^n K_i$. This formulation is equivalent to the following definition.

Definition 7. Consider a modular DES. One says that $L \subseteq A^*$ is decomposable with respect to $\{P_i, i \in \mathbb{Z}_n\}$ if $L = \parallel_{i=1}^n P_i(L)$.

Proposition 1. Consider a modular DES. $L \subseteq A^*$ is decomposable with respect to $\{P_i, i \in \mathbb{Z}_n\}$ if and only if there exists $\{L_i \subseteq A_i^*, i \in \mathbb{Z}_n\}$ such that

$$L = \parallel_{i=1}^n L_i = \cap_{i=1}^n (P_i)^{-1}(L_i). \quad (6)$$

Proof. (\Rightarrow) It is sufficient to consider $L_i := P_i(L)$.

(\Leftarrow) If $L = \parallel_{i=1}^n L_i = \cap_{i=1}^n (P_i)^{-1}(L_i)$, then it follows from properties of projections that

$$\begin{aligned} P_i(L) &\subseteq L_i \cap \cap_{j \neq i} P_i P_j^{-1}(L_j) \subseteq L_i, \forall i \in \mathbb{Z}_n. \text{ Thus we have that,} \\ \parallel_{i=1}^n P_i(L) &= \cap_{i=1}^n (P_i)^{-1} P_i(L) \subseteq \cap_{i=1}^n P_i^{-1}(L_i) = L, \end{aligned}$$

by our assumption. The first inclusion follows from the fact that $P_i(P_i)^{-1}$ is identity and that projection of an intersection is contained in the intersection of projections. The inclusion $L \subseteq \cap_{i=1}^n P_i^{-1}(P_i(L))$ is obvious.

Note that $P_i(L) \subseteq L_i$ means that for any tuple of languages $\{L'_i \subseteq A_i^*, \forall i \in \mathbb{Z}_n\}$ such that $L = \parallel_{i=1}^n L'_i$ we have $P_i(L) \subseteq L'_i$, i.e. $\{P_i(L), i \in \mathbb{Z}_n\}$ is the smallest possible decomposition of L into local languages.

Definition 8. Consider a modular discrete-event system and either a global specification language or a family of local specifications. From the local specifications one can always compute the global specification as described above.

Global control synthesis of a modular discrete-event system is the procedure by which first all modules are combined into the global plant and then control synthesis is carried out as described in Section 2. This can refer to either construction of a supervisor which meets the specification or to the supremal supervisor.

Modular control synthesis of a modular discrete-event system is the procedure by which control synthesis is carried out for each module or local subsystem. The global supervisor formally consists of the synchronous product of the local supervisors though that product is not computed in practice.

In terms of behaviors, the optimal global control synthesis is represented by the closed-loop language

$$\sup C(K, L, A_u) = \sup C(\|_{i=1}^n K_i, \|_{i=1}^n L_i, A_u),$$

using the operation supremal controllable sublanguage $\sup C(K, L, A_u)$ defined in the last section. Similarly, modular control synthesis yields in terms of behaviors the partial language

$$\|_{i=1}^n \sup C(K_i, L_i, A_{iu}).$$

Problem 2. Consider a modular discrete-event system and a decomposable specification language. Determine necessary and sufficient conditions with respect to which modular control synthesis equals global control synthesis for the supremal controllable sublanguage within the specification language. Equivalently, if,

$$\|_{i=1}^n \sup C(K_i, L_i, A_{iu}) = \sup C(\|_{i=1}^n K_i, \|_{i=1}^n L_i, A_u). \quad (7)$$

Later in the paper also the problems are investigated of when modular control synthesis equals global control synthesis for the supremal normal sublanguage, for the supremal controllable and normal sublanguage, and for the closed-loop language in case of an antipermissive control policy.

There exists an example which establishes that modular control synthesis does not equal global control synthesis in general. Theorem 3 provides necessary and sufficient conditions for modular control synthesis to equal global control synthesis. This problem has been studied algebraically in [18]. The concept of mutual controllability ([18]) plays the key role.

Definition 9. Consider a modular DES. The modular plant languages $\{L_i \subseteq A_i^*, i \in \mathbb{Z}_n\}$ are called mutually controllable if

$$\text{prefix}(L_j)(A_{ju} \cap A_i) \cap P_j(P_i)^{-1} \text{prefix}(L_i) \subseteq \text{prefix}(L_j), \quad \forall i, j \in \mathbb{Z}_n, i \neq j. \quad (8)$$

Note that both local and global plant languages are typically prefix closed, hence the prefix closures above can be removed.

Mutual controllability can be viewed as local controllability of the modular plant languages with respect to the shared uncontrollable events; thus, for all $i, j \in \mathbb{Z}_n$ with $i \neq j$, the modular language L_j is controllable with respect to the shared uncontrollable events $(A_{ju} \cap A_i)$ and the local view of the other module $(P_i(P_j)^{-1}(L_j))$ as the new plant. This condition is important for modular computation of global supremal controllable sublanguages.

The computational complexity of checking mutual controllability is much lower than that of checking controllability of a sublanguage over the global alphabet.

