Preprint, Institute of Mathematics, AS CR, Prague. 2007-12-18

Supervisory Control of Modular Systems with Global Specification Languages

J. Komenda [a] J.H. van Schuppen [b] B. Gaudin [c] H. Marchand [d]

[a] *Institute of Mathematics, Czech Academy of Sciences, Brno Branch, Zizkova 22, 616 62 Brno, Czech Republic*

[b] *CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands*

[c] *System Research Group, School of Computer Science and Informatics, UCD Belfield, Dublin 4, Ireland.*

[d] *IRISA/INRIA Rennes, Campus universitaire de Rennes 1, 35042 Rennes, France*

Abstract

The paper presents sufficient conditions for modular (supervisory) control synthesis to equal global control synthesis. In modular control synthesis a supervisory control is synthesized for each module separately and the supervisory control consists of the parallel composition of the modular supervisory controls. The general case of the specification that is indecomposable and not necessarily contained in the plant language, which is often the case in practice, is considered. The usual assumption that all shared events are controllable is relaxed by introducing two new structural conditions relying on the global mutual controllability condition. The novel concept used as a sufficient structural condition is strong global mutual controllability. The main result uses a weaker condition called global mutual controllability together with local consistency of the specification. An example illustrates the approach.

Key words: Modular control of discrete-event systems; Partial controllability; Global mutual controllability; Supremal controllable sublanguages

1 Introduction

In this paper supervisory control synthesis of modular discrete-event systems (DES; also called concurrent DES) is studied. DES represented by finite automata have been introduced by P.J. Ramadge and W.M. Wonham (see e.g. [11]). Large-scale modular DES are typically composed of a large number of relatively small (in size) local DES that run concurrently (in parallel). The global system is formed as a synchronous product of these local components with synchronization on shared actions.

The goal of the supervisory control is to impose a property on the system by running another automaton (supervisor) in parallel with the original plant so that the specification given by a specification language or its automaton is met. One of the key issues in supervisory control synthesis is the computation of the supremal controllable sublanguage of the specification language, from which the supervisor can be constructed. For modular DES it is very much welcome if the supremal controllable sublanguage can be computed without building the global plant since this computation may not be feasible because of the complexity of the system. In [12] and [3] quite a restrictive condition is imposed on events shared by several local alphabets: they must be controllable for all subsystems for modular (local) control synthesis to yield the same optimal solution as global control synthesis. This assumption has been generalized in [13] to the condition that the shared events must have the same control status for all subsystems that share a particular event. This rather general assumption together with general specification languages for the global plant (i.e. those that are not decomposable into local specification languages) are considered in this paper. The more difficult case of global (indecomposable) specification languages has been first considered in [3] under the conditions that all shared events are controllable. In this paper this condition is weakened to that used in [13] using a new concept of global mutual controllability, which can

Email addresses: komenda@ipm.cz (J. Komenda), J.H.van.Schuppen@cwi.nl (J.H. van Schuppen), benoit.gaudin@ucd.ie (B. Gaudin), herve.marchand@irisa.fr (H. Marchand).

be viewed as a natural counterpart of mutual controllability when dealing with general global specification languages.

In this paper we face moreover the problem of indecomposable specification languages, i.e. we do not have local specification languages. Nevertheless, as is shown in [3], it is still possible to exploit the modular structure of the plant and to avoid the manipulation with the global plant. On the other hand, the solution proposed in [3] relies on a structural condition, where all shared events are required to be controllable for all subsystems that share a particular event. We want to weaken this condition, while still preserving the possibility of "local" computation, i.e. without having to manipulate with the global plant language. Thus, the main scope of the paper is to generalize the results of [3] in two directions: 1) leave out the structural condition that all shared events are controllable and 2) leave out the restriction on the indecomposable specification by finding a new structural (involving only plant languages, i.e. specification independent) condition under which the methodology proposed in [3] can still be applied. Since it turns out that our new structural condition is too strong, we take finally a combination of a weaker structural condition together with a specification dependent condition used previously in [3]. A preliminary version of our results based on coalgebraic approach of [10] has appeared in [6]. Since dealing with global specification languages is in itself a difficult problem, our attention is restricted to modular control synthesis without blocking consideration as the blocking issue would pose even more difficulties.

In Section 2 basic notions and concepts of supervisory control theory are recalled. Section 3 is devoted to the presentation and motivation of the main problem studied in this paper. Section 4 is devoted to our main results. Two novel sufficient conditions (a specification dependent and a structural one) are presented under which optimal modular control synthesis with complete observations is possible without building the global plant and without a loss of global optimality. At the end of Section 4 an example is given to illustrate our main results. In conclusion we point out possible extensions of our work.

2 Notation and Problem Statement

This paper follows the standard notation of supervisory control, e.g. the book [2]. DES are modelled as deterministic generators that are finite automata with partial transition functions. A (deterministic) *generator* G over an event set A is $G = (Q, A, f, q_0)$, where Q is the *state set*, A is the *event set*, $f : Q \times A \to Q$ is the *partial transition function*, and $q_0 \in Q$ is the *initial state*. The marked states are not considered in this paper. If a transition is defined then this is denoted by $f(q, a)!$ The transition function f can be extended to $f : Q \times A^* \to Q$ by induction in the standard way. The behaviors of DES generators are defined in terms of languages (subsets of the free monoid of the words A^*). The prefix closure $\mathrm{prefix}(L)$ of a language L is the set of all prefixes of all its words. A language $L \subseteq A^*$ is said to be prefix-closed if $L = \mathrm{prefix}(L)$. Recall that the *language* of the generator is $L(G) = \{s \in A^* | f(q_0, s)!\}$.

