Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Supremal Sublanguages of General Specification Languages Arising in Modular Control of Discrete-Event Systems

J. Komenda
Institute of Mathematics, Czech Academy of Sciences, Brno Branch, Zizkova 22, 616 62 Brno, Czech Republic
komenda@ipm.cz

J.H. van Schuppen
CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
J.H.van.Schuppen@cwi.nl

***Abstract*— For supervisory control of large-scale modular DES the possibility of performing control-related computation locally (in components) is of utmost importance to computational complexity. Unlike our previous results, where the specification language is decomposable into local specification languages, this paper concerns the case of general specification languages that are neither necessarily decomposable nor contained in the global plant language. Still the conditions are found under which building of the global plant is avoided for the computation of the supremal normal sublanguages of the (global) specification language.**

Keywords— Modular supervisory control, Partial normality, Coalgebra, Supremal sublanguages

I. Introduction

Modular control of DES represented by finite automata has been introduced by P.J. Ramadge and W.M. Wonham in [9]. Large scale systems are typically composed of a large number of relatively small (in size) local components (subsystems) that run concurrently (in parallel). Global systems are formed as a synchronous product of these local components. Unlike the first papers on the topic, where the input alphabets of the local components were identical ([12]), the general case of different local inputs is considered. In [10] and [2] 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 [11] to the condition that the shared events must have the same control status for all subsystems that share a particular event. Specification languages for the global plant that are not decomposable into local specification languages are considered in this paper.

Our attention is restricted to modular control synthesis without blocking as the blocking issue requires different concepts and methods. Preliminary results on coalgebra and coinduction can be found in the apppendix. After the introductory Section 1, Section 2 is devoted to the problem formulation. Section 3 concerns the computation of the supremal normal sublanguage of a general specification language. Two novel sufficient conditions (a specification dependent and a structural one) are derived under which supremal normal sublanguage can be computed without building the global plant itself.

The research was supported by the EU Esprit LTR Project Control and Computation, ISO-2001-33520 and Acad. of Sci. of Czech Republic, Inst. Research Plan No. AV0Z10190503.

II. Problem formulation

In modular control the concurrent behavior of local subplants (partial automata) $G_1, \ldots, G_n$ is considered. The notation $\mathbb{Z}_n = \{1, 2, \ldots, n\}$ is used. The global plant is the synchronous product $G = \|_{i=1}^n G_i$. Denote $A = \cup_{i=1}^n A_i$ the global alphabet and $P_i : A^* \to A_i^*$ the projections to the local alphabets. The concept of inverse projection: $P_i^{-1} : \mathrm{Pwr}(A_i^*) \to \mathrm{Pwr}(A^*)$ is also used.

Denote the global plant and specification (partial) languages by L and K, respectively. In our modular setting, L is decomposable into local plant languages: $L = L_1 \parallel \cdots \parallel L_n$. 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 has been considered in [2] for complete observations under restrictive structural conditions (all shared events are controllable). We assume that each module S_i has only partial observation of its events. The main goal of the paper is to find methods for computation of supremal normal sublanguages that cope with the computational difficulty of this problem. We are looking for conditions that enable computation of the supremal normal sublanguage which avoid any manipulation with the global plant. In this way the modularity can be used to break down considerably the computational complexity.

III. Supremal Normal Sublanguages of General Specification Languages

Let $A_i = A_{o,i} \cup A_{uo,i}$ be the decomposition of local events into locally observable ($A_{o,i}$) and locally unobservable ($A_{uo,i}$) event subsets. The global system has the 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 $A_{uo} = A \setminus A_o$. We assume in this section that $\forall i \neq j \in \{1, \ldots, n\}$ we have $A_{o,i} \cap A_j = A_i \cap A_{o,j}$, i.e. observational status of a shared event must be the same for all modules that share a particular event. Therefore we have also $A_{uo} = \cup_{i=1}^n A_{uo,i}$. The corresponding global projection that erases unobservable events is denoted by $P : A^* \to A_o^*$.

Recall that the projections of the global alphabet into the local ones are denoted by $P_i : A^* \to A_i^*,\ i \in \mathbb{Z}_n$. Partial observations in individual modules are expressed via local projections from A_i^* to $A_{o,i}^*$, but due to the handling of general indecomposable (global) specification languages only the afore mentioned global projection $P : A^* \to A_o^*$ is considered in this section. The concept of $\varepsilon-$transitions is needed.

Definition 3.1: ($\varepsilon-$transitions.) For $s \in S$ we define $s \stackrel{\varepsilon}{\Rightarrow} s'$ if $\exists \tau \in A_{uo}^* :\ s \stackrel{\tau}{\to} s_\tau = s'$.

