Decentralized Supervisory Control of Discrete-Event Systems over Communication Networks

A. Mannani and P. Gohari

Abstract

In this paper we investigate the problem of designing embedded decentralized discrete-event controllers over communication networks. It is assumed that there is a path between every pair of processes in the network. The control objective is specified by a prefix-closed language which is controllable and observable, but not coobservabe. The paper is focused on communication among processes necessary to meet the control objective. As such, process models are left unspecified; it is only required that disabling any of the controllable events does not block communication among processes. Our findings support the idea that in the presence of ideal communication channels the protocol design for non-coobservable specifications can be reduced to the synthesis of communicating decentralized supervisors, and we propose solutions for a restricted class of problems. Also a couple of positive results are stated for the case where channels are unreliable.

Index Terms

decentralized supervisory control, discrete-event systems, protocol design.

I. Introduction

The synthesis problem for Discrete-Event Systems (DES) asks for the design of a controller so that the system under control satisfies the specification of some desired behavior. In RW framework [1] violations of a *controllable* and *observable* specification can always be prohibited by designing a centralized supervisor which limits the occurrence of some controllable

The authors are with the ECE Dept., Concordia University, Montreal, QC, Canada, H3G 1M8, amin_man,gohari@ece.concordia.ca.

events in the plant language. It was later shown in [2] that for distributed plants a set of non-communicating *decentralized* supervisors, each partially observing the plant's behavior, can be designed to confine this behavior within the given global specification behavior if and only if such a specification is both controllable and *coobservable*.

A common example of a distributed DES is a communication network in which processes exchange among themselves data messages under an ordering specified by a set of rules, known as a communication protocol [3]. While the use of formal methods has significantly contributed to specifying the desired behavior of systems and validation techniques [4], the control community has considered the automation of the synthesis problem. In [5] the example of Alternating Bit Protocol (ABP) was first introduced as a solution to a decentralized supervisory control problem. However, since both plant components (i.e. sender and receiver) and the specification models "spelled out" the solution, the synthesis was ad hoc. In a later formulation in [6], the specification does not contain the solution and requires only a linear ordering among some events. It is shown that inclusion of ABP in the sender model makes the specification coobservable with respect to the plant and thus there exists a set of non-communicating decentralized supervisors which yield ABP. Conversely, in the absence of this inclusion coobservability does not hold and such a set does not exist.

When coobservability fails it may still be possible to design decentralized supervisors by allowing communication among them. In fact, our findings support the idea that such problems yield protocols which require communication of some control- or observation-related information among the supervisors. In our formulation of the problem, assuming ideal communication channels, the protocol design for a special class of non-coobservable specifications, including ABP, is reduced to the Synthesis of Communicating Decentralized Supervisors (SCDS).

SCDS is motivated by the control of networks [7]. In [8] a necessary and sufficient condition for solving the control problem is formulated by a refinement relation between observation and control maps. "Minimal communication" among supervisors is studied in the general framework of information structures in [9]. The problem is further investigated in [10] based on the latest safe point for communication, and in the "knowledge" framework of [11] based on as-early-as-possible communication, where in both approaches state estimates are communicated. Finally, the algorithm in [12] is proved to communicate a minimal number of "events" between two agents. Another aspect of SCDS is communication delay studied in [13] where a characterization of communication networks into an infinite hierarchy of problems is presented.

Our perspective to study SCDS relies on the formalism of Extended Finite State Machines (EFSMs) in which bits of control information necessary in the process of decision-making, rather than events or state estimates, are communicated from one supervisor to another. An EFSM implements supervisory control [1] by employing boolean variables to encode the supervisor's states, a set of boolean functions to observe events, and boolean formulas to control transitions [14]. This formalism was extended in [15] to the decentralized case by assigning a set of private variables to each component EFSM to make decision-making possible at local sites. The decision as to whether to enable or disable a local event may in general depend on the values of supervisor's own private variables, and the local copies of variables owned by other supervisors. These copies are updated by communication among local supervisors. It is shown that the dependence of updating functions on copy variables is related to a modified version of "joint observability" [16]. Solutions are developed for a special class of problems where the sole purpose of communication is control [15].

