

# SUPREMAL NORMAL SUBLANGUAGES OF LARGE DISTRIBUTED DISCRETE-EVENT SYSTEMS

Jan Komenda <sup>\*,1</sup> Jan H. van Schuppen <sup>\*\*</sup>

*\* Czech Academy of Sciences, Brno Branch, Zizkova 22,  
616 62 Brno, Czech Republic*

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

Abstract: Coalgebraic methods provide new results and insights for modular supervisory control of discrete-event systems (DES), where the overall system is composed of subsystems that are themselves partially observed DES. It is well known that the computation of supremal normal sublanguages is computationally very difficult. The attention of this paper is focused on complex distributed systems that are composed of a large number of small subsystems that are combined in a modular fashion. Conditions are derived under which supremal normal sublanguages commute with synchronous product, i.e. the computation of supremal normal sublanguages can be done locally. The coinduction proof principle is used to obtain our main result.

Keywords: Modular discrete-event systems, Supremal normal sublanguages, Coalgebra, Coinduction

## 1. INTRODUCTION

The setting of this paper is that of the modular DES. A short historical overview of modular supervisory control of DES follows. The modular approach to the supervisory control of DES has been introduced in (Wonham and Ramadge, 1988) and (Ramadge and Wonham, 1989). The system is composed of local components (subsystems) that run concurrently (in parallel), i.e. the global system is the synchronous product of the local components. In the first papers on the topic the alphabets of the components were the same (Wonham and Ramadge, 1988), (Lin and Wonham, 1990). The general case of different local alphabets has been studied in (Willner and Hey-

mann, 1991), where a very restrictive condition is imposed on events shared by several local alphabets: they must be controllable for all subsystems. This assumption has been generalized recently in (Wong and Lee, 2002) to the condition that the shared events must have the same control status for all subsystems that share a particular event.

Very little attention has been paid so far to the modular control with partial observations. A special case of modular supervisory control with partial observations is studied in (Rohloff and Lafortune, 2003).

This paper aims at developing methods for computing supremal normal sublanguages of large distributed discrete-event systems. In this paper we focus only on questions that do not consider the blocking issues. These blocking issues are very important for modular supervisory control, but

---

<sup>1</sup> Partial financial support of the EU Esprit LTR Project Control and Computation, ISO-2001-33520 and the grant No. 201/03/P077 of the Grant Agency of Czech Republic is gratefully acknowledged.

they have already been studied in modular control with full observations.

The coalgebraic approach has been recently used for decentralized DES. The following problems are of interest in modular supervisory control: can the control be exerted at the local level without violating the global control objectives or without affecting the optimality of the solution? If the answers to these questions are positive, then there is an exponential save on the computational complexity. In our coalgebraic framework these two problems can be paraphrased as follows: when does the supervised product commute with the synchronous product and when does the supremal normal and/or controllable sublanguage commute with the synchronous product?

Commutativity between supremal controllable sublanguages and the synchronous product has been studied in (Wong and Lee, 2002). Conditions under which supremal normal sublanguages commute with synchronous product, i.e. the computation of supremal normal sublanguages can be done locally, are presented in this paper.

## 2. COALGEBRA AND COINDUCTION

Coalgebras are categorical duals of algebras (the corresponding functor operates from a given set rather than to a given set). The theory of universal coalgebra (Rutten 2000) has been developed in analogy with the corresponding theory of universal algebra. It turns out that state-transition systems, i.e. in particular various types of automata are coalgebras. They can be studied using coalgebraic concepts, e.g. bisimulations and coinduction in contrast with their usual algebraic study based on concepts like congruence and induction. Final coalgebras give rise to coinductive definition and proof principles. These are heavily used throughout the paper.

### 2.1 Partial automata

In this section we recall from (Rutten 1999) partial automata as coalgebras with a special emphasis on the final coalgebra of partial automata, i.e. partial automaton of partial languages.

