MODULAR ANTIPERMISSIVE CONTROL OF DISCRETE-EVENT SYSTEMS

Jan Komenda [*,1] **Jan H. van Schuppen** [**]

[*] *Math. Inst. of Czech Academy of Sciences, Brno Branch, Zizkova 22, 616 62 Brno, Czech Republic*
[**] *CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands*

Abstract: The paper concerns modular supervisory control of discrete-event systems (DES), where the overall system is composed of subsystems that are combined in a synchronous (parallel) product. Recently it has been shown that under very general conditions closed-loop languages under permissive control policy (introduced by coinduction as a new operation called supervised product) distribute with the synchronous (parallel) product. This paper is focused on the study of distributivity between synchronous product and closed-loop languages with respect to antipermissive control policy. It is guaranteed that under the conditions derived in the paper antipermissive control synthesis can be done locally and the local antipermissive control synthesis yields the same solution as the global control synthesis. An example illustrates that antipermissive control synthesis yields in general larger observable sublanguages than the supremal normal sublanguages are and that the mutual observability as a structural condition is not necessary for distributivity of closed-loop languages under antipermissive control policy with the synchronous product. *Copyright © 2005 IFAC*

Keywords: Discrete-event systems, Modular supervisory control, Observability, Coalgebra, Antipermissive control policy

1. INTRODUCTION

The modular approach to supervisory control of DES modeled by finite automata has been introduced in (Ramadge and Wonham, 1989) and further developped in e.g. (Willner and Heymann, 1991), and (Wong and Lee, 2002). The system is composed of local components (subsystems) that run concurrently (in parallel), i.e. the global system is a synchronous product of local components. Very little attention has been paid so far to the modular control with partial observations. It has been recently studied in (Komenda and Van Schuppen, 04), where computation of supremal normal sublanguages for modular DES has been considered. The main result of this paper is the formulation of sufficient conditions that guarantee that closed-loop languages under antipermissive control policy distribute with the synchronous product. This question is of a major interest, because we ensure that antipermissive control synthesis can be done locally.

In the next section basic facts about coalgebra and coinduction are recalled. Section 3 is devoted to basic concepts needed in this paper. Closed-loop languages under antipermissive control policies

[1] The research was supported by the EU Esprit LTR Project Control and Computation, ISO-2001-33520 and the Academy of Sciences of the Czech Republic, Institutional Research Plan No. AV0Z10190503.

are computed according to the algorithm developed specially for this purpose in Section 4. The main result of this paper, i.e. structural conditions under which closed-loop languages generated by antipermissive control policy distribute with the synchronous product, is presented in Section 5.

2. COALGEBRA AND COINDUCTION

It has been discovered recently that state transition systems, i.e. in particular various type of automata are coalgebras. Coalgebras are categorical duals of algebras (the corresponding functor operates from a given set rather than to a given set). The basic introduction to the theory of universal coalgebra is developed in analogy with the corresponding theory of universal algebra in (Rutten 2000). The concept of final coalgebras enables definitions and proofs by coinduction.

2.1 Partial automata

In this section partial automata as generators of DES are formulated coalgebraically as in (Rutten 1999). Final coalgebra of partial automata, i.e. a partial automaton of partial languages is then recalled. 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 $s \xrightarrow{a} s_a$.

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

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

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.

2.2 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 (Rutten 1999) 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 $l : S \to \mathcal{L}$. 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 $K = L$. Recall that the unique homomorphism l given by finality of $\mathcal{L}$ maps a state $s \in S$ to the partial language $l(s) = (L^1_s, L^2_s) = (\{w \in A^* \mid s \xrightarrow{w} \text{ and } s_w\downarrow\}, \{w \in A^* \mid s \xrightarrow{w}\})$.

2.3 Coinductive definitions

Recall from (Rutten 1999) the following coinductive definition of the synchronous product. For 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 $L\downarrow$.

Many other binary operations on partial languages have been defined by coinduction in (Komenda and Van Schuppen, 2003). We recall the concept of weak transition from (Komenda and Van Schuppen, 2003). Assume that $A = A_o \cup A_{uo}$ is a partition of A into observable events (A_o)

and unobservable events (A_{uo}) with the natural projection $P : A^* \to A_o^*$ which erases unobservable events.