In this paper we apply the above results to protocol design for non-coobservable specifications. This approach requires neither the plant components nor the specification to "spell out" the protocol (i.e. part or all of the protocol need not be designed and included in the transition structures of plant components or specification beforehand). In the first part of the paper, we define Discrete-Event Control over Communication Networks (DECCN) problem and present its partial solution under the assumption of ideal channels. Thereby we complete the previous works on ABP, a practical benchmark and an illustrative example, by showing that the protocol arises naturally as a solution to the corresponding control problem with no *a priori* inclusion of the solution in the plant model or the specification language. We then extend this result to other special classes of protocol design problems for ideal channels. Moreover, we comment on the difficulties of tackling unreliable channels and present some positive results.

The rest of the paper is organized as follows: Section II reviews the basics of EFSM formalism and Section III states the general problem. Section IV then formulates the ABP problem in the EFSM framework and synthesizes ABP as a solution to this problem under the assumption of ideal channels. Section V discusses ways in which the problem can be generalized for ideal channels and the case of unreliable communication channels are discussed in Section VI. Conclusions are drawn and suggestions for future research are made in Section VII.

II. Extended finite-state machines

Notation: In this paper we assume that all recognizers are deterministic. We denote a recognizer and its generated (closed) language by bold and regular capital letters, respectively.

In an EFSM a transition is equipped with a guard formula, and when it is taken it triggers a number of updating functions. A set X of boolean variables is introduced. A transition in the EFSM is enabled if and only if its *guard formula*, which is a predicate defined as a boolean formula over X, evaluates to *true* (1). When a transition is taken, $|X|$ *updating actions* may follow. An updating action is a boolean function that assigns a new value to a variable based on the old values of all variables. Given the set X, in the following definition let $k = |X|$, $\mathcal{G}$ denote the set of all boolean formulas over X, and $\mathcal{A}$ denote the set of all boolean functions $b : \mathbb{B}^k \to \mathbb{B}$.

Definition 1 [14] An EFSM $\mathbf{L_x}$ is defined as a 7-tuple $\mathbf{L_x} = (Q, \Sigma, \xi, q_0, X, g, a)$, where $\mathbf{L} = (Q, \Sigma, \xi, q_0)$ is a FSM in which Q is a finite set of states; Σ is a finite alphabet; $\xi : Q \times \Sigma \to Q$ is a partial transition function; q_0 is the initial state; X is a finite set of boolean variables; $g : \Sigma \to \mathcal{G}$ assigns to each event a *guard formula*; and $a : X \times \Sigma \to \mathcal{A}$ assigns to each pair of event and variable an *updating function*. When $\mathbf{L}$ is understood from the context, $\mathbf{L_x}$ is simply written as $\mathbf{L_x} = (-, X, g, a)$. □

Assume that all variables are initialized to *false* (0). We extend ξ to $Q \times \Sigma^*$ in the usual way. For $\alpha \in \Sigma$, the guard formula $g(\alpha)$ is a boolean formula with which all transitions labeled with α are guarded. For $\alpha \in \Sigma$ and $x \in X$, the updating function $a(x, \alpha) : \mathbb{B}^k \to \mathbb{B}$ is a boolean function. When α is taken, it results in the assignment $x := a(x, \alpha)(\underline{v})$, where the vector $\underline{v}$ represents the current values of variables in X.

Let $V : \Sigma^* \to \mathbb{B}^k$ be a map that assigns to every string $s \in \Sigma^*$ a tuple of boolean values obtained from recursively applying the updating functions of events in s to $\underline{0}$, that is:

$$V(s) = \big(v(s, x)\big)_{x \in X} \tag{1}$$

where for $s \in \Sigma^*$, $\sigma \in \Sigma$ and $x \in X$, the function $v : \Sigma^* \times X \to \mathbb{B}$ is recursively defined as $v(\epsilon, x) := 0$ and $v(s\sigma, x) := a(x, \sigma)(V(s))$. The closed language of $\mathbf{L_x}$, denoted by L_x, contains a string generated by $\mathbf{L_x}$ if guard formulas are respected at all its prefixes, i.e.:

$$\epsilon \in L_x \text{ and } s \in L_x \wedge \xi(q_0, s\sigma)! \wedge g(\sigma)(V(s)) = 1 \Leftrightarrow s\sigma \in L_x. \tag{2}$$

By virtue of having a control mechanism embedded in their structure, EFSMs can be used to model closed-loop systems. It is shown in [14] that when the control action of a centralized supervisor is encoded by plant components' EFSMs, the language of the synchronous product of the EFSMs is equal to the language of the system under supervision.