The sufficiency part of Theorem 3 is due to K.C. Wong and S.-H. Lee, see [18]. We have presented in [3] a coalgebraic version of the proof for commutativity of supremal controllable sublanguages with the synchronous product:

Theorem 3. Modular control synthesis equals global control synthesis for the supremal controllable sublanguage in case of a modular DES. Assume that the local plants agree on the controllability of their common events.

If the local plant languages $\{L_i \subseteq A_i^*, i \in Z_n\}$ are mutually controllable then

$$\|_{i=1}^n \sup C(K_i, L_i, A_{iu}) = \sup C(\|_{i=1}^n K_i, \|_{i=1}^n L_i, A_u). \quad (9)$$

Conversely, if for fixed local plant languages $\{L_i, i \in Z_n\}$ equation (9) holds true for any $\{K_i \subseteq L_i, i \in Z_n\}$ then, for all $i \in Z_n$, $P_i(L)$ is controllable with respect to L_i and A_{iu} ; equivalently, then

$$P_i(L)A_{iu} \cap L_i \subseteq P_i(L), \forall i \in Z_n. \quad (10)$$

Proof. We have shown the first part of the theorem in [3], the proof is stated in [6]. Note that an algebraic proof of sufficiency is stated in [18].

It remains to prove the necessity part of the theorem. Assume now that for fixed local plant languages $\{L_i, i \in Z_n\}$ equation (9) holds true for any local specification languages $K_i \subseteq L_i$. Then it holds true in particular for $K_i := P_i(L)$. Since L and K are decomposable we have

$$L = \bigcap_{i=1}^n P_i^{-1}P_i(L) = \bigcap_{i=1}^n P_i^{-1}K_i = K, \text{ (cf. Proposition 1)}. \quad (11)$$

$$\begin{aligned} L = K &= \sup C(L, L, A_u) = \|_{i=1}^n \sup C(P_i(L), L_i, A_{iu}) \\ &\subseteq \cap_{i=1}^n P_i^{-1}(\sup C(P_i(L), L_i, A_{iu})), \\ L &\subseteq (P_i)^{-1}(\sup C(P_i(L), L_i, A_{iu})), \forall i \in Z_n; \end{aligned} \quad (12)$$

Note that $P_i(L) \subseteq L_i, \forall i \in Z_n$, because L is decomposable. Indeed

$$P_i(L) = P_i\left(\bigcap_{i=1}^n P_i^{-1}(L_i)\right) \subseteq P_i(P_i)^{-1}(L_i) \cap \bigcap_{j \neq i} P_i(P_j)^{-1}(L_j) \subseteq L_i,$$

where the last inclusion follows from $P_i(P_i)^{-1}(L_i) = L_i$.

By applying the projection on both sides of inclusion (12)

it follows from the monotonicity of projections that

$$P_i(L) \subseteq P_i(P_i)^{-1} \sup C(P_i(L), L_i, A_{iu}) = \sup C(P_i(L), L_i, A_{iu}) \subseteq P_i(L),$$

from the definition of supremal controllable sublanguages. Hence,

$$P_i(L) = \sup C(P_i(L), L_i, A_{iu}),$$

which means that for all $i \in Z_n$, $P_i(L)$ is controllable with respect to L_i and A_{iu} .

4 Distributed supervisory control with a decomposable specification

In this section we consider the situation of a distributed plant, thus for which at least one module or subsystem does not have complete observations but only

partial observations. For each local plant, the local alphabet admits a partition, denoted by $A_i = A_{o,i} \cup A_{uo,i}, \forall i \in \mathbb{Z}_n$ into locally observable and locally unobservable event sets. The global system has observation set $A_o = \cup_{i=1}^n A_{o,i} \subseteq A = \cup_{i=1}^n A_i$. Globally unobservable events are denoted by $A_{uo} = A \setminus A_o$ and locally unobservable events by $A_{uo,i} = A_i \setminus A_{o,i}$. The projections of the global alphabet into the local ones are denoted by $P_i : A^* \rightarrow A_{o,i}^*, i = 1, 2$. Partial observations in individual modules are expressed via local projections $P_i^{loc} : A_i^* \rightarrow A_{o,i}^*$, while the global projection is denoted by $P : A^* \rightarrow A_o^*$.

Definition 10. Consider a distributed DES. The local plants are said to agree on the observability of their common events if

$$A_{o,i} \cap A_j = A_i \cap A_{o,j}, \forall i, j \in \mathbb{Z}_n. \quad (13)$$

The problem is to determine necessary and sufficient conditions for modular control synthesis to equal global control synthesis for a distributed control system. In Theorem 4 a condition similar to mutual controllability is needed. By analogy it is called mutual normality.

Definition 11. Consider a distributed DES. Local plant languages $\{L_i \subseteq A_i^*, i \in \mathbb{Z}_n\}$ are called mutually normal if

$$(P_i^{loc})^{-1} P_i^{loc}(L_i) \cap P_i(P_j)^{-1}(L_j) \subseteq L_i, \forall i, j \in \mathbb{Z}_n, i \neq j. \quad (14)$$

Mutual normality can be viewed as normality of the local plant languages with respect to the local views of the other plant languages.

Theorem 4. Modular control synthesis equals global control synthesis for supremal normal sublanguages in case of a distributed DES. Assume that the local plants agree on the observability of their common events.