Definition 2.2. (Nondeterministic weak transition) For $a \in A$ define $L \overset{P(a)}{\Rightarrow}$ if $\exists s \in A^* \ : \ P(s) = P(a)$ and $L \xrightarrow{s} L_s$. Denote in this case $L \overset{P(a)}{\Rightarrow} L_s$.

3. MODULAR CONTROL WITH PARTIAL OBSERVATIONS

Let us consider the concurrent behavior of local subplants (partial automata) $S_1, \dots, S_n$. Assume that local alphabets of these subplants, $A_i,\ i \in Z_n = \{1, \dots, n\}$ composed of uncontrollable events (A_{iu}) and controllable events (A_{ic}) are such that $A_i = A_{iu} \cup A_{ic},\ i \in Z_n$. First we assume that $\forall j \neq i \in Z_n :\ \ A_{iu} \cap A_j = A_i \cap A_{ju}$. This assumption originally introduced in (Wong and Lee, 2002) means that the events shared by two local subsystems must have the same control status for both controllers associated to these subsystems. Denote $A_c = \cup_{i=1}^n A_{ic}$ and $A_u = \cup_{i=1}^n A_{iu}$. We then still have the disjoint union $A = A_c \cup A_u$ due to the assumption that $A_{iu} \cap A_j = A_i \cap A_{ju}$. 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.

Moreover 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 $A_o = \cup_{i=1}^n A_{o,i} \subseteq A = \cup_{i=1}^n A_i$. Some additional notation is needed to set up our framework. Globally unobservable events are denoted by $A_{uo} = A \setminus A_o$. Partial observations in individual modules are expressed via local projections $P_i^{loc} : A_i^* \to A_{o,i}^*$, while global projection is denoted by $P : A^* \to A_o^*$. 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 from now on that $n = 2$ and that the global plant L and specification K languages are decomposable into local plant and local specification languages: $L = L_1 \parallel L_2$ and $K = K_1 \parallel K_2$. Recall that the parallel product of (ordinary) languages $L_i \subseteq A_i^*,\ i = 1, 2$ is defined by $L_1 \parallel L_2 = P_1^{-1}(L_1) \cap P_2^{-1}(L_2)$.

An auxiliary concept that reflects the fact that due to partial observations it is not possible to distinguish between states is needed. The following relation captures partial observability in a way that is suitable for our coalgebraic treatment.

Definition 3.1. (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) If $\langle s, t \rangle \in Aux(S)$ then $\forall a \in A :\ \ (s \overset{P(a)}{\Rightarrow} s'$ for some s' and $t \overset{P(a)}{\Rightarrow} t'$ for some t' $) \quad \Rightarrow \langle s', t' \rangle \in Aux(S)$

For $\langle s, s' \rangle \in Aux(S)$ the notation $s' \approx_{Aux(S)} s$ is used. $Aux(S)$ can be characterized by the following lemma (Komenda and Van Schuppen, 2003).

Lemma 1. 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 $s' = (s_0)_{w'}$.

The concept of an observer automaton has been formulated in (Cassandras and Lafortune, 1999). A partial automaton is said to be a state-partition automaton if different states of the associated observer automaton (as subsets of the state set) do not overlap.

4. ANTIPERMISSIVE CONTROL POLICY

In this section we formulate an algorithm for computation of closed-loop languages under antipermissive control policy. This algorithm will then be used for deriving conditions under which these languages are preserved by modular (local) control synthesis. First we recall from (Komenda and Van Schuppen, 2003) the definition of the antipermissive control policy. It is defined with respect to the partial automata $S = (S, \langle o, t \rangle)$ representing the plant and $S_1 = (S_1, \langle o_1, t_1 \rangle)$ representing the specification that we want to impose by a supervisory controller. As usual, we assume that S_1 is a subautomaton of S. The (common) initial state of S and S_1 is denoted by s_0. The transition function t_1 of S_1 is denoted by $\to_1$ and the transition function t of S is denoted by $\to$.