Denote the initial state of the DES generator S by s_0. An auxiliary concept that reflects the fact that due to partial observations it is not possible to distinguish between states is recalled from [3]:

Definition 3.2: (Observational indistinguishability relation on S.) A binary relation $Aux(S)$ on S, called the *observational indistinguishability relation* is the smallest relation satisfying:

(i) $\langle s_0, s_0 \rangle \in Aux(S)$

(ii) $\forall \langle s, t \rangle \in Aux(S) : (s \stackrel{\varepsilon}{\Rightarrow} s' \text{ and } t \stackrel{\varepsilon}{\Rightarrow} t') \Rightarrow \langle s', t' \rangle \in Aux(S)$

(iii) $\forall \langle s, t \rangle \in Aux(S)$ and $\forall a \in A_o : (s \stackrel{a}{\to} s_a \text{ and } t \stackrel{a}{\to} t_a) \Rightarrow \langle s_a, t_a \rangle \in Aux(S)$.

Next we recall the concept of state-partition automaton, which we define here as follows:

Definition 3.3: (State-partition automaton) A partial automaton S is called a state-partition automaton if $\forall s \in S$: $s = (s_0)_{w_1} = (s_0)_{v_1}$ for $w_1 \in A^*$ and $v_1 \in A^*$ and $\forall s' \in S$: $s' = (s_0)_{v_2}$ for $v_2 \in A^*$ with $P(v_2) = P(v_1)$ there exists $w_2 \in A^*$ with $P(w_2) = P(w_1)$ and $s' = (s_0)_{w_2}$.

Note that if $P(w_1) = P(v_1)$ then it is sufficient to put $w_2 = v_2$, i.e. it is equivalent to require the above condition only for $P(w_1) \neq P(v_1)$.

We assume that partial language L is represented by partial automaton S with initial state s_0. We mean by this that the corresponding behavior homomorphisms $l : S \to \mathcal{L}$ (see appendix) satisfies $l(s_0) = L$. A characterization of $Aux(S)$ follows:

Lemma 3.1: For any $s, s' \in S$: $\langle s, s' \rangle \in Aux(S)$ iff there exist $w, w' \in L^2$ such that $P(w) = P(w')$, $s = (s_0)_w$ and $s' = (s_0)_{w'}$. Moreover, if S is a state-partition automaton then $\forall v \in L^2$ and $s' \in S$ we have $\langle (s_0)_v, s' \rangle \in Aux(S)$ iff there exists $v' \in L^2$ such that $P(v) = P(v')$ and $s' = (s_0)_{v'}$.

Proof: The first part is very easy and has been shown in our previous works on the topic. The second part follows from the assumption that S is a state-partition automaton: if for $v \in L^2$ we have $\langle (s_0)_v, s' \rangle \in Aux(S)$ an application of the first part of this lemma yields there exist $w, w' \in K^2$ such that $P(w) = P(w')$, $(s_0)_v = (s_0)_w$ and $s' = (s_0)_{w'}$, the assumption that S is a state-partition automaton yields that there exists $v' \in L^2$ such that $P(v) = P(v')$, and $s' = (s_0)_{v'}$, which is the second claim. ■

Local plant languages will be denoted by L_i, $i \in \mathbb{Z}_n = \{1, \ldots, n\}$. We assume that global specification K is not decomposable into local specifications. Recall from [1] that K is called $(L, P)-$*normal* if $K^2 = P^{-1}P(K^2) \cap L^2$. It is known that normal languages are closed under unions, hence supremal normal sublanguages always exist.

Remark 3.2: An order relation on partial languages induced by second components only is used: we write $K \subseteq L$ iff $K^2 \subseteq L^2$.

We denote the supremal $(L, P)-$*normal* sublanguage of K by $\sup \mathrm{N}(K, L, P)$. Since K is not in general included in L we investigate $\sup \mathrm{N}(K \cap L, L, P)$. For any $i \in \mathbb{Z}_n$ we can consider language $K_i := K \cap P_i^{-1}(L_i)$, which will play the role of local specification in the case of indecomposable specification K. Note however that K_i is a partial language over the global alphabet A. Supremal normal sublanguages of K_i with respect to $P_i^{-1}(L_i)$ and P, i.e. $\sup \mathrm{N}(K_i, P_i^{-1}(L_i), P)$, are considered for any $i \in \mathbb{Z}_n$. It will be shown in the sequel that under certain conditions their intersection yields $\sup \mathrm{N}(K \cap L, L, P)$. It is natural that such an approach is not always possible and the optimality (supremality) is in general lost.

We are looking for conditions that enable computation of $\sup \mathrm{N}(K \cap L, L, P)$ without having to build the recognizer of L itself, i.e. using only given K and $P_i^{-1}(L_i),\ i \in \mathbb{Z}_n$. In order to make possible the computation of $\sup \mathrm{N}(K \cap L, L, P)$ using $\sup \mathrm{N}(K_i, P_i^{-1}(L_i), P)$, $(P_i^{-1}(L_i), P)-$normality of K_i for any $i \in \mathbb{Z}_n$ is required. This condition will be called $G-$*normality*.

Now we recall from [5]:

Algorithm 1: (Supremal normal sublanguage) Let (partial) automata S and T representing $K \cap L$ and L, respectively, be such that S is a subautomaton of T and S is a state-partition automaton. The transition functions of S and T are denoted by $\to_1$ and $\to$, respectively. Partial automaton $\tilde{S} = \langle \tilde{o}, \tilde{t} \rangle$, subautomaton of S, is constructed with $\tilde{t}$ denoted by $\to_\prime$.