A *controlled generator* is a structure (G, A_c, Γ_c), where G is a generator, $A_c \subseteq A$ is the subset of *controllable events*, $A_u = A \backslash A_c$ is the subset of *uncontrollable events*, and $\Gamma_c = \{\gamma \subseteq A | A_u \subseteq \gamma\}$, is called the *set of control patterns*. A *control supervisor* for the controlled generator is a map $g : L(G) \to \Gamma_c$. The *closed-loop system* associated with a controlled generator and a supervisor as denoted above is defined as the smallest language $L(S/G) \subseteq A^*$ which satisfies

(1) $\epsilon \in L(S/G)$,
(2) if $s \in L(S/G)$, $sa \in L(G)$ and $a \in g(s)$ then $sa \in L(S/G)$.

Note that at the automata level the supervision is implemented by a parallel composition of the plant and the supervisor. We only need to ensure that a supervisor never disables uncontrollable events, which can be made by adding selfloops of uncontrollable events around all states of the supervisor. Otherwise stated, the active event set of any state s of a supervisor (that corresponds exactly to the control patern $g(s)$ above) always contains all uncontrollable events.

It is very well known that not every specification language can be achieved by a supervisory controller [11]. Recall the basic definition of controllability that is needed for a specification language K to be exactly achieved by a supervisory controller.

Definition 2.1 (Controllability) *Let L be a prefix-closed language over an alphabet A and let $A_u \subseteq A$. A (specification) language $K \subseteq A^*$ is said to be controllable with respect to L and A_u, if* $\mathrm{prefix}(K)A_u \cap L \subseteq \mathrm{prefix}(K)$.

Since the blocking issue is not addressed in this paper, only prefix-closed languages are considered in the paper. This is a standard assumption if only safety issues are studied, which are most often the issues under consideration, i.e. the controlled behavior must be included in the specification language. The goal is to find a supervisor V such that $L(V/G) = K$. Such a supervisor exists if and only if the specification language K is controllable. This is why for specifications that are not controllable supremal controllable sublanguages of K are considered. The notation $\sup \mathrm{C}(\mathrm{K}, \mathrm{L}, \mathrm{A_u})$ is chosen for supremal controllable sublanguage of K with respect to L and A_u. This language always exists and corresponds to the union of all controllable sublanguages (see e.g [2]).

Modular DES (also called concurrent DES) are DES with the particular structure of the global plant that

is formed by the synchronous product of local subsystems. For this purpose, for any integer $n \in \mathbb{N}$, $\mathrm{Z_n} = \{1, 2, \ldots, n\}$ denotes the set of the first n natural numbers.

Definition 2.2 *A modular discrete-event system with $n \in \mathbb{Z}$ modules is defined as the synchronous product $G = \|_{i=1}^{n} G_i$ of the local generators G_i, where $\|$ denotes the classical parallel composition (see e.g [2] for more details).*

A denotes $\cup_{i=1}^{n} A_i$ and P_i denotes the natural projection from A to A_i: $P_i : A^* \to A_i^*$. The concept of inverse projection $P_i^{-1} : \mathrm{Pwr}(A_i^*) \to \mathrm{Pwr}(A^*)$ is also used. If for all $i \in \mathrm{Z_n}$, L_i represents a language of G_i, then the parallel composition of these languages is defined by

$$\|_i \, L_i = \bigcap_i P_i^{-1}(L_i). \tag{1}$$

It is known (e.g. [2]) that $L(G_1 \| G_2) = L(G_1) || L(G_2)$.

3 Modular Systems

In this section the concurrent behavior of local subplants $\{G_i, i \in \mathrm{Z_n}\}$ is considered. Consider the local alphabets of these sub-plants, $\{A_i, i \in \mathrm{Z_n}\}$, which are not necessarily pairwise disjoint. For all $i \in \mathrm{Z_n}$, A_i is the union of local controllable and local uncontrollable set of events, $A_i = A_{iu} \cup A_{ic}$.

For such modular systems there is an important structural condition that concerns the controllability status of events shared among the local sub-plants. It requires that the events shared by two or more local sub-plants must have the same controllability status in all these sub-plants. The following assumption stems from [13], where it was originally introduced.

The local sub-plants $\{G_i, i \in \mathrm{Z_n}\}$ *agree on the controllability of their shared events* if

$$A_{iu} \cap A_j = A_i \cap A_{ju}, \ \forall i, j \in \mathrm{Z_n}, \ i \neq j. \tag{2}$$

In the following it will be assumed that the local subplants satisfy (2). The set of controllable events of the global plant is denoted $A_c = \cup_{i=1}^{n} A_{ic}$. Equation (2) implies that for all $i \in \mathrm{Z_n}$, $A_{ic} = A_c \cap A_i$. Also, if we denote $A = \cup_{i=1}^{n} A_i$ and $A_u = \cup_{i=1}^{n} A_{iu}$ then we still have the disjoint union $A = A_c \cup A_u$.