Denote by V the supervisor associated with a partial automaton S, and the corresponding antipermissive control policy by $\gamma_A(V, :) :\ P(l(s_0)^2) \to \Gamma$, where Γ is the class of enabled events, also called control patterns (i.e. supersets of the event subset A_u that are always enabled). Algebraically, the antipermissive control policy is defined as follows:

$$\gamma_A(V, s) = A_{uc} \cup \{a \in A_c :\ \forall s' \in K^2 \cap P^{-1}P(s) \text{ we have } (s'a \in L^2 \ \Rightarrow\ s'a \in K^2)\}. \tag{1}$$

Similarly as for the permissive control policy the supervisor marks all states that have been marked in the plant and survive under supervision. Using the concept of observational indistinguishability relation we can formulate the following auxiliary algorithm for the closed-loop languages under antipermissive control policy.

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 $\tilde{S} = (\tilde{S}, \langle \tilde{o}, \tilde{t} \rangle)$ with $\tilde{t}$ denoted by $\rightarrow'$.
Define the auxiliary condition (*) as follows:
$a \in A_u$ or $a \in A_c$ and $\forall s' \approx_{Aux(S_1)} s$: $s' \xrightarrow{a} \Rightarrow s' \xrightarrow{a}_1$

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 $\tilde{o}(s) = o(s)$.

We denote by $\tilde{l} : \tilde{S} \to \mathcal{L}$ the unique (behavior) homomorphism given by finality of $\mathcal{L}$. Let us introduce the notation $AP(K, L, P)$ for the pair of closed-loop languages (marked and prefix-closed) under antipermissive control policy. Then we have:

Theorem 4.1. $\tilde{l}(s_0)$ is the tuple of closed-loop languages under the antipermissive control policy defined by equation (1) above, i.e. $\tilde{l}(s_0) = AP(K, L, P)$.

Proof. It is sufficient to notice from the definition of the antipermissive control policy that for $v \in \tilde{l}(s_0)^2$: $\tilde{l}(s_0)_v \xrightarrow{a}$ iff $(s_0)_v \xrightarrow{a}'$, iff $a \in \gamma_A(V, P(v))$, which is equivalent to $va \in AP(K, L, P)$, i.e. $AP(K, L, P)_v \xrightarrow{a}$, i.e. $\tilde{l}(s_0)$ and $AP(K, L, P)$ are bisimilar, thus equal by the coinduction proof principle. Indeed, according to Algorithm 1 $(s_0)_v \xrightarrow{a}'$ iff the condition (*) is fulfilled, which case happens iff $a \in \Gamma_A(V, P(v))$ from the definition of $\Gamma_A(V, .)$ due to Lemma 1: $P(v) = P(v')$ iff $\langle (s_0)_v, (s_0)_{v'} \rangle \in Aux(S_1)$, because S_1 is a state-partition automaton.

Remark 4.2. Algorithm 1 looks almost like a coinductive definition of the closed-loop languages under antipermissive control policy, which is not possible to do directly in $\mathcal{L}$. Thus Algorithm 1 is suitable for investigating our main problem:

Problem 4.3. When do the closed-loop languages under antipermissive control policy distribute with the synchronous product of partial languages?

The problem 4.3 is solved in Theorem 5.1.

5. DISTRIBUTIVITY

The main result of this paper is the formulation of conditions under which closed-loop languages under antipermissive control policy distribute with the synchronous product of partial languages. These conditions also indicate when it is possible to make antipermissive control synthesis with partial observations locally. Some auxiliary definitions and results are needed to formulate our main theorem. Closed-loop languages under global and local antipermissive control policies are computed according to Algorithm 1, specially designed for this purpose in Section 4. In our main theorem a condition similar to mutual controllability (Wong and Lee, 2002) is used. We call it by analogy mutual observability.

Definition 5.1. 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 observable if
1) $\forall s, s' \in L_i^2$ and $a \in A_{c,i}$: $(sa \in P_i(P_j)^{-1}(L_j^2)$ and $P_i^{loc}(s) = P_i^{loc}(s')$ and $s'a \in L_i^2) \Rightarrow sa \in L_i^2$.
2) $\forall s, s' \in L_j^2$ and $a \in A_{c,j}$: $(sa \in P_j(P_i)^{-1}(L_i^2)$ and $P_j^{loc}(s) = P_j^{loc}(s')$ and $s'a \in L_j^2) \Rightarrow sa \in L_j^2$.

The following lemmas concerning the relation between global and local observations are needed.