Define the auxiliary condition (1) consisting of (1a) and (1b) as follows:

(1a) if $a \in A_{uo}$ then $\forall u \in A_{uo}^*$: $s_a \stackrel{u}{\to} \Rightarrow\ s_a \stackrel{u}{\to}_1$;

(1b) if $a \in A_o$ then $\forall s' \approx_{Aux(S)} s : s' \stackrel{a}{\to} \Rightarrow\ s' \stackrel{a}{\to}_1$, in which case also $\forall u \in A_{uo}^*$: $s'_a \stackrel{u}{\to} \Rightarrow\ s'_a \stackrel{u}{\to}_1$.

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 $s \stackrel{a}{\to}_\prime s_a$ if $s \stackrel{a}{\to}_1$ and condition (1) is satisfied. We put in the case $s \stackrel{a}{\to}_\prime s_a$ also

$\tilde{S} := \tilde{S} \cup \{s_a\}$.
3. For any $s \in \tilde{S}$ we put $\tilde{o}(s) = o(s)$.

Let $\tilde{l} : \tilde{S} \to \mathcal{L}$ be the unique (behavior) homomorphism given by finality of $\mathcal{L}$. We know from [5]

Theorem 3.3: (Algorithm 1 is correct) $\tilde{l}(s_0) = \sup \mathrm{N}(K \cap L, L, P)$.

The same auxiliary algorithm (based on Algorithm 1) with modified data is used for computation of $\sup \mathrm{N}(K_i, P_i^{-1}(L_i), P)$. It is explicitly stated below in order to make clear our notation. Thus the proof of our main theorem becomes easier to follow.

Algorithm 2: Let partial automata $S_i = (S_i, \langle o_{1i}, t_{1i}\rangle)$ and $T_i = (T_i, \langle o_i, t_i\rangle)$ representing K_i and $P_i^{-1}(L_i)$, $i \in \mathbb{Z}_n$, respectively, be such that for all $i \in \mathbb{Z}_n$: S_i is a subautomaton of T_i, and S_i is a state-partition automaton. The common initial state of these automatata is denoted by s_0^i and the transition functions t_{1i} and t_i are denoted by $\to_{1i}$ and $\to_i$, respectively. Denote by $Aux(S_i)$ the observation indistinguishability relation with respect to the projection P.

Let us construct partial automata $\tilde{S}_i = (\tilde{S}_i, \langle \tilde{o}_i, \tilde{t}_i\rangle)$, subautomata of S_i, with $\tilde{t}_i$ denoted by $\to\!\!\!\!\!\!\sim_i$.
Define the auxiliary condition (2) consisting of (2a) and (2b) as follows:
(2a) if $a \in A_{uo}$ then $\forall u \in A_{uo}^*$: $s_a \overset{u}{\to}_i \Rightarrow\ s_a \overset{u}{\to}_{1i}$;
(2b) if $a \in A_o$ then $\forall s' \approx_{Aux(S_i)} s :\ s' \overset{a}{\to}_i \Rightarrow\ s' \overset{a}{\to}_{1i}$, in which case also $\forall u \in A_{uo}^*$: $s'_a \overset{u}{\to}_i \Rightarrow\ s'_a \overset{u}{\to}_{1i}$.

Below are the steps of the algorithm.
1. Put $\tilde{S}_i := \{s_0^i\}$.
2. For any $s \in \tilde{S}_i$ and $a \in A$ we put $s \overset{a}{\to\!\!\!\!\!\!\sim}_i s_a$ if $s \overset{a}{\to}_{1i} s_a$ and condition (2) is satisfied and we put in the case $s \overset{a}{\to\!\!\!\!\!\!\sim}_i s_a$ also $\tilde{S}_i := \tilde{S}_i \cup \{s_a\}$.
3. For any $s \in \tilde{S}_i$ we put $\tilde{o}_i(s) = o_i(s)$.

Let us denote by $\tilde{l}_i : \tilde{S}_i \to \mathcal{L}$ the unique (behavior) homomorphism given by finality of $\mathcal{L}$. It follows from Theorem 3.3 that

Theorem 3.4: (Algorithm 2 is correct) $\tilde{l}_i(s_0^i) = \sup \mathrm{N}(K_i, P_i^{-1}(L_i), P)$.

Now we formulate the concept of $G-$normality.

Definition 3.4: $K \subseteq L$ is called $G-$normal if for all $i \in \mathbb{Z}_n$:
$K_i := K \cap P_i^{-1}(L_i)$ is $(P_i^{-1}(L_i), P)-$normal.

We are ready to state the following simple theorem.