The concept of decomposability is defined using the synchronous product of languages:

Definition 3.1 ([12]) *$L \subseteq A^*$ is* decomposable *with respect to $\{P_i, \ i \in \mathrm{Z_n}\}$ if there exist $\{L_i \subseteq A_i^*, \ i \in \mathrm{Z_n}\}$ such that*

$$L = \|_{i=1}^{n} \, L_i = \bigcap_{i=1}^{n} P_i^{-1}(L_i) \tag{3}$$

Since we consider modular plants, the global plant language L is decomposable into local plant languages: $L = \|_{i=1}^{n} \, L_i$.

In modular supervisory control of DES the concurrent behavior of the local sub-plants (finite automata) $G_1, \ldots, G_n$ is considered. Although the computational complexity of the algorithm for supremal controllable sublanguages is satisfactory (unlike several other problems of supervisory control), in the case of large modular DES there is an exponential blow up of the computational complexity in terms of the number of local components. In order to cope with this issue, the modular structure should be exploited [13].

4 Supremal Controllable Sublanguages of General Specification Languages.

The global plant and the specification languages are respectively denoted by L and K. Unlike the global specification language K, the global plant language L is not given explicitly, but only through local plant languages: $L = L_1 \| \cdots \| L_n = \cap_{i=1}^{n} P_i^{-1}(L_i)$ (note that the L_i may have different alphabets). In most of the publications 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. This case has been studied in [3], where the assumption that all shared events are controllable is used. It was shown that if K is a subset of L, then local computations of the supremal controllable sublanguage could be performed.

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 viewed as local over-approximations of $K \cap L$, because clearly $K \cap L = \cap_{i=1}^{n} K_i$. However, it turns out that it is not in most cases possible to compute simply the supremal controllable sublanguages of K_i over $P_i^{-1}(L_i)$ and then take their intersection to obtain the most permissive supervisor over the whole system. Instead, another approach is proposed in [3] using the newly introduced concept of partial controllability.

Definition 4.1 (partial controllability) *A language $K' \subseteq K \subseteq L$ is said to be partially controllable with respect to A', A (with $A' \subseteq A$), K, and L, if*

(i) *K' is controllable with respect to A' and L;*
(ii) *K' is controllable with respect to A and K.*

It is shown in [3] that the supremal partially controllable sublanguage of K with respect to A', A (with $A' \subseteq A$), K, and L, denoted by $\sup \mathrm{PC}(K, A', A, L)$, exists. According to [3] we know that $\sup \mathrm{PC}(K, A', A, L)$ always exists and can be computed from the following formula:

$$\sup \mathrm{PC}(K, A', A, L) = \sup \mathrm{C}(\sup \mathrm{C}(K, A', L), A, K)$$

The concept, called $G-$controllability plays an important role. We recall here that if L is a language over an alphabet A and $s \in A^*$, then $(L)_s$ denotes the set of suffixes t of s such that $st \in L$.

Definition 4.2 ($G-$controllability) *A specification language K is said to be $G-$controllable if $\forall i \in \mathrm{Z_n}$ and $\forall s \in K_i = K \cap P_i^{-1}(L_i)$, $(K_i)_s \cap A_u^*$ is controllable with respect to $(P_i^{-1}(L_i))_s \cap A_u^*$ and A_{iu}.*

The notion of $G-$controllability [1] is important for modular verification of controllability of global specification languages. Indeed, if a global specification language turns out to be controllable, the whole procedure for synthesizing supremal controllable sublanguages that we present in the sequel can of course be avoided. $G-$controllability is easy to verify, because it is based on controllability of local languages. Since $G-$controllability implies that all K_i, $i \in \mathrm{Z_n}$ are partially controllable with respect to A_{iu}, A_u, $P_i^{-1}(L_i)$ and K_i, which in turn implies that $K \cap L$ is controllable with respect to L and A_u, the following Proposition holds true.

Proposition 4.1 *Assume that K is $G-$controllable and assume that the local plants agree on the controllability of their shared events. Then $K \cap L$ is controllable with respect to L and A_u, i.e. $\sup \mathrm{C}(K \cap L, L, A_u) = K \cap L$.*

This result offers an interesting and low complexity test for controllability: in order to verify controllability of $K \cap L$, it is sufficient to check $G-$controllability, which can be done without building the global plant, but using local sub-plants only. In the rest of this paper constructive conditions are provided. We need the following inductive characterization of supremal controllable sublanguages based on a formula from [1].

Proposition 4.2 *For any (prefix-closed) language $K \subseteq A^*$ with $K \subseteq L$, the supremal controllable sublanguage within K, denoted by $\sup \mathrm{C}(K, L, A_u)$, is characterized by the following two conditions:*

(i) *$\varepsilon \in \sup \mathrm{C}(K, L, A_u)$*
(ii) *For $s \in \sup \mathrm{C}(K, L, A_u)$ and $a \in A$ we have $sa \in \sup \mathrm{C}(K, L, A_u)$ iff*

$$sa \in K \cap L \text{ and } \forall u \in A_u^* : \ sau \in L \ \Rightarrow \ sau \in K$$

[1] Note that the condition of $G-$controllability is stronger than the condition of $G-$observability introduced in [3].

In this paper general specification languages are dealt with. As mentioned above, the case where $K \not\subseteq L$ is of great interest. In such a case, the specification to ensure by means of control is actually $K \cap L$. Because of the complexity blow up associated to modular systems, the computation of a recognizer for this language has to be avoided.