III. Problem statement

Fix an index set $I = \{1, \ldots, n\}$ and consider a system $\mathcal{N}$ consisting of n communicating parallel processes $\mathbf{P_{1x}}, \ldots, \mathbf{P_{nx}}$ which are connected through a strongly connected network of potentially unreliable channels in which data may be lost or delayed. Accordingly, an ideal channel is defined to be one in which data is instantly transmitted without any losses. We refer to the set of rules governing the exchange of data among these processes as *communication protocol* or in short *protocol* [3]. For brevity we write $\mathbf{P_{ix}} \rightarrow \mathbf{P_{jx}}$ when there is a potentially unreliable channel from $\mathbf{P_{ix}}$ to $\mathbf{P_{jx}}$. Fig. 1 shows the network topology for the case when $n = 4$. Each process $\mathbf{P_{ix}}$ is modeled by an EFSM, to which we assign sets $\Sigma_{o,i}$, $\Sigma_{uo,i}$, and $\Sigma_{c,i} \subseteq \Sigma_{o,i} \cup \Sigma_{uo,i}$ of respectively observable, unobservable, and controllable events by the process. Each β_{ij} label, $i, j \in I, i \neq j$, represents a *set* of communication-related events between two processes $\mathbf{P_{ix}}$ and $\mathbf{P_{jx}}$, each of which can be exclusively observed by the two processes (see the following subsection).

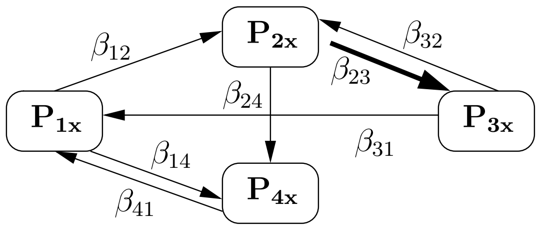


Fig. 1. A network of n communicating parallel processes ($n = 4$). Each process $\mathbf{P_{ix}}$ has a number of observable, unobservable and communication-related events. A process may be connected to others through ideal (bold arrows) or potentially unreliable (regular arrows) channels.

A. Processes

Each process $\mathbf{P_{ix}}$ is modeled by an EFSM $\mathbf{P_{ix}} = (Q_i, \Sigma_i, \xi_i, q_{0i}, X_i, g_i, a_i)$, where $(i \in I)$

- $\Sigma_i = \Sigma_{o,i} \dot\cup \Sigma_{uo,i} \dot\cup \{\beta^s_{ij} |\ \mathbf{P_{ix}} \to \mathbf{P_{jx}}\} \dot\cup \{\beta^e_{ji}, \beta^r_{ji} | \mathbf{P_{jx}} \to \mathbf{P_{ix}}\}$;
- $X_i = X_{ii} \dot\cup X_{ci}$ where X_{ii} is the set of private variables of process i whose kth ($k \in \mathbb{N}$) element is denoted by x^k_{ii} (k is removed when X_{ii} is a singleton), and $X_{ci} = \bigcup_{j \in I, j \neq i} X_{ij}$, where X_{ij} stores copies of process $\mathbf{P_{jx}}$'s private variables, $j \in I,\ j \neq i$. A copy of the kth private variable of process $j,\ j \neq i$, which is stored in X_{ij}, is denoted by x^k_{ij}. All sets are finite;
- Guards and updating functions are to be designed from the centralized supervisor, except for the following "updates" which are fixed *a priori:*
  - When $\mathbf{P_{jx}} \to \mathbf{P_{ix}}$, $a_i(X_{ik}, \beta^r_{ji}) = X_{jk}$, $k \in I,\ k \neq i$, which is an abbreviation for $\forall l \in \mathbb{N}.\ a_i(x^l_{ik}, \beta^r_{ji}) = x^l_{jk}$. Similarly, write $X_{ik} := X_{jk}$ when $\forall l \in \mathbb{N}.\ x^l_{ik} := x^l_{jk}$.