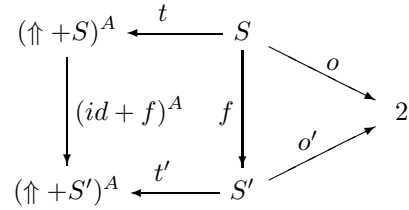
Let  $A$  be an arbitrary set (usually finite and referred to as the set of inputs or events). The empty string will be denoted by  $\varepsilon$ . Denote by  $\uparrow = \{\emptyset\}$  the one element set and by  $2 = \{0, 1\}$  the set of Booleans. A partial automaton is a pair  $S = (S, \langle o, t \rangle)$ , where  $S$  is a set of states, and a pair of functions  $\langle o, t \rangle : S \rightarrow 2 \times (\uparrow + S)^A$ , consists of an output function  $o : S \rightarrow 2$  and a transition function  $S \rightarrow (\uparrow + S)^A$ . The output

function  $o$  indicates whether a state  $s \in S$  is accepting (or terminating) :  $o(s) = 1$ , denoted also by  $s \downarrow$ , or not:  $o(s) = 0$ , denoted by  $s \uparrow$ . The transition function  $t$  associates to each state  $s$  in  $S$  a function  $t(s) : A \rightarrow (\uparrow + S)$ . The set  $\uparrow + S$  is the disjoint union of  $S$  and  $\uparrow$ . The meaning of the state transition function is that  $t(s)(a) = \emptyset$  iff  $t(s)(a)$  is undefined, which means that there is no  $a$ -transition from the state  $s \in S$ . Similarly,  $t(s)(a) \in S$  means that the  $a$ -transition from  $s$  is possible and we define in this case  $t(s)(a) = s_a$ , which is denoted mostly by  $s \xrightarrow{a} s_a$ . This notation can be extended by induction to arbitrary strings in  $A^*$ . Assuming that  $s \xrightarrow{w} s_w$  has been defined, define  $s \xrightarrow{wa} s_{wa}$  iff  $t(s_w)(a) \in S$ , in which case  $s_{wa} = t(s_w)(a)$  and  $s \xrightarrow{wa} s_{wa}$ .

A *homomorphism* between partial automata  $S = (S, \langle o, t \rangle)$  and  $S' = (S', \langle o', t' \rangle)$  is a function  $f : S \rightarrow S'$  with, for all  $s \in S$  and  $a \in A$ :

$$o'(f(s)) = o(s) \text{ and } s \xrightarrow{a} s_a \text{ iff } f(s) \xrightarrow{a} f(s)_a,$$

in which case  $f(s)_a = f(s_a)$  (see diagram below).



A partial automaton  $S' = (S', \langle o', t' \rangle)$  is a *sub-automaton* of  $S = (S, \langle o, t \rangle)$  if  $S' \subseteq S$  and the inclusion function  $i : S' \rightarrow S$  is a homomorphism. It is important to notice that the coalgebraic concept of subautomaton corresponds to the notion of strict subautomaton in (Cho and Marcus, 1989). In the sequel we use always subautomata in the coalgebraic sense defined above, i.e. strict subautomata are meant.

Note that partial automaton as defined above is just a coalgebraic reformulation of what is understood to be a generator of a DES. Indeed, the transition function can be viewed in the coalgebraic form above, and the output function determines the subset of marked or final states (those whose output value is equal to 1).