We can use a structural condition similar to the mutual controllability of [13]. Following the same idea as in the case of decomposable specification [5] we introduce the following concept:

Definition 4.3 *Local plant languages $\{L_i, i \in \mathbb{Z}_n\}$ are called* strongly globally mutually controllable *if $P_j^{-1}(L_j)A_u \cap P_i^{-1}(L_i) \subseteq P_j^{-1}(L_j)$, $\forall i, j \in \mathrm{Z_n}$, $i \neq j$.*

Although the condition concerns languages $P_i^{-1}(L_i)$ over the global alphabet, these are easily derived from the local plant languages L_i and we still avoid building the representation of the whole plant. We notice that the recognizers of $P_i^{-1}(L_i)$ are easily obtained from the recognizers of L_i by simply adding to all states the self-loops of events that are not in A_i. Unlike a typical situation in supervisory control, strong global mutual controllability is a symmetric notion of controllability, where it is not required that one language is a sublanguage of the other. Sufficient structural conditions for modular control synthesis to equal global control synthesis in the case of complete observations and of indecomposable specifications are formulated below.

Theorem 4.3 *Assume that $\{L_i, \ i \in \mathrm{Z_n}\}$ are strongly globally mutually controllable and the local plants agree on the controllability of their shared events. Then*

$$\sup \mathrm{C}(K \cap L, L, A_u) = \bigcap_{i=1}^{n} \sup \mathrm{PC}(K_i, A_{iu}, A_u, P_i^{-1}(L_i)). \tag{4}$$

Proof The proof by structural induction relies on the inductive characterization of supremal controllable sublanguages in Proposition 4.2. Since from the definitions of the $\sup \mathrm{C}$ and the $\sup \mathrm{PC}$ operators it follows that ε is in both sides of equality (4), the base step of the structural induction is obvious.
(i) For the "$\supseteq$" inclusion, see [3], where the assumption that all shared events are controllable is not needed for this inclusion.
(ii) Now show the inclusion "$\subseteq$". Assume that for $s \in A^*$ we have $s \in \sup \mathrm{C}(K \cap L, L, A_u) \ \Rightarrow$
$s \in \bigcap_{i=1}^{n} \sup \mathrm{PC}(K_i, A_{iu}, A_u, P_i^{-1}(L_i))$. We show that the implication remains true for sa with arbitrary $a \in A$. If for $a \in A$ we have $sa \in \sup \mathrm{C}(K \cap L, L, A_u)$ then it follows from Proposition 4.2 (with K replaced by $K \cap L$) that $sa \in K \cap L$ and $\forall u \in A_u^* : \ sau \in L \ \Rightarrow \ sau \in K \cap L$. We need to show that

$sa \in \cap_{i=1}^{n} \sup \mathrm{C}(\sup \mathrm{C}(K_i, A_{iu}, P_i^{-1}(L_i)), A_u, K_i)$, i.e. $\forall i \in \mathbb{Z}_n : sa \in \sup \mathrm{C}(\sup \mathrm{C}(K_i, A_{iu}, P_i^{-1}(L_i)), A_u, K_i)$.

According to inductive characterization of the outer supremal controllable sublanguage (cf. Proposition 4.2), we need to show that $\forall i \in \mathbb{Z}_n : sa \in K_i$, $sa \in \sup \mathrm{C}(K_i, A_{iu}, P_i^{-1}(L_i))$ and $\forall u \in A_u^* : sau \in K_i \Rightarrow sau \in \sup \mathrm{C}(K_i, A_{iu}, P_i^{-1}(L_i))$. According to Proposition 4.2 applied to the inner supremal controllable sublanguage this amounts to show also that $sau \in K_i$ implies $sau \in P_i^{-1}(L_i)$ and $\forall v \in A_{iu}^* : sauv \in P_i^{-1}(L_i) \Rightarrow sauv \in K_i$.

The first claim is obvious from $(K \cap L) = \bigcap_{i=1}^{n} K_i$: $sa \in K \cap L$ implies that $\forall i \in \mathbb{Z}_n : sa \in K_i$. The second claim is more difficult to show. Note that $sa \in \sup \mathrm{C}(K_i, A_{iu}, P_i^{-1}(L_i))$ is a special case of the claim $\forall u \in A_u^* : sau \in (K_i) \Rightarrow sau \in \sup \mathrm{C}(K_i, A_{iu}, P_i^{-1}(L_i))$ for $u = \varepsilon$.

Let $sau \in K_i$ for a $u \in A_u^*$. We need to show that $sau \in \sup \mathrm{C}(K_i, A_{iu}, P_i^{-1}(L_i))$, which according to Proposition 4.2 applied to the inner supremal controllable sublanguage means that $sau \in P_i^{-1}(L_i)$, $sau \in K_i$ (trivially satisfied) and $\forall v \in A_{iu}^* : sauv \in P_i^{-1}(L_i) \Rightarrow sauv \in K_i$. First of all, $sau \in P_i^{-1}(L_i)$ is obvious from $sau \in K_i$, because $\forall i : K_i \subseteq P_i^{-1}(L_i)$. Let us show that $\forall u \in A_u^*$ and $\forall v \in A_{iu}^* : sauv \in P_i^{-1}(L_i) \Rightarrow sauv \in K_i$. For this implication strong global mutual controllability is used. Let $sauv \in P_i^{-1}(L_i)$ for some $u \in A_u^*$ and $v \in A_{iu}^*$. Since $uv \in A_u^*$ we obtain that $uv = v_1 \dots v_k$ for some $k \in \mathbb{N}$, where $v_i \in A_u$, $i \in \mathbb{Z}_k$. Now we proceed by induction along the string v. According to strong global mutual controllability we obtain $sav_1 \in P_j^{-1}(L_j)(A_u) \cap P_i^{-1}(L_i) \subseteq P_j^{-1}(L_j)$. Thus, $sav_1 \in L = \cap_{i=1}^{n} P_i^{-1}(L_i)$. Similar argument is made for any v_l, $l \in \mathbb{Z}_k$. Thus, we obtain after an inductive application of the same argument that $sauv = av_1 \dots v_k \in L$. Recall that $uv \in A_u^*$. A direct application of the assumption that $sa \in \sup \mathrm{C}(K \cap L, L, A_u)$ now yields $sauv \in K \cap L$, which means that $sauv \in K \cap L = \cap_{i=1}^{n} K \cap P_i^{-1}(L_i)$, i.e. $\forall i \in \mathbb{Z}_n$: we have $sauv \in K_i = K \cap P_i^{-1}(L_i)$, which was to be shown. □