If $\{L_i \subseteq A_i^*, i \in \mathbb{Z}_n\}$ are mutually normal then

$$\sup N(\|_{i=1}^n K_i, \|_{i=1}^n L_i, P) = \|_{i=1}^n \sup N(K_i, L_i, P_i^{loc}). \quad (15)$$

Conversely, if for fixed local plant languages $\{L_i, i \in \mathbb{Z}_n\}$ equation (15) holds true for any $\{K_i \subseteq L_i, i \in \mathbb{Z}_n\}$, then

$$P_i(L) \text{ are normal with respect to } L_i \text{ and } P_i^{loc}, \forall i \in \mathbb{Z}_n. \quad (16)$$

Proof. The sufficiency part has been shown in [6], where the coinductive proof principle has been used. Now we show the necessity part of the theorem. Assume now that for fixed local plant languages $\{L_i, i \in \mathbb{Z}_n\}$ equation (15) holds true. Consider any $i \in \mathbb{Z}_n$ and any local specification languages $K_i \subseteq L_i$. Then it holds true in particular for $K_i := P_i(L)$. Since L is decomposable we have

$$\begin{aligned} L &= \bigcap_{i=1}^n P_i^{-1} P_i(L) = K; \\ L &= \sup N(L, L, A_u) = \|_{i=1}^n \sup N(P_i(L), L_i, P_i^{loc}). \\ L &\subseteq (P_i)^{-1} \sup N(P_i(L), L_i, P_i^{loc}), \forall i \in \mathbb{Z}_n. \end{aligned} \quad (17)$$

$$P_i(L) \subseteq L_i, \text{ because } L \text{ is decomposable.} \quad (18)$$

By applying the projection P_i on both sides of the inclusion (17)

it follows from the monotonicity of projections that

$$P_i(L) \subseteq P_i(P_i)^{-1} \sup N(P_i(L), L_i, P_i^{loc}) = \sup N(P_i(L), L_i, P_i^{loc}).$$

$$\sup N(P_i(L), L_i, P_i^{loc}) \subseteq P_i(L),$$

from the definition of supremal normal sublanguages.

$$P_i(L) = \sup N(P_i(L), L_i, P_i^{loc}),$$

which means that for all $i \in \mathbb{Z}_n$, $P_i(L)$ is normal with respect to L_i and P_i^{loc} .

The proof of the above theorem for one direction depends only on the assumption that the local plants agree on the observability of their common events. Hence the following corollary is obtained.

Corollary 2. *If the local plants agree on the observability of their common events then we have*

$$\sup N(\|_{i=1}^n K_i, \|_{i=1}^n L_i, P) \supseteq \|_{i=1}^n \sup N(K_i, L_i, P_i^{loc}). \quad (19)$$

Theorem 4 is useful for the computation of (global) supremal normal sublanguages of large distributed plants. If the conditions of the theorem are satisfied, then it is sufficient to compute local supremal normal sublanguages and to synchronize them.

The interest of this theorem should be clear: Under the conditions which are stated it is possible to do the optimal (least restrictive) control synthesis with partial observations locally, and this represents an exponential savings on the computational complexity and makes in fact the optimal control synthesis of large distributed plants feasible.

Let us introduce the notation $\sup CN(K, L, P, A_u)$ for the supremal (L, P) -normal and controllable sublanguage of K with respect to A_u . Using a single-step algorithm for computation of supremal controllable and normal sublanguages ([8]), we have proven in [4] the sufficiency part of the following theorem:

Theorem 5. *Modular control synthesis equals global control synthesis for supremal controllable and normal sublanguage in case of a distributed DES. Assume that the local plants agree on the controllability of their common events and on the observability of their common events.*

If the local plant languages $\{L_i \subseteq A_i^, i \in \mathbb{Z}_n\}$ are mutually controllable and mutually normal then*

$$\|_{i=1}^n \sup CN(K_i, L_i, P_i^{loc}, A_{iu}) = \sup CN(\|_{i=1}^n K_i, \|_{i=1}^n L_i, P, A_u). \quad (20)$$

Conversely, if for fixed local plant languages $\{L_i, i \in \mathbb{Z}_n\}$ equation (20) holds true for any $\{K_i \subseteq L_i, i \in \mathbb{Z}_n\}$, then for any $i, j \in \mathbb{Z}_n$ with $i \neq j$, $\{P_i(L), i \in \mathbb{Z}_n\}$ are (1) normal with respect to L_i and P_i^{loc} and (2) controllable with respect to L_i and A_{iu} .

Proof. The sufficiency was shown in [4]. In view of the preceding theorems for sup C and sup N the necessity part follows.

In [18] there is a procedure to change a plant which does not satisfy the mutual controllability condition to one that satisfies it. It may be that a similar procedure can be found in the future for mutual normality. Nevertheless one cannot hope to find a universal procedure how to make a set of local plant languages mutually normal. Indeed, in the shuffle case mutual normality cannot hold as we show in the next section. However specific methods exist for this simpler case.