Lemma 2. 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 $i = 1, 2$: $P_i^{loc} P_i(s) = P_i^{loc} P_i(s')$.

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 $P(v) = P(v')$.

Our main theorem follows.

Theorem 5.1. If $A_{o,2} \cap A_1 = A_2 \cap A_{o,1}$ and L_1 and L_2 are mutually observable, then

$$AP(K_1, L_1, P_1^{loc}) \parallel AP(K_2, L_2, P_2^{loc}) = \qquad (2)$$

$$AP(K_1 \parallel K_2, L_1 \parallel L_2, P).$$

Proof. The coinductive proof principle will be used. Closed-loop languages under local and global antipermissive control policies are computed according to Algorithm 1. The notation is as follows: let S representing K and T representing L be such that S is a subautomaton of T and S is a state-partition automaton. Algorithm 1 yields partial automaton $\tilde{S} = \langle \tilde{o}, \tilde{t} \rangle$ with $\tilde{t}$ denoted by $\rightarrow'$ and its behavior by $\tilde{l} : \tilde{S} \to \mathcal{L}$.

Similarly, for $i \in \{1, 2\}$, S_i and T_i representing K_i and L_i, respectively, are such that S_i is a subautomaton of T_i and S_i is a state-partition automaton. Construction of Algorithm 1 applied to local modules yields partial automaton $\tilde{S}_i = (\tilde{S}_i, \langle \tilde{o}_i, \tilde{t}_i \rangle)$ with $\tilde{t}_i$ denoted by $\rightarrow'$ and its behavior by $\tilde{l}_i : \tilde{S} \rightarrow \mathcal{L}$.

The (common) initial state of S and T is denoted by s_0 and for $i = 1, 2$ the (common) initial states of S_i and T_i are denoted by s_0^i. The transition function of S_i and S is denoted by $\rightarrow_1$ and the transition function of T_i and T is denoted by $\rightarrow$. Therefore, according to Theorem 4.1 $\tilde{l}(s_0) = AP(K, L, P)$ and $\tilde{l}_i(s_0^i) = AP(K_i, L_i, P_i^{loc})$ for $i = 1, 2$, where $K = K_1 \parallel K_2$ and $L = L_1 \parallel L_2$.

We show that

$$R = \{\langle [\tilde{l}(s_0)]_v, [\tilde{l}_1(s_0^1) \parallel \tilde{l}_2(s_0^2)]_v \rangle \mid v \in (\tilde{l}(s_0))^2\}$$

is a bisimulation relation, from where the claim of the theorem follows by coinduction. Take a $v \in (\tilde{l}(s_0))^2$ arbitrary, but fixed.
(i) is obvious, Algorithm 1 does not affect the marking components.
(ii) Let $[\tilde{l}(s_0)]_v \xrightarrow{a}$, i.e. condition (*) of Algorithm 1 is satisfied for $s = [(s_0)]_v$. It must be shown that $[\tilde{l}_1(s_0^1) \parallel \tilde{l}_2(s_0^2)]_v \xrightarrow{a}$.
First we assume that $a \in A_1 \cap A_2$. Then $[\tilde{l}_1(s_0^1) \parallel \tilde{l}_2(s_0^2)]_v = [\tilde{l}_1(s_0^1)]_{v_1} \parallel [\tilde{l}_2(s_0^2)]_{v_2}$ with $P_1(v) = v_1$ and $P_2(v) = v_2$. We show that $[\tilde{l}_1(s_0^1)]_{v_1} \xrightarrow{a}$, i.e. $(s_0^1)_{v_1} \xrightarrow{a}'$. According to Algorithm 1 applied to S_1 and T_1 we must show that condition (*) holds. First of all note that $(s_0^1)_{v_1} \xrightarrow{a}_1$. Indeed, $[\tilde{l}(s_0)]_v \xrightarrow{a}$ implies that $(s_0)_v \xrightarrow{a}_1$, i.e. $va \in K^2 = (K_1 \parallel K_2)^2$. Therefore $v_1 a = P_1(va) \in K_1^2 \subseteq L_1^2$, i.e. $(s_0^1)_{v_1} \xrightarrow{a}_1$. Similarly $v_2 a = P_2(va) \in K_2^2 \subseteq L_2^2$.