The last theorem provides a structural condition under which supremal controllable sublanguages can be computed without having to build the global plant. Moreover as a structural condition, it does not depend on a particular specification, which is very important for indecomposable specifications. Interestingly we notice that strong global mutual controllability is equivalent to a seemingly weaker condition, where A_u is replaced by local uncontrollable events A_{ju}.

Proposition 4.4 *Assume that the local plants agree on the controllability of their shared events, then strong global mutual controllability is equivalent to the following property:*

$$P_j^{-1}(L_j)A_{ju} \cap P_i^{-1}(L_i) \subseteq P_j^{-1}(L_j), \ \forall i, j \in \mathrm{Z_n}, \ i \neq j. \tag{5}$$

Proof Note that the only difference is that A_u in strong global mutual controllability (SGMC) is replaced by A_{ju}. Therefore the new property is clearly weaker than SGMC. Surprisingly the converse implication is satisfied as well. Let condition (5) holds true. We show that SGMC holds true as well. $s \in P_j^{-1}(L_j)$, $u \in A_u$, 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 shared event controllability status assumption. Using (5) we conclude $su \in P_j^{-1}(L_j)$. In the latter case we notice that $P_j(u) = \varepsilon$, i.e. $P_j(su) = P_j(s)$, hence $P_j(su) \in L_j$ and $s \in P_j^{-1}(L_j)$. Thus SGMC holds. □

In spite of Proposition 4.4, the condition of strong global mutual controllability is restrictive. This condition indeed implies that whatever the sequence s of $L_j \parallel L_i$, any event of A_{ju}, which is not shared by A_i, must be eligible after s. This is due to the fact that $P_i^{-1}(L_i)$ always allows such uncontrollable events from being triggered. In the sequel, we thus adopt a weaker condition than strong global mutual controllability. Nevertheless, it turns out that we need moreover a specification dependent condition from [3]. We obtain a combined, more elaborate, approach and use the condition of local consistency together with the much weaker notion of global mutual controllability similar to mutual controllability [13]. Very recently it was shown in [7] that global mutual controllability is equivalent to to mutual controllability of [13]. However the formulation below simplifies its usage in the proof of Theorem 4.5.

Definition 4.4 (Global mutual controllability) *Local plant languages* $\{L_i, \ i \in \mathrm{Z_n}\}$ *are called* globally mutually controllable *if* $\forall i, j \in \mathrm{Z_n}, \ i \neq j$

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

Without any structural condition on the system, a condition called local consistency is required in [3] to ensure a modular computation of the supervisor. It should not be surprising that the same condition appears in our setting.

Definition 4.5 (Local consistency) *A global specification K is said to be locally consistent with respect to A_u and L_i, $i \in \mathbb{Z}_n$ if for any $i \in \mathbb{Z}_n$ we have: $\forall s \in K_i$ and $\forall u \in A_u^*$ such that $su \in K_i$ and $\forall v \in A_{iu}^*$: $sP_i(u)v \in K_i$ and $suv \in P_i^{-1}(L_i) \Rightarrow suv \in K_i$.*

We show that the only condition that is moreover required to make our method work in the presence of shared uncontrollable events is the quite weak structural condition of mutual controllability. The main result of this paper is given by Theorem 4.5. It provides sufficient conditions for modular control synthesis to equal global control synthesis in the case of complete observations and indecomposable specifications.

Theorem 4.5 *Assume that K is locally consistent with respect to A_u and L_i, $i \in \mathbb{Z}_n$; $\{L_i,\ i \in \mathbb{Z}_n\}$ are globally mutually controllable; and the local plants agree on the controllability of their shared events. Then*

$$\sup \mathrm{C}(K \cap L, L, A_u) = \bigcap_{i=1}^{n} \sup \mathrm{PC}(K_i, A_{iu}, A_u, P_i^{-1}(L_i)). \tag{6}$$

Proof For the "$\supseteq$" inclusion, see [3]. It remains to show the opposite inclusion. Let $sa \in \sup \mathrm{C}(K \cap L, L, A_u)$ for $a \in A$. It follows from the inductive characterization of the supremal controllable sublanguage that $sa \in K \cap L$, $sa \in L$, and $\forall u \in A_u^* : sau \in L \Rightarrow sau \in K \cap L$. We need to show that $sa \in \bigcap_{i=1}^{n} \sup \mathrm{PC}(K_i, A_{iu}, A_u, P_i^{-1}L_i)$, i.e. that $\forall i \in \mathbb{Z}_n : sa \in \sup \mathrm{PC}(K_i, A_{iu}, A_u, P_i^{-1}L_i)$. The proof follows the same lines as (ii) of the proof of the theorem 4.3. Therefore we only need to show the part, where strong global mutual controllability is used. Let us show that $\forall u \in A_u^*$ and $\forall v \in A_{iu}^* : sauv \in P_i^{-1}(L_i) \Rightarrow sauv \in K_i$. For this implication global mutual controllability and local consistency are used (instead of strong global mutual controllability).