Theorem 3.5: (Sufficient conditions) Assume that $\forall i \neq j \in \{1, \ldots, n\}$ we have $A_{o,i} \cap A_j = A_i \cap A_{o,j}$ and K is $G-$normal. Then

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

Proof: Algorithm 1 is used for the computation of $\sup \mathrm{N}(K \cap L, L, P)$ and Algorithm 2 for computation of $\sup \mathrm{N}(K_i, P_i^{-1}(L_i), P)$. This enables us to consider the behaviors of the corresponding output automata $\tilde{S}$ and $\tilde{S}_i$ of Algorithms 1 and 2, i.e. coinductive proof principle can be used (see appendix). The notation is as follows: let S representing $K \cap L$ and T representing L be the same as in Algorithm 1. Similarly, for $i \in \mathbb{Z}_n$, S_i and T_i representing $K_i = K \cap P_i^{-1} L_i$ and $P_i^{-1} L_i$ be the same as in Algorithm 2. We show that

$$R = \{\langle [\tilde{l}(s_0)]_w, [\bigcap_{i=1}^{n} \tilde{l}_i(s_0^i)]_w \rangle \mid w \in (\tilde{l}(s_0))^2\}$$

is a bisimulation relation, from which the claim of the theorem follows by coinduction. Take a $w \in (\tilde{l}(s_0))^2$ arbitrary, but fixed.
(i) is trivial: marking is not considered.
(ii) Let $[\tilde{l}(s_0)]_w \overset{a}{\to}$ for $a \in A$, i.e. $(s_0)_w \overset{a}{\to\!\!\!\!\!\!\sim}$. Thus, according to step 2 of Algorithm 1 $(s_0)_w \overset{a}{\to}_1$ and condition (1) of Algorithm 1 is satisfied. It must be shown that $[\cap_{i=1}^n \tilde{l}_i(s_0^i)]_w \overset{a}{\to}$. We need show that for any $i \in \mathbb{Z}_n$ we have $[\tilde{l}_i(s_0^i)]_w \overset{a}{\to}$, i.e. $(s_0^i)_w \overset{a}{\to\!\!\!\!\!\!\sim}_i$. According to Algorithm 2 this amounts to show that for any $i \in \mathbb{Z}_n$ we have $(s_0^i)_w \overset{a}{\to}_{1i}$ and condition (2) of Algorithm 2 holds. First of all note that $(s_0^i)_w \overset{a}{\to}_{1i}$. Indeed, $(s_0)_w \overset{a}{\to\!\!\!\!\!\!\sim}$ implies that $(s_0)_w \overset{a}{\to}_1$, i.e. $wa \in (K \cap L)^2$. Since we have also that $(s_0)_w \overset{a}{\to}$, i.e. $wa \in L^2 = \cap_{i=1}^n P_i^{-1}(L_i^2)$, it follows that $wa \in K_i^2 = K^2 \cap P_i^{-1}(L_i^2)$, which is equivalent to $(s_0^i)_w \overset{a}{\to}_{1i}$. We also need to show that the implications (2) of Algorithm 2 hold in order to prove that $(s_0^i)_w \overset{a}{\to\!\!\!\!\!\!\sim}_i$. If $a \in A_{uo}$ then it must be shown that condition (2a) of Algorithm 2 holds true: $\forall u \in A_{uo}^*$: $(s_0^i)_{wa} \overset{u}{\to}_i \Rightarrow\ (s_0^i)_{wa} \overset{u}{\to}_{1i}$. Let $u \in A_{uo}^*$: $(s_0^i)_{wa} \overset{u}{\to}_i$. This is equivalent to $wau \in P_i^{-1}(L_i^2)$. Since $P(wau) = P(wa)$ and $wa \in K_i^2$ (which is shown above) we obtain from $G-$normality that $wau \in K_i^2$, i.e. $(s_0^i)_{wa} \overset{u}{\to}_{1i}$. If $a \in A_o$ then it must be shown that condition (2b) of Algorithm 2 holds true: $\forall s' \approx_{Aux(S_i)} (s_0^i)_w :\ s' \overset{a}{\to}_i \Rightarrow\ s' \overset{a}{\to}_{1i}$, in which case also $\forall u \in A_{uo}^*$: $s'_a \overset{u}{\to}_i \Rightarrow\ s'_a \overset{u}{\to}_{1i}$. Let $s' \approx_{Aux(S_i)} (s_0^i)_w :\ s' \overset{a}{\to}_i$. Since S_i is a state-partition automaton, according to Lemma 3.1 there exists $w' \in A^*$ such that $P(w') = P(w)$ and $s' = (s_0^i)_{w'}$. Hence, $s' \overset{a}{\to}_i$ is equivalent to $w'a \in P_i^{-1}(L_i^2)$. Recall again that $wa \in K_i^2$ and notice that $P(w'a) = P(wa)$. An application of $G-$normality yields $w'a \in K_i^2$, i.e. $s' \overset{a}{\to}_{1i}$. The rest is the same as for $a \in A_{uo}$: if $u \in A_{uo}^*$ such that $s'_a \overset{u}{\to}_i$, then $w'au \in P_i^{-1}(L_i^2)$ and $w'a \in K_i^2$ yield by $G-$normality $w'au \in K_i^2$, i.e. $s'_a \overset{u}{\to}_{1i}$. We conclude that $(s_0^i)_w \overset{a}{\to\!\!\!\!\!\!\sim}_i$, which was to be shown.
(iii) Assume that $\cap_{i=1}^n \tilde{l}_i(s_0^i)_w \overset{a}{\to}$. We know that for any