Let $a \in A_{c,1} \subseteq A_c$ then we know that $\forall s' \approx_{Aux(S)} (s_0)_v : s' \xrightarrow{a} \Rightarrow s' \xrightarrow{a}_1$. It must be shown that $(s_0^1)_{v_1} \xrightarrow{a}'$, i.e. $\forall q^1 \approx_{Aux(S_1)} (s_0^1)_{v_1} : q^1 \xrightarrow{a} \Rightarrow q^1 \xrightarrow{a}_1$. Let $q^1 \approx_{Aux(S_1)} (s_0^1)_v : q^1 \xrightarrow{a}$. Since S_1 is a state-partition automaton, there exists $v_1' \in A_1^*$ such that $P_1^{loc}(v_1') = P_1^{loc}(v_1)$ and $q^1 = (s_0^1)_{v_1'}$. For this $v_1' \in A_1^*$ and $v \in A^*$ above there exists according to Lemma 3 $v' \in A^*$ such that $P_1(v') = v_1'$ satisfying moreover $P_2^{loc} P_2(v') = P_2^{loc}(v_2)$ and $P(v) = P(v')$. Since $q^1 \xrightarrow{a}_1$ we have $v_1' a \in K_1^2 \subseteq L_1^2$. Then $v'a \in P_1^{-1}(v_1' a)$. Therefore $v'a \in P_1^{-1}(L_1^2)$. We show that $v'a \in P_2^{-1}(L_2^2)$ using mutual observability. Notice that $P_2(v') = v_2'$ satisfies $P_2^{loc}(v_2') = P_2^{loc}(v_2)$ and $v_2 a \in K_2^2 \subseteq L_2^2$. Also $v_2' a \in P_2(P_1)^{-1}(L_1^2)$, because $v_1' a \in L_1^2$. Therefore $v_2' a \in L_2^2$ by applying mutual observability. Thus, $v'a \in P_1^{-1}(L_1^2) \cap P_2^{-1}(L_2^2) = L^2$. Since $P(v') = P(v)$, we have $(s_0)_{v'} \approx_{Aux(S)} (s_0)_v$. From $(s_0)_v \xrightarrow{a}'$ and condition (*) of Algorithm 1, it follows that $(s_0)_{v'} \xrightarrow{a} \Rightarrow (s_0)_{v'} \xrightarrow{a}_1$. But this implies that $(s_0^1)_{v_1'} = q^1 \xrightarrow{a}_1$, because $v'a \in K^2 = (K_1 \parallel K_2)^2$ implies that $v_1' a = P_1(v'a) \in K_1^2$.
In a symmetric way $[\tilde{l}_2(s_0^2)]_{v_2} \xrightarrow{a}$, i.e. $[\tilde{l}_1(s_0^1) \parallel \tilde{l}_2(s_0^2)]_v \xrightarrow{a}$.
The cases $a \in A_1 \setminus A_2$ and $a \in A_2 \setminus A_1$ are simpler. We need to show that $[\tilde{l}_1(s_0^1)]_{v_1} \xrightarrow{a}$. The proof is very similar as above, but it is much simpler due to $P_2(a) = \varepsilon$, i.e. in order to show e.g. $va \in P_2^{-1}(L_2^2)$ for $a \in A_1 \setminus A_2$ it is sufficient to show $P_2(va) = v_2 \in L_2^2$, which is trivially true.
(iii) Let $[\tilde{l}_1(s_0^1) \parallel \tilde{l}_2(s_0^2)]_v \xrightarrow{a}$. It must be shown that $[\tilde{l}(s_0)]_v \xrightarrow{a}$, i.e. condition (*) of Algorithm 1 is satisfied. According to the coinductive definition of synchronized product inductively applied we have: $[\tilde{l}_1(s_0^1) \parallel \tilde{l}_2(s_0^2)]_v = \tilde{l}_1(s_0^1)_{v_1} \parallel \tilde{l}_2(s_0^2)_{v_2}$ with $P_1(v) = v_1$ and $P_2(v) = v_2$. It follows that $\tilde{l}_1(s_0^1)_{v_1} \xrightarrow{a}$ and $l_2(s_0^2)_{v_2} \xrightarrow{a}$, i.e. $(s_0^1)_{v_1} \xrightarrow{a}'$ and $(s_0^2)_{v_2} \xrightarrow{a}'$. It must be shown that $[\tilde{l}(s_0)]_v \xrightarrow{a}$, which is equivalent to $(s_0)_v \xrightarrow{a}'$, i.e. condition (*) of Algorithm 1 applied to (global) automata S and T is satisfied.