The alphabet of process $\mathbf{P_{ix}}$ includes its observable and unobservable events in $\Sigma_{o,i}$ and $\Sigma_{uo,i}$, communication events β^s_{ij} for each process $\mathbf{P_{jx}}$ to which process $\mathbf{P_{ix}}$ sends communication through a potentially unreliable channel, and two events β^e_{ji} and β^r_{ji} for each process $\mathbf{P_{jx}}$ from which process $\mathbf{P_{ix}}$ receives communication through an unreliable channel (erroneous and error-free, respectively). The set X_i consists of variables in the set X_{ii} which are *private* to $\mathbf{P_{ix}}$, and sets of variables X_{ij}, $j \neq i$, which are used to store copies of private variables of processes $\mathbf{P_{jx}}$ (i.e. X_{jj}). When a communication from $\mathbf{P_{jx}}$ is received error-free (event β^r_{ji}) all local copies in X_{ci} are updated with the values of the corresponding variables in $\mathbf{P_{jx}}$, that is, $\forall k \neq i.\ X_{ik} := X_{jk}$. This guarantees that process $\mathbf{P_{ix}}$ is updated with the values of the private variables of $\mathbf{P_{jx}}$, i.e. $\forall l.\ x^l_{ij} := x^l_{jj}$. Moreover, by updating other local copies in $X_{ik},\ k \neq j$, with the corresponding values of the variables in $\mathbf{P_{jx}}$, one can insure that local copies are updated even when no *direct* connection between a pair of processes exists. For instance, if process $\mathbf{P_{3x}}$ communicates to process $\mathbf{P_{1x}}$ only through process $\mathbf{P_{2x}}$, then variables in X_{13} are updated with the values of variables in X_{33} after communication events β^r_{32} and β^r_{21} occur in sequence: β^r_{32} results in the assignment $X_{23} := X_{33}$, and subsequently β^r_{21} updates $X_{13} := X_{23} (= X_{33})$.

We impose no restriction on the structure of $\mathbf{P_{ix}}$ except that when an event in $\Sigma_{c,i}$ is disabled by a protocol none of the communication events β^s_{ij}, β^e_{ji} or β^r_{ji} are affected. This restriction insures that the assumption that the network is strongly connected always remains valid.

Notation: In what follows we let $\Xi = \bigcup_{i\in I}\Sigma_i$, $\Sigma = \bigcup_{i\in I}\left(\Sigma_{o,i} \cup \Sigma_{uo,i}\right)$, $\Sigma_o = \bigcup_{i\in I}\Sigma_{o,i}$, $\Sigma_{uo} = \Sigma \setminus \Sigma_o$, and $I_o(\sigma) = \{i \in I \mid \sigma \in \Sigma_{o,i}\}$. Define the natural projections $\pi : \Xi^* \longrightarrow \Sigma^*$ to erase the communication-related (β) events, $\pi_o : \Sigma^* \longrightarrow \Sigma_o^*$ to erase unobservable events, and $\pi_i : \Sigma^* \longrightarrow \Sigma_{o,i}^*$ to specify the observation window of process i. Also let L and E be respectively the plant and specification languages such that $L \subseteq \Xi^*$ and $E \subseteq \pi(L) \subseteq \Sigma^*$.

B. Channels

A process $\mathbf{P_{ix}}$ may communicate to process $\mathbf{P_{jx}}$ through a communication channel C_{ij} whenever it exists. Channels may be *ideal* or *unreliable*. In our diagrams, unreliable and ideal channels are denoted by regular and bold arrows, respectively.

1) Unreliable channels: In practice channels can be potentially unreliable: data could get lost or corrupted, and communication delay cannot be ignored. When $\mathbf{P_{ix}} \rightarrow \mathbf{P_{jx}}$, the unreliable channel C_{ij} is modeled as depicted in Fig. 2.