A *simulation* between two partial automata  $S = (S, \langle o, t \rangle)$  and  $S' = (S', \langle o', t' \rangle)$  is a relation  $R \subseteq S \times S'$  with, for all  $s \in S$  and  $s' \in S' : \langle s, s' \rangle \in R$

- (i)  $o(s) \leq o(s')$ , i.e.  $s \downarrow \Rightarrow s' \downarrow$ , and
- (ii)  $\forall a \in A : s \xrightarrow{a} \Rightarrow (s' \xrightarrow{a} \text{ and } \langle s_a, s'_a \rangle \in R)$ ,

A *bisimulation* between two partial automata  $S = (S, \langle o, t \rangle)$  and  $S' = (S', \langle o', t' \rangle)$  is a relation  $R \subseteq S \times S'$  with, for all  $s \in S$  and  $s' \in S' : \langle s, s' \rangle \in R$ :

- (i)  $o(s) = o(s')$ , i.e.  $s \downarrow$  iff  $s' \downarrow$
- (ii)  $\forall a \in A : s \xrightarrow{a} \Rightarrow (s' \xrightarrow{a} \text{ and } \langle s_a, s'_a \rangle \in R)$ ,
- (iii)  $\forall a \in A : s' \xrightarrow{a} \Rightarrow (s \xrightarrow{a} \text{ and } \langle s_a, s'_a \rangle \in R)$ .

We write  $s \sim s'$  whenever there exists a bisimulation  $R$  with  $\langle s, s' \rangle \in R$ . This relation is the union of all bisimulations, i.e. the greatest bisimulation also called bisimilarity. It is immediate from the definition of bisimulation that two states are bisimilar iff they can make the same transitions and they give rise to the same outputs:

*Proposition 2.1.* For any partial automaton  $S = (S, \langle o, t \rangle)$  and any  $s, s' \in S$ :  $s \sim s'$  iff  $\forall w \in A^* : s \xrightarrow{w} \iff s' \xrightarrow{w}$ , in which case  $o(s_w) = o'(s'_w)$ .

## 2.2 Final automaton of partial languages

In this subsection we define a partial automaton that is final among all partial automata and satisfies a proof principle called coinduction. The states of this final automaton represent minimal realizations of all possible behaviors (called partial languages) of all partial automata. Partial languages will be endowed with a (partial) automaton structure, which has the universal property of being final among all (partial) automata. The partial automaton of partial languages is defined using the Brzozowski notion of input derivative. Below we define the partial automaton of partial languages over an alphabet (input set)  $A$ , denoted by  $\mathcal{L} = (\mathcal{L}, \langle o_{\mathcal{L}}, t_{\mathcal{L}} \rangle)$ . More formally,

$\mathcal{L} = \{\Phi : A^* \rightarrow (\uparrow + 2) \mid \text{dom}(\Phi) = \{w \in A^* \mid \Phi(w) \in 2\} \neq \emptyset \text{ is prefix-closed}\}$ .

To each partial language  $\Phi$  a pair  $\langle V, W \rangle$  can be assigned:  $W = \text{dom}(\Phi)$  and  $V = \{w \in \text{dom}(\Phi) \mid \Phi(w) = 1(\in 2)\}$ . Conversely, to a pair  $\langle V, W \rangle \in \mathcal{L}$ , a function  $\Phi$  can be assigned:  $\Phi(w) = 1$  if  $w \in V$ ,  $\Phi(w) = 0$  if  $w \in W$  and  $w \notin V$ , and  $\Phi(w)$  is undefined if  $w \notin W$ . Therefore we can write :

$\mathcal{L} = \{\langle V, W \rangle \mid V \subseteq W \subseteq A^*, W \neq \emptyset, \text{ and } W \text{ is prefix-closed}\}$ .

The transition function  $t_{\mathcal{L}} : \mathcal{L} \rightarrow (1 + \mathcal{L})^A$  is defined using input derivatives. Recall that for any partial language  $L = (L^1, L^2) \in \mathcal{L}$ ,  $L_a = (L_a^1, L_a^2)$ , where  $L_a^i = \{w \in A^* \mid aw \in L^i\}$ ,  $i = 1, 2$ . If  $a \notin L^2$  then  $L_a$  is undefined. Given any  $L = (L^1, L^2) \in \mathcal{L}$ , the partial automaton structure of  $\mathcal{L}$  is given by:

$$o_{\mathcal{L}}(L) = \begin{cases} 1 & \text{if } \varepsilon \in L^1 \\ 0 & \text{if } \varepsilon \notin L^1 \end{cases}$$

and

$$t_{\mathcal{L}}(L)(a) = \begin{cases} L_a & \text{if } L_a \text{ is defined} \\ \emptyset & \text{otherwise} \end{cases}$$