Let $sauv \in P_i^{-1}(L_i)$ for some $u \in A_u^*$ and $v \in A_{iu}^*$. Then $saP_i(u)v \in P_i^{-1}(L_i)$ as well, because $P_i(saP_i(u)v) = P_i(sauv)$. Since $P_i(u)v \in A_{iu}^*$ we obtain after excluding the trivial case $P_i(u)v = \varepsilon$ that $P_i(u)v = v_1 \dots v_k$ for some $k \in \mathbb{N}$, where $v_i \in A_{iu},\ i \in \mathbb{Z}_k$. Now we proceed by induction along the string $P_i(u)v$. We know that $P_i(u)v \in A_{iu}^*$. For any $j \in \mathbb{Z}_n :\ j \neq i$ we have according to our assumption that $A_{iu} \cap A_j = A_i \cap A_{ju}$ two possibilities: either $v_1 \in A_{ju}$ and then $v_1 \in A_{iu} \cap A_j$ or $v_1 \notin A_j$. The case $v_1 \notin A_j$ is easy, because then $P_j(v_1) = \varepsilon$, i.e. $P_j(sav_1) = P_j(sa)$. Since $sa \in L$ we have also $sa \in P_j^{-1}(L_j)$, thus $sav_1 \in P_j^{-1}(L_j)$ as well.

If $v_1 \in A_{iu} \cap A_j$, then according to global mutual controllability we obtain $sav_1 \in P_j^{-1}(L_j)(A_{ju} \cap A_i) \cap P_i^{-1}(L_i) \subseteq P_j^{-1}(L_j)$. Thus, in both cases $sav_1 \in L = \cap_{i=1}^{n} P_i^{-1}(L_i)$. Similar construction is made for any $v_m,\ m \in \mathbb{Z}_k$, where cases $v_m \in A_{ju}$ and $v_m \notin A_j$ are distinguished. Thus, we obtain after an inductive application of the same argument along the string $P_i(u)v \in A_{iu}^*$ that $saP_i(u)v = sav_1 \dots v_k \in L$. Notice that in particular $P_i(u)v \in A_u^*$. A direct application of the assumption that $sa \in \sup \mathrm{C}(K \cap L, L, A_u)$ now yields $saP_i(u)v \in K \cap L = \cap_{i=1}^{n} K \cap P_i^{-1}(L_i)$, i.e. $saP_i(u)v \in K_i = K \cap P_i^{-1}(L_i)$. Let us recall that $sauv \in P_i^{-1}(L_i)$. This implies using local consistency with sa playing the role of s in definition 4.5 that $sauv \in K_i$. Therefore $\forall i \in \mathbb{Z}_n$: we have $sauv \in K_i$, which was to be shown. □

Note that the sufficient conditions of Theorems 4.3 and 4.5 are not comparable in general, only global mutual controllability is clearly weaker than strong global mutual controllability. Finally we mention that all sufficient conditions we have presented are checkable in polynomial time and moreover in size of local systems and the specification. We now present an example illustrating Theorem 4.5.

Example 1 *In this example, the system G under consideration is a modular DES which consists of two subsystems G_1 and G_2. G_1 and G_2 are represented in Figure 1. Moreover, for $i = 1, 2$, $\Sigma_i = \{a_i, a_i', b_i, c, d, s_1, s_2, s_3, u, u', u_i\}$ represents the alphabet of G_i, with $\Sigma_{i,u} = \{u, u', u_i\}$. The set of shared events corresponds to $\Sigma_s = \{c, d, s_1, s_2, s_3, u, u'\}$.*

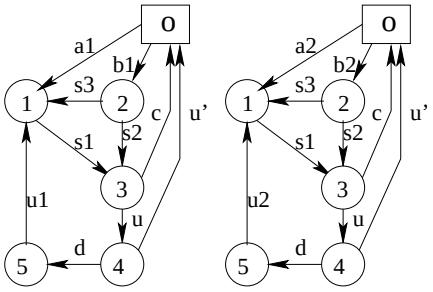

Fig. 1. Subsystems G_1 and G_2

Subsystems G_1 and G_2 interact as follows. For $i = 1, 2$, there is an initialization phase which makes G_1 (resp. G_2) evolve from its initial state triggering a_1 or b_1 (resp. a_2 or a_2' or b_2). Once this initialization phase is done, the subsystems communicate by means of synchronization events $(s_j)_{1 \leq j \leq 3}$, but system G_1 can evolve independently from state 1 to state 2 by occurrence of a_1'. Then both subsystems reach their state number 3, in which event c can occur so that each G_i enters into its initial state again. But it may happen that an uncontrollable event u, triggered by the environment, occurs at state 3. Each subsystem enters then into its state 4. In this state, either the subsystems are allowed to synchronize with each other triggering event d, or event u' occurs so that each G_i enters into its initial state again. If d occurs, then each G_i waits for u_i to be triggered. Since u_i is the only event which permits G to evolve from state 5, it is assumed to be uncontrollable. This ensures indeed that state 5 cannot be a deadlock state due to the action of a supervisor.