$i \in \mathbb{Z}_n$ we have $[\tilde{l}_i(s_0^i)]_w \xrightarrow{a}$, i.e. $(s_0^i)_w \xrightarrow{a}{}_{\prime i}$. We must show that $[\tilde{l}(s_0)]_w \xrightarrow{a}$ for $a \in A$, i.e. $(s_0)_w \xrightarrow{a}{}_1$ and condition (1) of Algorithm 1 applied to automata S and T is satisfied. Firstly, according to step 2. of Algorithm 1 we must show that $(s_0)_w \xrightarrow{a}{}_1$. It follows from $\forall i \in \mathbb{Z}_n$ $(s_0^i)_w \xrightarrow{a}{}_{\prime i}$ that in particular $(s_0^i)_w \xrightarrow{a}{}_i$ and $(s_0^i)_w \xrightarrow{a}{}_{1i}$. These claims are equivalent to $\forall i \in \mathbb{Z}_n$: $wa \in P_i^{-1}(L_i)^2$ and $wa \in K_i^2$. Thus, we have $wa \in L^2 = \bigcap_{i=1}^n P_i^{-1}(L_i)^2$ and $wa \in (K \cap L)^2 = \bigcap_{i=1}^n K_i^2$. The last two statements are equivalent to $(s_0)_w \xrightarrow{a}$ and $(s_0)_w \xrightarrow{a}{}_1$. Secondly, we must show that condition (1) of Algorithm 1 holds true: if $a \in A_{uo}$ then $\forall u \in A_{uo}^*$: $(s_0)_{wa} \xrightarrow{u} \Rightarrow (s_0)_{wa} \xrightarrow{u}{}_1$; and if $a \in A_o$ then $\forall s' \approx_{Aux(S)} (s_0)_w : s' \xrightarrow{a} \Rightarrow s' \xrightarrow{a}{}_1$, in which case also $\forall u \in A_{uo}^*$: $s'_a \xrightarrow{u} \Rightarrow s'_a \xrightarrow{u}{}_1$. Let $a \in A_{uo}$ and $u \in A_{uo}^*$: $(s_0)_{wa} \xrightarrow{u}$. Hence, we have $wau \in L^2$, i.e. $\forall i \in \mathbb{Z}_n$: $wau \in P_i^{-1}(L_i)^2$. The last statement is equivalent to $\forall i \in \mathbb{Z}_n$: $(s_0^i)_{wa} \xrightarrow{u}{}_i$. Our assumption that $\forall i \in \mathbb{Z}_n$: $(s_0^i)_w \xrightarrow{a}{}_{\prime i}$ implies according to condition (2a) of Algorithm 2 that $(s_0^i)_{wa} \xrightarrow{u}{}_i$ implies $(s_0^i)_{wa} \xrightarrow{u}{}_{1i}$, which is equivalent to $wau \in K_i^2$. Hence $wau \in K^2$, which together with $wau \in L^2$ yields $wau \in (K \cap L)^2$, which is equivalent to $(s_0)_{wa} \xrightarrow{u}{}_1$.

Now, let $a \in A_o$ and $s' \approx_{Aux(S)} (s_0)_w : s' \xrightarrow{a}$. According to condition (1b) of Algorithm 1 we must show that $s' \xrightarrow{a}{}_1$ and $\forall u \in A_{uo}^*$: $s'_a \xrightarrow{u} \Rightarrow s'_a \xrightarrow{u}{}_1$. It follows from Lemma 3.1 and $s' \approx_{Aux(S)} (s_0)_w$ that there exists $w' \in A^*$ such that $P(w') = P(w)$ and $s' = (s_0)_{w'} \in S$. Hence, $s'_a \xrightarrow{u}$ is equivalent to $w'a \in L^2$. An application of Lemma 3.1 to S_i yields also $(s_0^i)_{w'} \approx_{Aux(S_i)} (s_0^i)_w$. We notice that $w'a \in L^2 \subseteq P_i^{-1}(L_i)^2$. Therefore $(s_0^i)_{w'} \xrightarrow{a}$, i.e. according to condition (2b) of Algorithm 2 $(s_0^i)_{w'} \xrightarrow{a}{}_{1i}$. The last statement is equivalent to $w'a \in K_i^2$. Therefore $w'a \in \cap_{i=1}^n K_i^2 = (K \cap L)^2$. But this is equivalent to $\xrightarrow{a}{}_1$. The rest is similar to the case $a \in A_{uo}$: if $u \in A_{uo}^*$: $s'_a \xrightarrow{u}$, then $w'au \in L^2$ with the same w' as above. Thus, $\forall i \in \mathbb{Z}_n$: $w'au \in P_i^{-1}(L_i)^2$. The last statement is equivalent to $\forall i \in \mathbb{Z}_n$: $(s_0^i)_{w'a} \xrightarrow{u}{}_i$. Our assumption that $\forall i \in \mathbb{Z}_n$: $(s_0^i)_w \xrightarrow{a}{}_{\prime i}$ implies according to the second part of condition (2b) of Algorithm 2 that $(s_0^i)_{w'a} \xrightarrow{u}{}_i$ (playing the role of $s'_a \xrightarrow{u}{}_i$) implies $(s_0^i)_{w'a} \xrightarrow{u}{}_{1i}$, which is equivalent to $w'au \in K_i^2$. Hence $w'au \in (K \cap L)^2$, which is equivalent to $s'_a \xrightarrow{u}{}_1$. The conclusion is $(s_0)_w \xrightarrow{a}{}_{\prime}$, which is equivalent to $[\tilde{l}(s_0)]_w \xrightarrow{a}$. ■