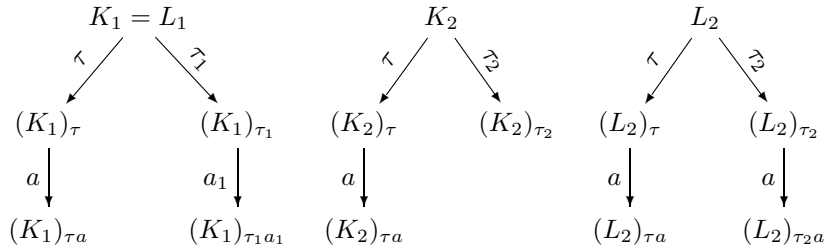
Example and Verification of Sufficient conditions

The purpose of this section is mainly to illustrate our results with an example. Before starting with concrete examples we consider several extreme cases of distributed DES. First of all, if all event alphabets are disjoint, the so called shuffle case, we notice that $P_i(P_j)^{-1}(L_j) = A_i^*$ for any $L_j \subseteq A_j^*$. This means that the condition of mutual normality cannot be satisfied. The intuitive reason is that there is no relation between local subsystems in this case. This is not surprising, because the observations of local agents are in this case completely independent and therefore there is a huge gap between local and global observations.

On the other hand, it is obvious from the definition of mutual normality that in the case of full local observations (all P_i^{loc} 's become identity mappings), mutual normality is trivially satisfied. Another extreme case occurs when all subsystems have the same event alphabets. Then all the P_i 's are identity mappings, i.e. the mutual normality becomes usual normality between two languages in a slightly more general sense (the assumption is lifted that one of the languages is a sublanguage of the other). This might justify why we call our condition mutual normality, it is a symmetric notion of normality.

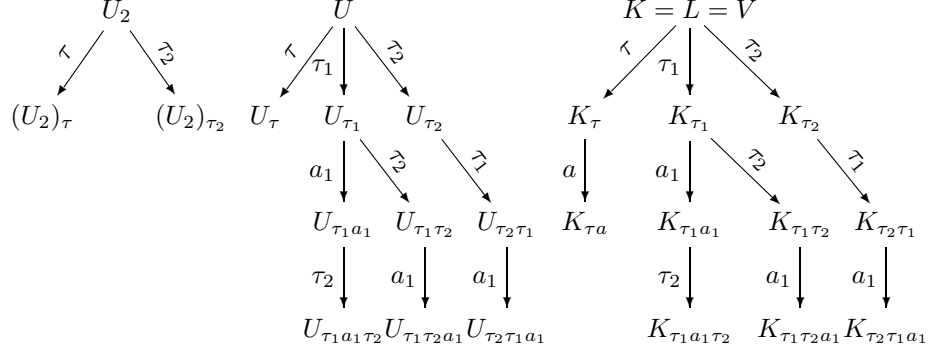
We show an example of a plant composed of two modules, where the commutativity between the supremal normal sublanguages and parallel product does not hold. Therefore mutual normality does not hold either.

Example 1. Let $A = \{a, a_1, a_2, \tau, \tau_1, \tau_2\}$, $A_1 = \{a_1, \tau_1, a, \tau\}$, $A_2 = \{a_2, \tau_2, a, \tau\}$, $A_o = \{a_1, a_2, a\}$, $A_{o,1} = \{a_1, a\}$, and $A_{o,2} = \{a_2, a\}$. Consider the following plant languages and specification sublanguages (the marked languages are not considered):



We use the notation $U_1 = \sup N(K_1, L_1, P_1^{loc})$, $U_2 = \sup N(K_2, L_2, P_2^{loc})$, $U = \sup N(K_1, L_1, P_1^{loc}) \parallel \sup N(K_2, L_2, P_2^{loc})$, and $V = \sup N(K_1 \parallel K_2, L_1 \parallel L_2, P^{loc})$.

L_2, P). We have trivially that $U_1 = K_1 = L_1$. It is easy to see that $U_2 = \sup \mathbf{N}(K_2, L_2, P_2^{loc}) = \{\varepsilon, \tau, \tau_2\}$. Computing the parallel products $K = K_1 \parallel K_2$ and $L = L_1 \parallel L_2$ yields $K = L$, i.e. we obtain trivially $K = L = V$ as is shown in the diagram below, where $U = U_1 \parallel U_2$ is also computed:



Thus, $U \neq V$, because $U_\tau \not\stackrel{a}{\rightarrow}$, while $V_\tau \stackrel{a}{\rightarrow}$. Therefore we only have the strict inclusion $U \subset V$ and the commutativity studied in this paper does not hold for this example. According to Theorem 4 mutual normality cannot hold. Indeed, we have

$$(P_1^{loc})^{-1}P_1^{loc}(L_1) \cap P_1(P_2)^{-1}(L_2) = \tau_1^*(\tau\tau_1^*a_1 + a_1\tau_1^*\tau)\tau_1^*,$$

but we have e.g. $(\tau_1)^n \notin L_1$ for $n \geq 2$.

Antipermissive control policy

The standard permissive control law is useful for safety control problems only if the specification is observable, otherwise it yields infimal coobservable and controllable superlanguages of K . If K represents safety specifications then these are violated if permissive control policy is applied.