In this example, the specification language K, is given

in Figure 2[2] *. This specification aims at ensuring several properties. The first one is to avoid a blocking due to the initialization phases of each subsystem. Let us indeed imagine that from the initial states, b_1 and a_2 are triggered. In this case, G_1 and G_2 respectively enter into states 2 and 1. The global system is then in a deadlock state. Control specification K ensures that if b_1 (resp. a_2 or a'_2) is triggered then neither a_2 nor a'_2 (resp. b_1) can be triggered. Although there would not be any blocking issue, the control specification also ensures that if a_1 (resp. b_2) is triggered then b_1 (resp. a_1) can not be triggered. Moreover, K ensures two other properties: not only u must not be triggered if s_2 occurs, but also u_1 and u_2 must not occur if s_1 is triggered.*

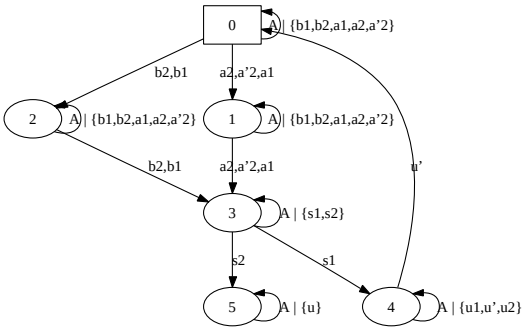

Fig. 2. Control specification K

The $\sup \mathrm{PC}$ *function, as well as an algorithm to check for Global Mutual Controllability and Local Consistency were implemented into a prototype called SynTool. The system under consideration here is actually Globally Mutually Controllable, and the control objective is Locally Consistent according to G and Σ_u. The model of* $\sup \mathrm{PC}(K_1, \Sigma_{1,u}, \Sigma_1, P_1^{-1}(G_1))$ *is given in Figure 3 (the one of* $\sup \mathrm{PC}(K_2, \Sigma_{2,u}, \Sigma_2, P_2^{-1}(G_2))$ *as a similar structure and is not represented). According to theorem 4.7, the synchronous execution of these two systems with G exactly achieves the most permissive controllable behavior included into K.*

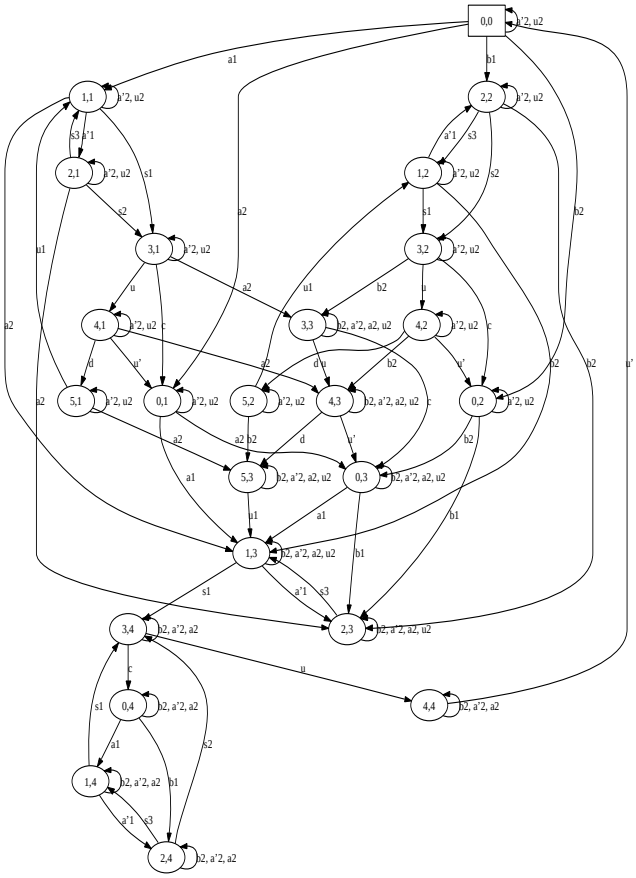

Fig. 3. $\sup \mathrm{PC}(K_1, \Sigma_{1,u}, \Sigma_1, P_1^{-1}(G_1))$

5 Conclusion

New methods for modular computation of supremal controllable sublanguages of indecomposable specification languages have been presented. The principle of the proofs presented here is much simpler than the one introduced in [6]. All the sufficient conditions we have presented can be checked in polynomial time in the number of states of each subsystem and the number of subsystems. Because each subsystem is usually exponentially smaller than the global system, so is the complexity of our approach compared to the usual one. The structural condition of strong global mutual controllability has been introduced and no conditions are required on the specification for a modular approach to be effective. However, this condition is quite restrictive, and our main result provides a weaker version, called global mutual controllability. Thanks to this condition a modular approach can still be applied, provided that the specification fulfills some requirements.

A natural continuation of this paper would be to find a partial controllability counterpart for partially observed discrete-event systems. It is not yet clear at this point what the right concept to be called partial normality should be. However, in any case it seems promising to investigate an iteration process which can achieve supremal controllable and normal sublanguage by alterating supremal partially controllable sublanguages and supremal normal sublanguages built only using local plant languages. Note that since the control synthesis of partially observed DES suffers from exponential worst-case complexity it is all the more important to find modular methods that can make supervisory control of large scale and partially observed DES at least in some cases feasible.