Remark 3.6: It is easy to see that (iii) of the above proof (corresponding to the inclusion $\bigcap_{i=1}^n \sup \mathrm{N}(K_i, P_i^{-1}(L_i), P_i^{loc}, P) \subseteq \sup \mathrm{N}(K \cap L, L, P)$) has been proven without the assumption of $G-$normality. Otherwise stated, this inclusion is true in a very general setting.

G-normality is a very strong condition which may hold only in very special cases. Indeed, G-normality of the specification implies that $K_i = supN(K_i, P_i^{-1}(L_i), P)$. Since $K = \cap_{i=1}^n K_i$ this means that the result of the last theorem is reduced to

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

i.e. K is $(L, P)-$normal. Otherwise stated, the last theorem is not so effective. However, it is useful for a low complexity test for normality of indecomposable specification languages. Modular verification of normality is indeed very important. An effective result is presented in the sequel of this paper using a structural condition that does not depend on K.

In the forthcoming theorem $G-$normality is replaced by a structural condition called global mutual normality. It is similar to mutual normality in the case of decomposable specification ([5]), but concerns $P_i^{-1}(L_i)$ instead of L_i themselves.

Definition 3.5: (Global mutual normality) The local plant (partial) languages $L_i \subseteq (A_i^* \times A_i^*),\ i \in \mathbb{Z}_n$ are called globally mutually normal if for any $i \neq j \in \mathbb{Z}_n$ we have

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

We obtain the following result:

Theorem 3.7: (Sufficient structural conditions) Assume that $L_i,\ i \in \mathbb{Z}_n$ are globally mutually normal and $\forall i \neq j \in \{1, \ldots, n\}$ we have $A_{o,i} \cap A_j = A_i \cap A_{o,j}$. Then $\sup \mathrm{N}(K \cap L, L, P) =$

$$= \bigcap_{i=1}^n \sup \mathrm{N}(K_i, P_i^{-1}(L_i), P).$$

Proof: The proof is very similar to that of the last theorem. The main difference is the use of global mutual normality instead of $G-$normality. First of all, according to Remark 3.6 only (ii) of the proof above is different. Thus, we only need to show that (ii) of bisimulation relations corresponding to (the more difficult) inclusion holds true. Consider again the relation R from the proof of Theorem 3.5. Let $[\tilde{l}(s_0)]_w \xrightarrow{a}$ for $a \in A$, i.e. $(s_0)_w \xrightarrow{a}{}_{\prime}$. Thus, according to step 2 of Algorithm 1 $(s_0)_w \xrightarrow{a}{}_1$ and condition (1) of Algorithm 1 is satisfied. It must be shown that $[\cap_{i=1}^n \tilde{l}_i(s_0^i)]_w \xrightarrow{a}$. We need show that for any $i \in \mathbb{Z}_n$ we have $[\tilde{l}_i(s_0^i)]_w \xrightarrow{a}$, i.e. $(s_0^i)_w \xrightarrow{a}{}_{\prime i}$. According to Algorithm 2 this amounts to show that for any $i \in \mathbb{Z}_n$ we have $(s_0^i)_w \xrightarrow{a}{}_{1i}$ and condition (2) of Algorithm 2 holds. It has been shown in the proof of Theorem 3.5 that $(s_0^i)_w \xrightarrow{a}{}_{1i}$ without any use of $G-$normality. Next we show that condition (2) of Algorithm 2 is satisfied. If $a \in A_{uo}$ then according to (2a) of Algorithm 2 it must be checked that $\forall v \in A_{uo}^*$: $(s_0^i)_{wa} \xrightarrow{v}{}_i \Rightarrow (s_0^i)_{wa} \xrightarrow{v}{}_{1i}$.
Let $v \in A_{uo}^*$: $(s_0^i)_{wa} \xrightarrow{v}{}_i$. This is equivalent to $wav \in P_i^{-1}L_i^2$. Recall that $(s_0^i)_w \xrightarrow{a}{}_{1i} s_a$, which is equivalent to $wa \in K_i^2$. Now we use global mutual normality. Since $P(wav) = P(wa)$ and $wa \in L^2 \subseteq P_j^{-1}L_j^2$ for any $j \neq i \in \mathbb{Z}_n$, an application of global mutual normality