There exists a dual control policy for DES with partial observations, called *antipermissive*. For the supervisor to enable event $a \in A$ it is necessary that all indistinguishable events corresponding to this trace can be prolonged within K (in permissive control policy it is sufficient that one of those strings can be prolonged within K). The interest of the antipermissive control policy is in its safety and the fact that the synthesized languages are observable languages in general larger than supremal normal sublanguages.

Denote by V the supervisor associated with a partial automaton S , and the corresponding antipermissive control policy by $S_A : P(L(G)) \rightarrow \Gamma_c$, where Γ_c is the class of enabled events, also called control patterns (i.e. supersets of the event subset A_u that are always enabled). Algebraically, the antipermissive control policy is defined as follows:

$$S_A(s) = A_{uc} \cup \left\{ a \in A_c : \forall s' \in \text{prefix}(K) \cap P^{-1}P(s) \right. \\ \left. (s'a \in L \Rightarrow s'a \in \text{prefix}(K)) \right\}. \quad (21)$$

Similarly as for the permissive control policy the supervisor marks all states that have been marked in the plant and that 'survive' under supervision.

We have formulated in [8] a single-step algorithm for computation of closed-loop languages with respect to the antipermissive control policy. This algorithm has been used in [5] for deriving sufficient conditions under which these languages are preserved by modular (local) control synthesis.

Denote for all $i \in \mathbb{Z}_n$ by $\text{AP}(K_i, L_i, P_i^{loc})$ the closed-loop language corresponding to the local antipermissive control synthesis (with local projection P_i^{loc} , local DES L_i , and local specification K_i). Similarly, $\text{AP}(\|_{i=1}^n K_i, \|_{i=1}^n L_i, P)$ stands for closed-loop language corresponding to the global antipermissive control synthesis.

In Theorem 6 a condition similar to mutual controllability (Wong and Lee, 2002) is used. By analogy it is called mutual observability.

Definition 12. Consider a distributed DES. Local plant languages $\{L_i \subseteq A_i^*, i \in \mathbb{Z}_n\}$ are called mutually observable if

$$\begin{aligned} & \forall i, j \in \mathbb{Z}_n, i \neq j, \forall s, s' \in L_i \text{ and } a \in A_{ic} : \\ & (sa \in P_i(P_j)^{-1}(L_j) \text{ and } P_i^{loc}(s) = P_i^{loc}(s') \text{ and } s'a \in L_i) \\ & \Rightarrow sa \in L_i. \end{aligned} \quad (22)$$

Now we extend the main result of [5] where only sufficient conditions were presented.

Theorem 6. Modular control synthesis equals global control synthesis for the closed-loop language in case of a completely-observed DES using an antipermissive control policy. Assume that the modular plants agree on the observability of their common events.

If the local plant languages $\{L_i \subseteq A_i^*, i \in \mathbb{Z}_n\}$ are mutually observable then

$$\text{AP}(\|_{i=1}^n K_i, \|_{i=1}^n L_i, P) = \|_{i=1}^n \text{AP}(K_i, L_i, P_i^{loc}). \quad (23)$$

Conversely, if for fixed local plant languages $\{L_i, i \in \mathbb{Z}_n\}$ equation (23) holds true for any $\{K_i \subseteq L_i, i \in \mathbb{Z}_n\}$, then, for any $i \neq j$, $\{P_i(L), i \in \mathbb{Z}_n\}$ are observable with respect to L_i and $A_{o,i}$.

In view of our previous results it is straightforward to show that a necessary structural condition for local antipermissive control policy to equal global antipermissive control policy is that $\text{AP}(P_i(L), L_i, P_i^{loc}) = P_i(L)$, which means that for all $i \in \mathbb{Z}_n$, $P_i(L)$ is observable with respect to L_i and $A_{i,o}$.

The interest of this theorem should be clear: under the conditions that are stated it is possible to perform the antipermissive control synthesis with partial observations locally, which represents an exponential save on the computational complexity and makes in fact the antipermissive control synthesis of large distributed plants feasible. Recall that under very general conditions we have one inclusion meaning that global antipermissive control synthesis yields in general a larger language than the local antipermissive control synthesis.

5 Modular supervisory control with an indecomposable specification

In many engineering problems the specification is only defined globally and is not decomposable unlike the plant language. An example is the specification for a communication protocol of a wireless network. In this section the case of general specification languages that are neither necessarily decomposable nor contained in the global plant language is studied. Necessary and sufficient conditions are found with respect to which handling of the global plant is avoided for the computation of supremal controllable sublanguages of (global) indecomposable specification languages.

Consider a modular discrete-event system and assume that the local plants agree on the controllability of their common events. Denote the global plant and the specification languages by L and K , respectively. In our setting, L is decomposable into local plant languages: $L = L_1 \parallel \dots \parallel L_n$ (note that the L_i may have different alphabets). In most of the works on this topic K is similarly decomposable into local specification languages and $K \subseteq L$. The general case is when this condition is not satisfied and, moreover, K may not be included in L . This case has been studied in [2], where the assumption that all shared events are controllable is used. A condition on K called G -observability was needed for local synthesis of the supremal controllable sublanguage.

Instead of local specifications, languages $K_i := K \cap P_i^{-1}(L_i)$ are considered. These will play the role of local components of specification languages, although their alphabet is the global alphabet A . They can be considered as local over-approximations of $K \cap L$, because clearly $K \cap L = \bigcap_{i=1}^n K_i$.