Notice that if  $L_a$  is defined, then  $L_a^1 \subseteq L_a^2$ ,  $L_a^2 \neq \emptyset$ , and  $L_a^2$  is prefix-closed. The following notational conventions will be used:  $L \downarrow$  iff  $\varepsilon \in L^1$ , and  $L \xrightarrow{w} L_w$  iff  $L_w$  is defined iff  $w \in L^2$ . The

following two theorems are recalled from (Rutten 1999).

*Theorem 2.2.* Partial automaton  $\mathcal{L} = (\mathcal{L}, \langle o_{\mathcal{L}}, t_{\mathcal{L}} \rangle)$  is final among all partial automata: for any partial automaton  $S = (S, \langle o, t \rangle)$  there exists a unique homomorphism  $l : S \rightarrow \mathcal{L}$ . This homomorphism identifies bisimilar states: for  $s, s' \in S$ :  $l(s) = l(s')$  iff  $s \sim s'$ .

*Theorem 2.3.*  $\mathcal{L}$  satisfies the principle of coinduction: for all  $K$  and  $L$  in  $\mathcal{L}$ , if  $K \sim L$  then  $K = L$ .

Coinduction is used as a proof and definition principle throughout this paper. The use of coinduction is limited to final coalgebras. Behavior equivalence of two elements of final coalgebra means that these are equal.

Proofs by coinduction consist in constructing appropriate relations: for instance a proof of equality of two elements of a final coalgebra consists in finding a bisimulation relation that relates them. Definition by coinduction of an operation on elements of a final coalgebra consists in defining the same coalgebraic structure on the operation (for instance we define binary operations on partial languages by defining derivatives and output functions further in this paper).

Recall from (Rutten 1999) the following coinductive definitions of the synchronous product. We assume that  $K$  is defined over the alphabet  $A_1$  and  $L$  over  $A_2$ . Then the synchronous product  $K \parallel L$  is a language over  $A_1 \cup A_2$  with the following coinductive definition:

*Definition 2.1.*

$$(K \parallel L)_a = \begin{cases} K_a \parallel L_a & \text{if } a \in A_1 \cap A_2 \\ K_a \parallel L & \text{if } a \in A_1 \setminus A_2 \\ K \parallel L_a & \text{if } a \in A_2 \setminus A_1 \end{cases}$$

and  $(K \parallel L) \downarrow$  iff  $K \downarrow$  and  $L \downarrow$ .

## 3. MODULAR DES WITH PARTIAL OBSERVATIONS

A modular DES represented by a partial automaton  $S$  consists of local subsystems (modules)  $S_i$ ,  $i \in Z_n = \{1, \dots, n\}$  such that the global DES  $S$  is the parallel composition (synchronous product) of  $S_i$ . We assume that each module  $S_i$  has only partial observation of its events, i.e.  $A_i = A_{o,i} \cup A_{uo,i}$  is the decomposition of local events into locally observable and locally unobservable. The global system has observation set  $A_o = \cup_{i=1}^n A_{o,i} \subseteq A = \cup_{i=1}^n A_i$ . Additional notation is needed to set up our framework. 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 local ones are denoted by  $P_i : A^* \rightarrow A_{o,i}^*$ . Partial observations in individual modules are expressed via local projections  $P_i^{loc} : A_i^* \rightarrow A_{o,i}^*$ , while global projection is denoted by  $P : A^* \rightarrow A_o^*$ . Local plant languages will be denoted by  $L_i$ ,  $i \in Z_n$  and local specification languages by  $K_i$ ,  $i \in Z_n$ . We assume in our main theorem that  $n = 2$ , because the generalization to arbitrary  $n$  is quite straightforward. The following lemma concerning the relation between global and local observations is needed.