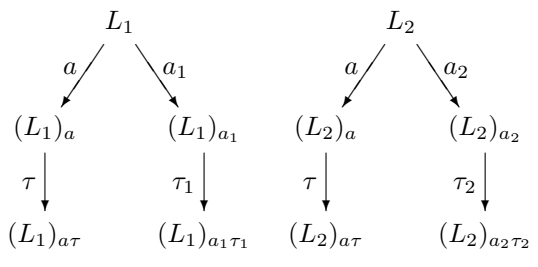
yields $wav \in (P^{-1}PP_j^{-1})(L_j^2) \cap P_i^{-1}L_i^2 \subseteq P_j^{-1}L_j^2$. Hence, $wav \in \bigcap_{i=1}^n P_i^{-1}L_i^2 = L^2$, which is equivalent to $(s_0)_{wa} \xrightarrow{v}$. Using the assumption $(s_0)_w \xrightarrow{a}{}_!$, we obtain according to condition (1a) of Algorithm 1 that $(s_0)_{wa} \xrightarrow{v}_1$, which is equivalent to $wav \in (K \cap L)^2$. Therefore $wav \in K_i^2 = K^2 \cap P_i^{-1}L_i^2$, which shows that $(s_0^i)_{wa} \xrightarrow{v}_{1i}$.

If $a \in A_o$ then according to condition (2b) of Algorithm 2 it must be checked that $\forall s' \approx_{Aux(S_i)} (s_0^i)_w : \; s' \xrightarrow{a}_i \Rightarrow s' \xrightarrow{a}_{1i}$, in which case also $\forall v \in A_{uo}^*$: $s'_a \xrightarrow{v}_i \Rightarrow \; s'_a \xrightarrow{v}_{1i}$. Let $s' \approx_{Aux(S_i)} (s_0^i)_w : \; s' \xrightarrow{a}_i$. According to Lemma 3.1 there exists $r \in A^*$ such that $P(w) = P(r)$ and $s' = (s_0^i)_r$. Thus, $s' \xrightarrow{a}_i$ is equivalent to $ra \in P_i^{-1}L_i^2$. Recall that $wa \in K_i^2$ and $P(wa) = P(ra)$. Hence for any $j \neq i \in \mathbb{Z}_n$ we obtain using global mutual normality that $ra \in (P^{-1}PP_j^{-1})(L_j^2) \cap P_i^{-1}L_i^2 \subseteq P_j^{-1}L_j^2$. Note that again $ra \in (P^{-1}PP_j^{-1})(L_j^2)$, because $wa \in L^2 \subseteq P_j^{-1}L_j^2$ and $P(wa) = P(ra)$. Thus, we have $ra \in L^2$, which is equivalent to $(s_0)_r \xrightarrow{a}$. Since by Lemma 3.1 $(s_0)_r \approx_{Aux(S)} (s_0)_w$, condition (1b) of Algorithm 1 (applied to $(s_0)_r$ playing the role of s') implies that $(s_0)_r \xrightarrow{a}_1$, which is equivalent to $ra \in (K \cap L)^2 \subseteq K_i^2$, i.e. $s' \xrightarrow{a}_{1i}$. The rest is the same as for $a \in A_{uo}$: second part of condition (2b) of Algorithm 2 is similar to condition (2a) of Algorithm 2. We conclude that $(s_0^i)_w \xrightarrow{a}{}_{!i}$ for any $i \in \mathbb{Z}_n$, i.e. $[\cap_{i=1}^n \tilde{l}_i(s_0^i)]_w \xrightarrow{a}$. ■

The last theorem is very useful, because we have found structural conditions under which supremal $(L,P)-$normal sublanguages can be computed without manipulating with the global plant L, but using only $P_i^{-1}L_i$ for $i \in \mathbb{Z}_n$. Hence the combinatorial explosion in terms of n (number of modules) is avoided.

Now we present an example, where it is shown that global mutual normality (GMN) is not a necessary condition. This should not be surprising, because GMN condition does not depend on the specification.

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 local plant languages, where only second (prefix-closed) components are considered:



Let $K^2 = \{\varepsilon, a, a_1, a_1a_2\}$. One can easily verify that K is not decomposable. Indeed, the inclusion $K^2 \subset P_1^{-1}P_1(K^2) \cap P_2^{-1}P_2(K^2)$ is strict, i.e. $K^2 \neq P_1^{-1}P_1(K^2) \cap P_2^{-1}P_2(K^2)$. Computing further parallel product $L = L_1 \parallel L_2$ yields:

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

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 $\sup \mathrm{N}(K_i, P_i^{-1}(L_i), P) = \{\varepsilon\}$ 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.

$$\tau_1 \in (P^{-1}PP_1^{-1}(L_1^2) \cap (P_2)^{-1}(L_2^2) \setminus P_1^{-1}(L_1^2).$$

IV. Conclusion

New methods for modular computation of supremal normal sublanguages of indecomposable specification languages have been presented. The structural condition of global mutual normality does not depend on the particular specification, which is very important because general indecomposable specifications are studied.