Definition 13. Consider a modular DES. The modular plant languages $\{L_i, i \in \mathbb{Z}_n\}$ are called globally mutually controllable if

$$P_j^{-1}(L_j)(A_{ju}) \cap P_i^{-1}(L_i) \subseteq P_j^{-1}(L_j), \forall i, j \in \mathbb{Z}_n, i \neq j. \quad (24)$$

Proposition 2. Consider a modular DES. Global mutual controllability (GMC) is equivalent to the following property:

$$P_j^{-1}(L_j)A_u \cap P_i^{-1}(L_i) \subseteq P_j^{-1}(L_j), \forall i, j \in \mathbb{Z}_n, i \neq j. \quad (25)$$

Proof. Note that the only difference is that A_{ju} in GMC is replaced by A_u . Therefore the new property is clearly stronger than GMC. Maybe surprisingly the converse implication is satisfied as well. Let GMC hold true, $s \in P_j^{-1}(L_j)$, $u \in A_u$, and $su \in P_i^{-1}(L_i)$. Then we have two cases: either $u \in A_j$ or $u \notin A_j$. The former case entails that $u \in A_{ju}$ due to the shared event controllability status assumption. Using GMC we conclude $su \in P_j^{-1}(L_j)$. In the latter case we notice that $P_j(u) = \varepsilon$, i.e. $P_j(su) = P_j(s)$, which means that $P_j(su) \in L_j$, i.e. $s \in P_j^{-1}(L_j)$.

Definition 14. Consider a modular DES. The modular plant languages $\{L_i, i \in \mathbb{Z}_n\}$ are called modularly controllable if

$$(LA_u) \cap P_i^{-1}(L_i) \subseteq L, \forall i \in \mathbb{Z}_n. \quad (26)$$

Modular controllability is in general weaker than global mutual controllability and it will play the role of a necessary condition.

Proposition 3. *Consider a modular DES. Global mutual controllability (GMC) implies modular controllability.*

Proof. Let for any $i \neq j \in Z_n$:

$$P_j^{-1}(L_j)A_u \cap P_i^{-1}(L_i) \subseteq P_j^{-1}(L_j).$$

Since for $j = i$ the inclusion is trivially true, we have the inclusion for any $i, j \in Z_n$. Then, by taking intersection for j over Z_n we obtain:

$$\bigcap_{j=1}^n [P_j^{-1}(L_j)A_u \cap P_i^{-1}(L_i)] \subseteq \bigcap_{i=1}^n [P_j^{-1}(L_j).A_u]$$

Since $[\bigcap_{i=1}^n P_j^{-1}(L_j)]A_u \subseteq \bigcap_{i=1}^n [P_j^{-1}(L_j).A_u]$ we obtain finally, $LA_u \cap P_i^{-1}(L_i) \subseteq L$, i.e. modular controllability (MC).

The next theorem provides novel necessary and sufficient conditions for modular control synthesis to equal global control synthesis. In [9] it was shown only that global mutual controllability is a sufficient condition.

Theorem 7. Modular control synthesis equals global control synthesis for the supremal controllable sublanguage in case of complete observations and of indecomposable specifications. *Consider a modular discrete-event system. Assume that the local plants agree on the controllability of their common events.*

If the local plants $\{L_i, i \in Z_n\}$ are modularly controllable then

$$\sup C(K \cap L, L, A_u) = \bigcap_{i=1}^n \sup C(K_i, P_i^{-1}(L_i), A_u). \quad (27)$$

Conversely, if for a given modular plant equality (27) holds true for any global specification K then the local plant languages, $\{L_i, i \in Z_n\}$ are modularly controllable.

6 Distributed supervisory control with an indecomposable specification

In this section the case is studied of control of a distributed discrete-event system, thus for which one or more of the local plants has only partial observations, and of an indecomposable specification. First a structural condition called global mutual normality is introduced. It is similar to mutual normality in the case of decomposable specification ([3]), but it concerns $P_i^{-1}(L_i)$ instead of L_i .

Definition 15. *Consider a distributed DES. The modular plant languages $\{L_i \subseteq A_i^*, i \in Z_n\}$ are called globally mutually normal if*

$$(P^{-1}PP_j^{-1})(L_j) \cap P_i^{-1}L_i \subseteq P_j^{-1}L_j, \forall i, j \in Z_n, i \neq j. \quad (28)$$

Definition 16. Consider a distributed DES. Modular plant languages $\{L_i, i \in Z_n\}$ are called modularly normal if L is $(P_i^{-1}(L_i), P)$ -normal; or, equivalently,

$$P^{-1}P(L) \cap P_i^{-1}(L_i) \subseteq L, \forall i \in Z_n. \quad (29)$$

Modular normality is in general weaker than global mutual normality (GMN).

Proposition 4. Consider a distributed DES. Global mutual normality (GMN) implies modular normality (MN).