*Lemma 1.* If  $A_{o,2} \cap A_1 = A_2 \cap A_{o,1}$  then for any  $s, s' \in A^*$  we have:  
if  $P(s) = P(s')$  then for  $i = 1, 2$ :  $P_i^{loc} P_i(s) = P_i^{loc} P_i(s')$ .

An auxiliary concept that reflects the fact that due to partial observations it is not possible to distinguish between states is needed. The notation  $\xrightarrow{\varepsilon}$  is used for unobservable moves, i.e.  $s \xrightarrow{\varepsilon} s'$  iff  $\exists \tau \in A_{uo}^* : s \xrightarrow{\tau} s_\tau = s'$ .

*Definition 3.1.* (Observational indistinguishability relation on  $S$ .) A binary relation  $Aux(S)$  on  $S$ , called *observational indistinguishability relation* is the smallest relation satisfying:

- (i)  $\langle s_0, s_0 \rangle \in Aux(S)$
- (ii) If  $\langle s, t \rangle \in Aux(S)$  then :  $(s \xrightarrow{\varepsilon} s' \text{ for some } s' \text{ and } t \xrightarrow{\varepsilon} t' \text{ for some } t') \Rightarrow \langle s', t' \rangle \in Aux(S)$
- (iii) If  $\langle s, t \rangle \in Aux(S)$  then  $\forall a \in A_o : (s \xrightarrow{a} s_a \text{ and } t \xrightarrow{a} t_a) \Rightarrow \langle s_a, t_a \rangle \in Aux(S)$ .

$Aux(S)$  can be characterized by the following lemma.

*Lemma 2.* For any  $s, s' \in S$ :  $\langle s, s' \rangle \in Aux(S)$  iff there exist two strings  $w, w' \in A^*$  such that  $P(w) = P(w')$ ,  $s = (s_0)_w$  and  $s' = (s_0)_{w'}$ .

#### 4. SUPREMAL NORMAL SUBLANGUGES

It has been shown in (Wong and Lee, 2002) that a structural condition, called mutual controllability, is important for preserving optimality in the modular control with full observations (i.e. commutativity of the supremal controllable sublanguage with the synchronous product of partial languages). The natural question arises: With respect to which conditions this can be generalized to modular control with partial observations? Main problem of this paper is: when does the supremal normal sublanguage commutes with the synchronous product? Unfortunately, it is not possible to the best of our knowledge to define the supremal normal sublanguage by coinduc-

tion. The argument is similar to the corresponding argument for closed-loop languages under the antipermissive control policy (Komenda, 2004). Nevertheless, using suitable automata representations: state-partition automata (Yoo *et al.*, 2001) there is a monolithic algorithm for the computation of the supremal normal sublanguage. Algorithm 1 below looks almost like a coinductive definition and it will be used in the coinductive proof of our main theorem.

We use the representations of languages  $K$  and  $L$  by automata  $S_1$  and  $S$ , where  $S_1$  is a state-partition automaton (Yoo *et al.*, 2001). It is known how to construct such representations, see (Cho and Marcus, 1989) or (Yoo *et al.*, 2001). Let  $s_0$  denote the common initial state of  $S_1$  and  $S$ . The transition structure of  $S_1$  and  $S$  is denoted by  $\rightarrow_1$  and  $\rightarrow$ , respectively.

Normality of a specification language is an important property of partially observed DES, introduced in (Lin and Wonham, 1988).

*Definition 4.1.* (Normality.) Let  $K, L \in \mathcal{L}$ :  $K \subseteq L$ .  $K$  is said to be  $(L, P)$ -normal if  $K^2 = L^2 \cap P^{-1}(P(K^2))$ .

We recall below normal relations from (Komenda, 2002).