On the other hand further research is needed in order to improve our results: the sufficient conditions we have obtained might be weakened at least in special cases. Another open direction is a possible extension of the concept of partial controllability to our setting of modular control with partial observations. Such a notion of partial normality may prove to be useful in a future refinement of our results.

References

[1] S.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, 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. Computation of Supremal Sublanguages of Supervisory Control Using Coalgebra. Proceedings *WODES'02*, Zaragoza, pp. 26-33, October 2-4, 2002.

[4] Jan Komenda and Jan 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. Proceedings *WODES'04*, pp. 73-78, Reims, September 22-24, 2004.

[6] J.J.M.M. Rutten. Coalgebra, Concurrency, and Control. *Research Report CWI*, SEN-R9921, Amsterdam, November 1999. Available also at `http://www.cwi.nl/~janr`.

[7] J.J.M.M. Rutten. Universal Coalgebra: A Theory of Systems. *Theoretical Computer Science* 249:3-80, 2000.

[8] 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.*

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

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

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

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

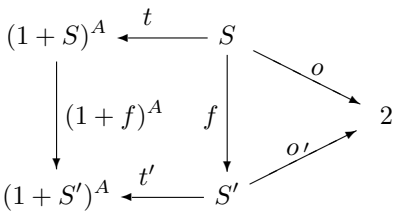
APPENDIX

A. Partial automata

Partial automata as generators of DES are formulated coalgebraically as in [6]. Let A be the set of events. The empty string will be denoted by ε. Denote by $1 = \{\emptyset\}$ 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 $\langle o, t\rangle : S \to 2 \times (1+S)^A$, consists of an output function $o : S \to 2$ and a transition function $t : S \to (1 + S)^A$. 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 $t(s) : A \to (1+S)$. The set $1+S$ is the disjoint union of S and 1. 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 $s \in S$. $t(s)(a) \in S$ 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, we define $s \xrightarrow{wa}$ 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 $f : S \to S'$ 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: $f(s)_a = f(s_a)$.



A partial automaton $S' = (S', \langle o', t'\rangle)$ is a *subautomaton* of $S = (S, \langle o, t\rangle)$ if $S' \subseteq S$ and the inclusion function $i : S' \to S$ is a homomorphism.

A *bisimulation* between two partial automata $S = (S, \langle o, t\rangle)$ and $S' = (S', \langle o', t'\rangle)$ is a relation $R \subseteq S \times S'$ such that: if $\langle s, s'\rangle \in R$ then
(i) $o(s) = o(s')$, i.e. $s\downarrow$ iff $s'\downarrow$
(ii) $\forall a \in A : s \xrightarrow{a}\Rightarrow (s' \xrightarrow{a}$ and $\langle s_a, s'_a\rangle \in R)$,
(iii) $\forall a \in A : s' \xrightarrow{a}\Rightarrow (s \xrightarrow{a}$ 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.

There is the following simple characterization of bisimilarity on a partial automaton S.

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

B. Final automaton of partial languages

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,

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

The transition function $t_{\mathcal{L}} : \mathcal{L} \to (1 + \mathcal{L})^A$ is defined using input derivatives. Recall that for any partial language $L = (L^1, L^2) \in \mathcal{L}$, $L_a = (L^1_a, L^2_a)$, where $L^i_a = \{w \in A^* \mid aw \in L^i\}$, $i = 1, 2$. 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:

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

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

Notice that if L_a is defined, then $L^1_a \subseteq L^2_a$, $L^2_a \neq \emptyset$, and L^2_a 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 $w \in L^2$.

Recall from [6] that $\mathcal{L} = (\mathcal{L}, \langle o_{\mathcal{L}}, t_{\mathcal{L}}\rangle)$ is final among all partial automata: for any partial automaton $S = (S, \langle o, t\rangle)$ there exists a unique homomorphism $l : S \to \mathcal{L}$. This homomorphism identifies bisimilar states: for $s, s' \in S$: $l(s) = l(s')$ iff $s \sim s'$.

Another characterization of finality of $\mathcal{L}$ is that it satisfies the principle of coinduction: for all K and L in $\mathcal{L}$, if $K \sim L$ then $K = L$. Since the converse implication holds trivially true this means that bisimilarity coincides with equality in the automaton of partial languages. This is a general property of final coalgebras that enables proofs by coinduction. In order to show that two elements of a final coalgebra (e.g. two partial languages) are equal, it is sufficient to construct a bisimulation relation that relates them.

Recall that the unique homomorphism l given by finality of $\mathcal{L}$ maps a state $s \in S$ to $l(s) = (L^1_s, L^2_s) = (\{w \in A^* \mid s \xrightarrow{w}$ and $s_w\downarrow\}, \{w \in A^* \mid s \xrightarrow{w}\}) \in \mathcal{L}$.

Inclusion of partial languages that corresponds to a simulation relation (a relation that we obtain by dropping condition (iii) in the definition of bisimulation relation) is meant componentwise.