Proof. Let GMN holds true, or, equivalently,

$$(P^{-1}PP_j^{-1})(L_j) \cap P_i^{-1}L_i \subseteq P_j^{-1}L_j, \forall i, j \in Z_n, i \neq j.$$

Since for $i = j$ the inclusion becomes trivial, we may assume that the inclusion is satisfied for any $i, j \in Z_n$. We obtain: $P^{-1}PL \cap P_i^{-1}L_i = P^{-1}P(\bigcap_{i=1}^n P_j^{-1}L_j) \subseteq \bigcap_{j=1}^n (P^{-1}PP_j^{-1})(L_j) \cap P_i^{-1}L_i \subseteq \bigcap_{i=1}^n P_j^{-1}L_j = L$, where the last inclusion follows from intersecting both sides of the first inclusion (GMN) for j ranging over Z_n . Thus MN holds true.

Theorem 8. Modular control synthesis equals global control synthesis for the supremal normal sublanguage in case of a distributed DES and of an indecomposable specification. Consider a distributed DES. Assume that the local plants agree on the observability of their common events.

If the local plant languages $\{L_i, i \in Z_n\}$ are modularly normal then

$$\sup N(K \cap L, L, P) = \bigcap_{i=1}^n \sup N(K_i, P_i^{-1}(L_i), P). \quad (30)$$

Conversely, if for a given modular plant equality (30) holds true for any global specification $K \subseteq A^*$ then the local plant (partial) languages $\{L_i, i \in Z_n\}$ are modularly normal.

Now we present an example, where it is shown that global mutual normality (GMN) is not a necessary condition.

Example 2. Let $A = \{a, a_1, a_2, \tau, \tau_1, \tau_2\}$, $A_1 = \{a_1, \tau_1, a, \tau\}$, $A_2 = \{a_2, \tau_2, a, \tau\}$, $A_o = \{a_1, a_2, a\}$, $A_{o,1} = \{a_1, a\}$, and $A_{o,2} = \{a_2, a\}$. Consider the following local plant languages (the marked languages are not considered): Let $\text{prefix}(K) = \{\varepsilon, a, a_1, a_1a_2\}$. One can easily verify that K is not decomposable. Indeed, the inclusion $\text{prefix}(K) \subset P_1^{-1}P_1(\text{prefix}(K)) \cap P_2^{-1}P_2(\text{prefix}(K))$ is strict, i.e. $\text{prefix}(K) \neq P_1^{-1}P_1(\text{prefix}(K)) \cap P_2^{-1}P_2(\text{prefix}(K))$. Computing further the parallel product $L = L_1 \parallel L_2$ yields:

$$\sup N(K \cap L, L, P) = \{\varepsilon\}.$$

Note that in this example $K_i = K \cap P_i^{-1}L_i = K$ for $i = 1, 2$. It is also easy to see that $\sup N(K_i, P_i^{-1}(L_i), P) = \{\varepsilon\}$ as well for $i = 1, 2$, i.e. the commutativity holds trivially true.

On the other hand, global mutual normality does not hold. We have e.g.

$$\tau_1 \in (P^{-1}PP_1^{-1})(L_1) \cap (P_2)^{-1}(L_2) \setminus P_1^{-1}(L_1).$$

Modular computation of supremal controllable and normal sublanguages

Theorem 9. Modular control synthesis equals global control synthesis for the supremal controllable and normal sublanguage in case of a distributed DES and of an indecomposable specification. *Consider a distributed DES. Assume that the local plants (1) agree on the controllability of their common events and (2) agree on the observability of their common events.*

If the local plant languages $\{L_i, i \in Z_n\}$ are modularly controllable and modularly normal, then

$$\sup \text{CN}(K \cap L, L, P, A_u) = \bigcap_{i=1}^n \sup \text{CN}(K_i, P_i^{-1}(L_i), P, A_u). \quad (31)$$

Conversely, if for a given modular plant equality (31) holds true for any global specification K then local plant languages $\{L_i, i \in Z_n\}$ are modularly controllable and modularly normal.

Proof. The proof of sufficiency relies on a coinduction-like algorithm for computation of $\sup \text{CN}$ from [8] and the coinduction proof principle. The proof for necessity is the same as in the previous theorems.

Note that global mutual controllability together with global mutual normality imply modular controllability and modular normality, and the former notions are easier to verify than the latter ones, because they do not include the global plant. In fact, although modular controllability and modular normality can be checked in polynomial time (but in size of global DES!), their verification requires the construction of the global system, which contradicts the modular approach, because the size of global system grows exponentially with the number of components. On the other hand verification of global mutual controllability and global mutual normality is polynomial in the size of the local components.

Antipermissive control policy

In this subsection our intention is to generalize the results concerning modular computation of supremal normal sublanguages of global specifications to modular computation of closed-loop languages using the antipermissive control policy. This will be done in the very same way as was done in section 4 for local specification. Unlike the setting of section 4 we will work with $P_i^{-1}(L_i)$ instead of L_i . Since we work with the global alphabet, synchronous composition coincides with intersection. The concept of modular observability using Definition 2 is needed:

Definition 17. *Consider a distributed DES. The modular plant languages $\{L_i, i \in Z_n\}$ are called modularly observable if for any $i \in Z_n$ we have that L is observable with respect to $P_i^{-1}(L_i)$ and A_o .*

Theorem 10. Modular control synthesis equals global control synthesis for the closed-loop language in case of a distributed DES, of an indecomposable specification, and of an antipermissive control policy. *Consider a distributed discrete-event system. Assume that the local plants (1) agree on the controllability of their common events and (2) agree on the observability of their common events.*

If $\{L_i, i \in Z_n\}$ are modularly observable then

$$\text{AP}(K \cap L, L, P) = \bigcap_{i=1}^n \text{AP}(K_i, P_i^{-1}(L_i), P). \quad (32)$$

Conversely, if for a given modular plant equality (32) holds true for any global specification K then the local plant (partial) languages $\{L_i, i \in Z_n\}$ are modularly observable.