*Definition 4.2.* (Normal relation.) Given two (partial) automata  $S_1 = (S_1, \langle o_1, t_1 \rangle)$  and  $S = (S, \langle o, t \rangle)$  as above with common initial state  $s_0 \in S$ , a binary relation  $N(S_1, S)$  on  $S_1 \times S$  is called a *normal relation* if for any  $\langle s, t \rangle \in N(S_1, S)$  the following items hold:

- (i)  $\forall a \in A : s \xrightarrow{a}_1 s_a \Rightarrow t \xrightarrow{a} t_a \text{ and } \langle s_a, t_a \rangle \in N(S_1, S)$
- (ii)  $\forall u \in A_{uo} : t \xrightarrow{u} t_u \Rightarrow s \xrightarrow{u}_1 s_u$ .
- (iii)  $\forall a \in A_o : t \xrightarrow{a} t_a \text{ and } (\exists s' : \langle s, s' \rangle \in Aux(S_1) : s' \xrightarrow{a}_1 s'_a) \Rightarrow s \xrightarrow{a}_1 s_a$ .

We have shown in (Komenda, 2002) that

*Theorem 4.1.* A (partial) language  $K$  is  $(L, P)$ -normal iff there exists a normal relation  $N(S_1, S)$  on  $S_1 \times S$  such that  $\langle s_0, s_0 \rangle \in N(S_1, S)$ .

*Remark 4.2.* We consider from now on an order relation on  $\mathcal{L}$  induced by their second components only, i.e. we write  $K \subseteq L \in \mathcal{L}$  iff  $K^2 \subseteq L^2$ . The same applies for supremum operation. Note that only the second condition (ii) of simulation relations must be checked to prove such defined inclusion of partial languages.

Now we are ready to formulate the algorithm for computation of supremal  $(L, P)$ -normal sublanguages.

*Algorithm 1.* Let automata  $S_1$  and  $S$  representing  $K$  and  $L$ , respectively be such that  $S_1$  is a subautomaton of  $S$  and  $S_1$  is a state-partition automaton. Let us construct partial automaton  $\tilde{S} = \langle \tilde{o}, \tilde{t} \rangle$  with  $\tilde{t}$  denoted by  $\rightarrow$ .

Define the auxiliary condition (\*) as follows:

if  $a \in A_{uo}$  then  $\forall u \in A_{uo}^*: s_a \xrightarrow{u} \Rightarrow s_a \xrightarrow{u}_1$ ;  
if  $a \in A_o$  then  $\forall s' \approx_{Aux(S_1)} s : s' \xrightarrow{a} \Rightarrow s' \xrightarrow{a}_1$ , in which case also  $\forall u \in A_{uo}^*: s'_a \xrightarrow{u} \Rightarrow s'_a \xrightarrow{u}_1$ .

Below are the steps of the algorithm.

1. Put  $\tilde{S} := \{s_0\}$ .
2. For any  $s \in \tilde{S}$  and  $a \in A$  we put  $s \xrightarrow{a}$ ,  $s_a$  if condition (\*) is satisfied and we put in the case  $s \xrightarrow{a}$ , also  $\tilde{S} := \tilde{S} \cup \{s_a\}$ .
3. For any  $s \in \tilde{S}$  we put  $\tilde{o}(s) = o(s)$ .

Let us denote by  $\tilde{l} : \tilde{S} \rightarrow \mathcal{L}$  the unique (behavior) homomorphism given by finality of  $\mathcal{L}$ . It can be shown that

*Theorem 4.3.*  $\tilde{l}(s_0)$  is the supremal  $(L, P)$ -normal sublanguage of  $K$ .

In our main theorem a condition similar to mutual controllability (Wong and Lee, 2002) is needed. We call it by analogy mutual normality.