We know that if $a \in A_{c,1}$ then $\forall s' \approx_{Aux(S_1)} (s_0^1)_{v_1} : s' \xrightarrow{a} \Rightarrow s' \xrightarrow{a}_1$.

We know also that if $a \in A_{c,2}$ then $\forall s' \approx_{Aux(S_2)} (s_0^2)_{v_2} : s' \xrightarrow{a} \Rightarrow s' \xrightarrow{a}_1$.

We need to show that if $a \in A_c$ then $\forall s' \approx_{Aux(S)} (s_0)_v : s' \xrightarrow{a} \Rightarrow s' \xrightarrow{a}_1$.

First we assume that $a \in A_1 \cap A_2$. Let $a \in A_c$ and $s' \approx_{Aux(S)} (s_0)_v : s' \xrightarrow{a}$. Since S is a state-partition automaton, there exists $v' \in K^2 \subseteq L^2 : P(v') = P(v)$ and $s' = (s_0)_{v'}$. Thus $s' \xrightarrow{a}$ is equivalent to $v'a \in L^2$. Therefore $P_i(v'a) := v_i' a \in L_i^2$. Using Lemma 2 we have $P_i^{loc}(v_i') = P_i^{loc}(v_i)$, $i = 1, 2$, i.e. $s_i' := (s_0^i)_{v_i'} \approx_{Aux(S_i)} (s_0^i)_{v_i}$, where $s_i' \xrightarrow{a}$, $i = 1, 2$. Hence according to our assumption $s_i' \xrightarrow{a}$ $i = 1, 2$ implies that $s_i' \xrightarrow{a}_1$ $i = 1, 2$. This means that $P_i(v'a) = v_i' a \in K_i^2$, $i = 1, 2$, hence $v'a \in P_1^{-1}(K_1^2) \cap P_2^{-1}(K_2^2) = K^2$, which is equivalent to $s' \xrightarrow{a}_1$. This proves that $[\tilde{l}(s_0)]_v \xrightarrow{a}$ for $a \in A_1 \cap A_2$.
If $a \in A_1 \setminus A_2$, then we only have $\tilde{l}_1(s_0^1)_{v_1} \xrightarrow{a}$, i.e. condition (*) is satisfied for S_1 subautomaton T_1: if $a \in A_{c,1}$ then $\forall s' \approx_{Aux(S_1)} (s_0^1)_{v_1} : s' \xrightarrow{a} \Rightarrow s' \xrightarrow{a}_1$. Nevertheless, this is still sufficient to prove that the condition (*) is satisfied for S subautomaton T, because $a \notin A_2$, which means that $v'a \in P_2^{-1}(K_2^2)$ iff $v' \in P_2^{-1}(K_2^2)$.

Remark 5.2. The interest of this theorem should be clear: under the conditions that are stated it is possible to perform the antipermissive control synthesis with partial observations locally, which represents an exponential save on the computational complexity and makes in fact the antipermissive control synthesis of some large distributed plants feasible. Note that an extension of all the results in this paper from 2 to $n \in N$ local modules is easy. Condition of mutual observability between any pair of local plants is then required.

Notice also that for (iii) in the above proof no assumption is used (except $A_{o,2} \cap A_1 = A_2 \cap A_{o,1}$ that is needed for Lemma 2). This means that under very general conditions we have one inclusion meaning that global antipermissive control synthesis yields in general a larger language that the local antipermissive control synthesis.

Corollary 1. If $A_{o,2} \cap A_1 = A_2 \cap A_{o,1}$, then

$$AP(K_1, L_1, P_1^{loc}) \parallel AP(K_2, L_2, P_2^{loc}) \subseteq \qquad (3)$$

$$AP(K_1 \parallel K_2, L_1 \parallel L_2, P)2, P).$$

Next an example is given, where the distributivity between the closed-loop languages under antipermissive control policy and the synchronous product holds without the mutual observability condition. Therefore mutual observability is not a necessary condition for the distributivity. It is related to the fact that mutual observability as a structural condition concerns only open-loop languages and not specification languages.