7 Conclusion

An overview has been presented of different methods for modular control of DES together with new methods and conditions. Most of the possible special cases of modular and distributed control have been covered (both fully and partially observed systems, both local and global specifications). Among the new results we provided for all presented cases necessary conditions for modular control synthesis to equal global control synthesis. Antipermissive control policy was proposed for the global specification and necessary and sufficient conditions have been presented for local antipermissive control policy to yield the same result as computationally much more efficient local antipermissive control policy.

These results are important for the optimal supervisory control of large distributed plants, because with respect to the derived conditions control synthesis can be implemented modularly. All the sufficient conditions we have presented are easier to check than their counterparts for global systems. The structural condition do not depend on a particular specification, which is very important for global (indecomposable) specifications.

Acknowledgements Partial financial support of the Grant GA AV No. B100190609 and of the Academy of Sciences of the Czech Republic, Institutional Research Plan No. AV0Z10190503 is gratefully acknowledged.

References

1. S.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, Dordrecht 1999.
2. B. Gaudin and H. Marchand. Modular Supervisory Control of a Class of Concurrent Discrete Event Systems. Proceedings *WODES'04*, Workshop on Discrete-Event Systems, pp. 181-186, Reims, September 22-24, 2004.
3. J. Komenda and J.H. van Schuppen. Supremal Normal Sublanguages of Large Distributed Discrete-Event Systems. Proceedings *WODES'04*, Workshop on Discrete-Event Systems, Reims, September 22-24, 2004.

4. J. Komenda. Modular Control of Large Distributed Discrete-Event Systems with Partial Observations. In Proceedings of the 15th International Conference on Systems Science, Vol. II, pp. 175-184, Wroclaw, Poland, September 2004.
5. J. Komenda and J. H. van Schuppen. Modular antipermissive control of discrete-event systems. Proceedings of IFAC World Congress 2005, Prague, July 2005.
6. J. Komenda and J. H. van Schuppen. Supremal Sublanguages of General Specification Languages Arising in Modular Control of Discrete-Event Systems. Proceedings of Joint 44th IEEE Conference on Decision and Control and European Control Conference, Sevilla, pp. 2775-2780, December 2005.
7. J. Komenda, J. H. van Schuppen. Modular control of discrete-event systems with coalgebra. Submitted, August 2005.
8. J. Komenda and J. H. van Schuppen. Control of Discrete-Event Systems with Partial Observations Using Coalgebra and Coinduction. Discrete Event Dynamical Systems: Theory and Applications 15(3), 257-315, 2005.
9. J. Komenda and J. H. van Schuppen. Optimal Solutions of Modular Supervisory Control Problems with Indecomposable Specification Languages. Proceedings International Workshop on Discrete Event Systems (WODES), 2006, to appear.
10. M. Lamouchi and J.G. Thistle. Effective Control Synthesis For Discrete Event Systems Under Partial Observations. In Proceedings of the IEEE Conference on Decision and Control, 2000.
11. F. Lin and W.M. Wonham, On Observability of Discrete-Event Systems, *Information Sciences*,44: 173-198, 1988.
12. R. Milner. Communication and Concurrency. *Prentice Hall International Series in Computer Science*. Prentice Hall International, New York, 1989.
13. A. Pnueli and R. Rosner. Distributed reactive systems are hard to synthesize, *Proceedings of the 1990 IEEE Symposium on the Foundations of Computer Science*, IEEE, New York, 1990, 746-757.
14. P.J. Ramadge and W.M. Wonham. The Control of Discrete-Event Systems. *Proc. IEEE*, 77:81-98, 1989.
15. J.J.M.M. Rutten. Universal Coalgebra: A Theory of Systems. *Theoretical Computer Science* 249(1):3-80, 2000.
16. S. Tripakis. Undecidable Problems of Decentralized Observation and Control. In Proceedings of the IEEE Conference on Decision and Control, 2001.
17. J.N. Tsitsiklis. On the Control of Discrete-Event Dynamical Systems, *Mathematics of Control, Signal, and Systems*, 95-107, 1989.
18. K.C. Wong and S. Lee. Structural Decentralized Control of Concurrent Discrete-Event Systems. *European Journal of Control*, 8:477-491, 2002.
19. W.M. Wonham. Lecture notes on control of discrete-event systems, University of Toronto, Department ECE, Toronto, 2005.
<http://www.control.toronto.edu/people/profs/wonham/wonham.html>
20. W.M. Wonham and P.J. Ramadge. On the Supremal Controllable Sublanguage of a Given Language, *SIAM J. Control Optim.*, 25:637-659, 1987.