*Definition 4.3.* Given partial languages  $L_i = (L_i^1, L_i^2)$  and  $L_j = (L_j^1, L_j^2)$ ,  $L_i$  and  $L_j$  are said to be mutually normal if

$$(P_i^{loc})^{-1} P_i^{loc} (L_i^2) \cap P_i (P_j)^{-1} (L_j^2) \subseteq L_i^2$$

and

$$(P_j^{loc})^{-1} P_j^{loc} (L_j^2) \cap P_j (P_i)^{-1} (L_i^2) \subseteq L_j^2.$$

The following lemma will be needed for the proof of our main theorem.

*Lemma 3.* Assume that  $A_{o,2} \cap A_1 = A_2 \cap A_{o,1}$ . Let  $v \in A^*$  with  $P_1(v) = v_1 \in A_1^*$  and  $P_2(v) = v_2 \in A_2^*$ . Let  $v'_1 \in A_1^*$  be such that  $P_1^{loc}(v_1) = P_1^{loc}(v'_1)$ . Then there exists  $v' \in A^*$  such that  $P_1(v') = v'_1$ ,  $P_2^{loc} P_2(v') = P_2^{loc}(v_2)$ , and  $P(v) = P(v')$ .

Let us introduce the notation  $\sup N(K, L, P)$  for the supremal  $(L, P)$ -normal sublanguage of  $K$ . Recall that local plant languages are denoted by  $L_i$ ,  $i \in Z_2$  and local specification languages by  $K_i$ ,  $i \in Z_2$ . This means that we assume that the global specification language  $K$  is decomposable into local specification languages  $K_i$ ,  $i \in Z_2$ , where  $K = K_1 \parallel K_2$ . Our main theorem follows. The coinductive proof principle and Algorithm 1 are used in the proof.

*Theorem 4.4.* If  $A_{o,2} \cap A_1 = A_2 \cap A_{o,1}$ , and  $L_1$  and  $L_2$  are mutually normal, then  
 $\sup N(K_1, L_1, P_1^{loc}) \parallel \sup N(K_2, L_2, P_2^{loc}) = \sup N(K_1 \parallel K_2, L_1 \parallel L_2, P)$ .

Under very general conditions we have always one inclusion:

*Corollary 1.* If  $A_{o,2} \cap A_1 = A_2 \cap A_{o,1}$ , then we have  
 $\sup N(K_1, L_1, P_1^{loc}) \parallel \sup N(K_2, L_2, P_2^{loc}) \subseteq \sup N(K_1 \parallel K_2, L_1 \parallel L_2, P)$ .

Note that Theorem 4.4 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 synchronize them.

The interest of this theorem should be clear: under the conditions that are stated it is possible to do the optimal (less restrictive) control synthesis with partial observations locally, which represents an exponential save on the computational complexity and makes in fact the optimal control synthesis of some large distributed plants feasible. Note that an extension of our results from  $n = 2$  to an arbitrary number  $n \in N$  of local modules is quite straightforward and thus omitted in this paper. The condition of mutual normality between any pair of local plants is required. In (Wong and Lee, 2002) there is a procedure how to arrange the mutual controllability for distributed plants, where this condition is not satisfied. It is conjectured that a similar procedure exists for mutual normality, possibly for special cases.

The condition of mutual normality is a structural condition, because it depends only on the (local) plant languages and not on the specification languages. The advantage is that for a given modular DES it must be checked only once and then it holds for arbitrary specification languages. In order to get more intuition about mutual normality we consider some extreme special cases of distributed DES. First of all, if all event alphabets are disjoint, the so called shuffle case, we notice that  $P_i(P_j)^{-1}(L_j^2) = A_i^*$  for any  $L_j^2 \subseteq A_j^*$ . This means that the condition of mutual normality can not be satisfied. Thus one can not hope to find a universal procedure how to make a set of local plant languages mutually normal. The intuitive reason is that there is only a weak interconnection 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, mutual nor-

mality is trivially satisfied. Another extreme case occurs when all subsystems have the same event alphabets. Then all  $P_i$ '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. Note that mutual normality is not a necessary condition for commutativity between supremal normal sublanguages and the synchronous product as can be seen from simple examples.