Example 1. Let $A = A_c = \{a, a_1, a_2, \tau, \tau_1, \tau_2\}$, $A_1 = A_{c,1} = \{a_1, \tau_1, a, \tau\}$, $A_2 = A_{c,2} = \{a_2, \tau_2, a, \tau\}$, $A_o = \{a_1, a_2, a\}$, $A_{o,1} = \{a_1, a\}$, and $A_{o,2} = \{a_2, a\}$. Consider the following local specification and plant languages, where only second (prefix-closed) components are considered:

$$\begin{tikzcd}
 & K_1 \arrow[dl, "a"'] \arrow[dr, "a_1"] & & & L_1 \arrow[dl, "a"'] \arrow[dr, "a_1"] & \\
(K_1)_a & & (K_1)_{a_1} & (L_1)_a \arrow[d, "\tau"'] & & (L_1)_{a_1} \arrow[d, "\tau_1"'] \\
 & & & (L_1)_{a\tau} & & (L_1)_{a_1\tau_1}
\end{tikzcd}$$

and

$$\begin{tikzcd}
 & K_2 \arrow[dl, "a"'] \arrow[dr, "a_2"] & & & L_2 \arrow[dl, "a"'] \arrow[dr, "a_2"] & \\
(K_2)_a \arrow[d, "\tau"'] & & (K_2)_{a_2} & (L_2)_a \arrow[d, "\tau"'] & & (L_2)_{a_2} \arrow[d, "\tau_2"'] \\
(K_2)_{a\tau} & & & (L_2)_{a\tau} & & (L_2)_{a_2\tau_2}
\end{tikzcd}$$

According to the local and global antipermissive control policies we obtain $AP(K_1, L_1, P_1^{loc}) = K_1$ and similarly $AP(K_2, L_2, P_2^{loc}) = K_2$. After computing parallel products $K = K_1 \parallel K_2$ and $L = L_1 \parallel L_2$, it is easy to check that $AP(K, L, P) = K$. Thus, we have

$$K = AP(K, L, P) =$$

$$AP(K_1, L_1, P_1^{loc}) \parallel AP(K_2, L_2, P_2^{loc}),$$

i.e. the distributivity holds true. On the other hand, mutual observability does not hold: for $s = a_1\tau_1$ and $s' = a_1$ we have $s, s' \in L_1^2$, $s'\tau_1 \in L_1^2$, $P_1^{loc}(s') = P_1^{loc}(s)$, $\tau_1 \in A_{c,1}$ and $s\tau_1 = a_1\tau_1\tau_1 \in P_1(P_2)^{-1}(\varepsilon) \subseteq P_1(P_2)^{-1}(L_2^2)$, but $s\tau_1 \notin L_1^2$.

We conclude that mutual observability is not necessary in Theorem 5.1. Verification of mutual observability can be reduced to the verification of observability with respect to new modified plants $P_1(P_2)^{-1}(L_2^2)$ and $P_2(P_1)^{-1}(L_1^2)$. This is easier than checking observability of large plants.

6. CONCLUSION

We have studied modular supervisory control with partially observed modules in the coalgebraic framework. Our result is important for feasibility of the antipermissive supervisory control of large distributed plants (when antipermissive control synthesis can be executed locally).

There are open problems for a future investigation: our conditions are only sufficient conditions that might be weakened in special cases.

REFERENCES

Cassandras, S.G. and Lafortune, S. (1999). *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, Dordrecht 1999.

Komenda, J. and Van Schuppen, J.H. (2003). Coalgebra and coinduction in discrete-event control, submitted. Also available as CWI report.

Komenda, J. and Van Schuppen, J.H. (2004). Supremal Normal Sublanguages of Large Distributed Discrete-Event Systems. *Proceedings Workshop on Discrete-Event Systems*, 73-78.

Rutten, J.J.M.M. (1999). Coalgebra, Concurrency, and Control. *Research Report CWI*, SEN-R9921 (`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. and Lee, S. (2002). Structural Decentralized Control of Concurrent Discrete-Event Systems. *European J. 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.