[2] Note that on this figure, A actually denotes $\Sigma_1 \cup \Sigma_2$ and that | denotes the set difference operator $\setminus$.

References

[1] R. D. Brandt, V. K. Garg, R. Kumar, F. Lin, S. I. Marcus and W. M. Wonham, Formulas for Calculating Supremal and Normal Sublanguages, *Systems and Control Letters*, 15(8)–111–117. 1990.

[2] S.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, Dordrecht 1999.

[3] 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.

[4] 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.

[5] J. Komenda and J.H. van Schuppen. Supremal Normal Sublanguages of Large Distributed Discrete-Event Systems. Workshop on Discrete-Event Systems, pp. 73-78, Reims, September 22-24, 2004.

[6] J. Komenda, J. H. van Schuppen, B. Gaudin, H. Marchand. Modular Supervisory Control with General Indecomposable Specification Languages. Proceedings of Joint 44th IEEE Conference on Decision and Control and European Control Conference, Sevilla, 12-15.12, 2005.

[7] J. Komenda and J.H. van Schuppen. Structural control of concurrent DES. To appear in Proceedings of MSR 2007 (Modélisation des Systèmes Réactifs), ENS Lyon, Editions Hermes, Paris, October 2007.

[8] F. Lin and W.M. Wonham. Decentralized Supervisory Control of Discrete-Event Systems, *Information Sciences* 44:199-224, 1988.

[9] K. Rohloff and S. Lafortune. On the Computational Complexity of the Verification of Modular Discrete-Event Systems. In *Proc. 41 st IEEE Conference on Decision and Control, Las Vegas, Nevada, USA, December 2002.*

[10] J.J.M.M. Rutten. Coalgebra, Concurrency, and Control. *Research Report CWI*, SEN-R9921, Amsterdam, November 1999.

[11] P.J. Ramadge and W.M. Wonham. The Control of Discrete-Event Systems. *Proc. IEEE*, 77:81-98, 1989.

[12] Y. Willner and M. Heymann. Supervisory Control of Concurrent Discrete-Event Systems. *International Journal of Control,* 54:1143-1166, 1991.

[13] K.C. Wong and S. Lee. Structural Decentralized Control of Concurrent Discrete-Event Systems. *European Journal of Control*, 8:477-491, 2002.

[14] K.C. Wong and W.M. Wonham. Modular Control and Coordination of Discrete-Event Systems. *Discrete Event Dynamical Systems: Theory and Applications*, 8:247-297, 1998.

[15] W.M. Wonham and P.J. Ramadge. Modular Supervisory Control of Discrete-Event Processes, *Mathematics of Control, Signal and Systems*, 1:13-30, 1988.

Jan Komenda received a M.Sc. degree in mathematics from the Masaryk University (Brno), Czech Republic, in 1994, and was awarded a Ph.D. degree in control and computer science from Université de Franche-Comté (Besançon), France in 1999. He worked at CWI Amsterdam (The Netherlands) in 2001-2003 in the group of J.H. van Schuppen. He is researcher at the Institute of Mathematics, Czech Academy of Sciences, branch Brno, Czech Republic. His research interests are in supervisory control of logical and timed discrete event systems using methods of coalgebra and idempotent algebra.

Jan van Schuppen is affiliated with the research institute Centrum voor Wiskunde en Informatica (CWI) in Amsterdam, The Netherlands and, in a part-time affiliation for one day a week, with the Department of Mathematics of the Vrije Universiteit in Amsterdam. He has studied at the Department of Applied Physics of Delft University of Technology and was awarded a Ph.D. diploma by the Department of Electrical Engineering and Computer Science of the University of California at Berkeley, CA, U.S.A. in 1973 where his research advisor was Pravin Varaiya. Van Schuppen's research interests include control of discrete-event systems and of hybrid systems, stochastic control, realization, and system identification. In discrete-event systems his interests are primarily: system theory, the use of coalgebra, decentralized control, and control and diagnosis of asynchronous systems. In applied research his interests include engineering problems of control of motorway traffic, of communication networks, and modeling, identification, and control in the life sciences. He is Editor-in-Chief of the journal Mathematics of Control, Signals, and Systems, and was Department Editor of the journal Discrete Event Dynamic Systems.

Benoît Gaudin graduated in mathematics in 1999 and received a Ph.D degree in computer science in 2004, both from the University Rennes 1. From October 2004 to August 2005, he held a research and teaching position in the University of Rennes. From September 2005 to May 2006, he was a postdoctoral researcher in the FOKUS Fraunhofer institute of Berlin. Since June 2006, Dr Gaudin has been a postdoctoral researcher in University College Dublin. His research interests focus on the control of Discrete Event Systems and graph visualization.

Hervé Marchand received the master's degree in mathematics from Université de Rennes 1 in 1993 and a Ph.D degree in computer science from the University of Rennes 1 in October 1997. From November 1997-October 1998, he was a Postdoctoral fellow at the University of Michigan, Ann Arbor, MI. Since 1998, he holds an INRIA research position at IRISA in Rennes in the VerTeCs project. His research interests include Supervisory Control, automatic test generation and diagnosis of Discrete Events Systems. He is also interested in high-level languages for reactive and real-time systems programming.