## 5. CONCLUSION

Supremal normal sublanguages are important for the design of supervisory controllers with partial observations. For large distributed plants there is a problem of high computational complexity (Cho and Marcus 1989). Therefore the conditions under which the supremal normal sublanguage of the global plant can be computed using supremal normal sublanguages of the local plants are of a special interest. Coalgebraic framework for supervisory control of distributed systems with partially observed modules is used. The sufficient conditions obtained in this paper are discussed in some special cases.

There are many open problems left for a future investigation. For instance, all conditions we have obtained are only sufficient conditions. The question is whether they can be weakened, at least in special cases that occur in some relevant applications to be found. Another direction of future research is to study the conditions for commutativity between the synchronous product and the closed-loop languages using the antipermissive control policy. In this paper the blocking issues have not been considered. It is to be expected that for modular DES with partial observations the conditions for nonblocking are also the same as for modular DES with full observations (Wong and Lee, 2002), however it must still be proven.

Effective procedures for verification of our conditions must be found. For this purpose our coalgebraic approach helps through the relational characterization of all conditions that have been defined algebraically (mutual normality, mutual controllability and decomposability). Since the new properties are similar to their classical counterparts, we believe that such effective procedures can be found. Even more, mutual controllability and mutual normality are likely to be much easier to verify than global controllability and normality, because the corresponding plant and specification languages involved are local, thus much smaller.

## REFERENCES

- Cassandras, S.G. and Lafortune, S. (1999). *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, Dordrecht 1999.
- Cho, H. and Marcus, S. I. (1989). Supremal and Maximal Sublanguages Arising in Supervisor Synthesis Problems with Partial Observations. *Math. Systems Theory*, **22**, 171-211, 1989.
- Komenda, J. (2002). Computation of Supremal Sublanguages of Supervisory Control Using Coalgebra. Proceedings *WODES'02*, Workshop on Discrete-Event Systems, Zaragoza, 26-33.
- Komenda, J. (2004). Coalgebra and coinduction in discrete-event control, submitted to *Discrete Event Dynamical Systems: Theory and Applications*.
- Lin, F. and Wonham, W.M. (1988). On Observability of Discrete-Event Systems. *Information Sciences*, **44**, 173-198.
- Lin, F. and Wonham, W.M. (1990). Decentralized Control and Coordination of Discrete-Event Systems with Partial Observations, *IEEE Trans. Automatic Control*, **35**, 1330-1337.
- Rohloff, K. and Lafortune, S. (2003). The control and Verification of Similar Agents Operating in a Broadcast Network Environment. *In Proceedings CDC 2003*, Hawaii, USA.
- Rutten, J.J.M.M. (1999). Coalgebra, Concurrency, and Control. *Research Report CWI, SEN-R9921*, Amsterdam. Available also at <http://www.cwi.nl/~janr>.
- Rutten, J.J.M.M. (2000). Universal Coalgebra: A Theory of Systems. *Theoretical Computer Science* **249(1)**, 3-80.
- Ramadge, P.J. and Wonham, W.M. (1989). The Control of Discrete-Event Systems. *Proc. IEEE*, **77**:81-98.
- Willner, Y. and Heymann, M. (1991). Supervisory control of concurrent discrete-event systems. *International Journal of Control*, **54(5)**, 1143-1166.
- Wong, K.C. and Lee, S. (2002). Structural Decentralized Control of Concurrent Discrete-Event Systems. *European Journal of Control*, **0**, 1-15.
- Wonham, W.M. and Ramadge, P.J. (1988) Modular supervisory control of discrete-event processes, *Mathematics of Control, Signal and Systems*, **1**, 13-30.
- Yoo, T.S., Lafortune, S., and Lin, F. (2001). A Uniform Approach for Computing Supremal Sublanguages Arising in Supervisory Control theory. *Control group report CGR 02-04 Dept. of EECS, Univ. of Michigan*, Ann Arbor.