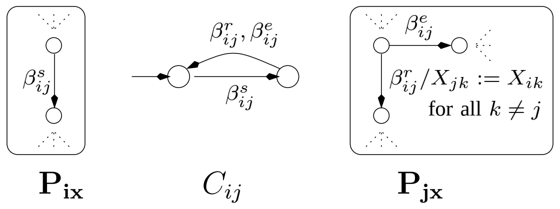


Fig. 2. An unreliable channel.

When event β_{ij}^s occurs in $\mathbf{P_{ix}}$ the values of all variables in X_i are transmitted to the channel C_{ij}. *Eventually* the message is delivered to $\mathbf{P_{jx}}$. We assume that each process has perfect error-detection facilities. If the message delivered by the channel is erroneous (event β_{ij}^e), communication has failed and process $\mathbf{P_{jx}}$ might just change state. If the communication is successful (event β_{ij}^r), then $\mathbf{P_{jx}}$ updates all but its own private variables with the values of $\mathbf{P_{ix}}$'s variables received from the channel.

2) Ideal channels: A channel is ideal when it is free from any communication loss or delay. While the assumption of ideality lets one focus on the "logical" aspects of the control problem, it is also valid in communication networks where communication delay is negligible compared to the processing time at each site. In an ideal network, each process has instant access to all variables of all other processes which it needs for reevaluating its guard and updating functions.

Focusing on ideal channels enables us to find out *what* needs to be communicated in order to achieve the control objective, without worrying about the logistics of such communication, which will be dealt with in Section VI.

C. Control problem

Assumption 1 We assume that the desired behavior of the network is specified by a prefix-closed language E which is controllable with respect to $\pi(L)$ and observable with respect to $\pi(L)$ and π_o. Therefore, there always exists a centralized supervisor, say $\mathbf{S} = (R, \Sigma, \eta, r_0)$, which enforces E, i.e. $\pi(S||L) = E$ [17]. Note that events in Σ_{uo} may appear only as selfloops in $\mathbf{S}$ and are left out from our transition diagrams.

The control objective is then to design a controller for each process such that the natural projection of the language $P_{1x}||P_{2x}||\dots||P_{nx}$ onto Σ is equal to E. Notice that since the control map is embedded in each process model, implementing the centralized control map reduces to finding suitable guard formulas and updating functions for each process.

Definition 2 Discrete-Event Control over Communication Networks (DECCN): Let $\mathcal{N}$ be a system consisting of n communicating parallel processes $\mathbf{P_{1x}}, \dots, \mathbf{P_{nx}}$, each modeled by an EFSM as in Subsection III-A, which are connected through a strongly connected network of potentially unreliable channels, and let E and S be respectively the languages of specification and its enforcing supervisor as described in Assumption 1. Design guard formulas and updating functions for each process such that $P_{1x}||P_{2x}||\dots||P_{nx} = S||L$. $\square$

In the next two sections we focus on the logics of control implementation by assuming that channels are ideal, while in Section VI we study the problem when channels are unreliable.

IV. DECCN SOLUTION—SPECIAL CASE

In this section we present a solution to a subclass of DECCN problems under the assumptions that a) communication channels are ideal, b) for each $i \in I$, $\Sigma_{o,i} = \Sigma_{c,i} = \{\alpha_i\}$ are singleton and $\Sigma_{uo,i} = \emptyset$, where α_i is called the "significant" event of process i, and c) occurrence of each significant event is counted modulo 2.

Note that under assumption a) variables in X_{ci} are identical to the private variables of other processes, which justifies using x^k_{jj} instead of x^k_{ij} when needed ($j \neq i$). The controllability and observability of significant events make E controllable and observable, and thus $\mathbf{E}$ may be used as the centralized supervisor in this section. Following the simplifying assumptions b) and c) we use Alternating Bit Protocol (ABP) as a running example, and partially design the protocol as the solution to the corresponding DECCN problem. This simplification leads us to a key observation of the solution approach for general DECCN problems in the next section. The protocol design will be complete in Section VI after the assumption of ideal channels is lifted.

Since the significant event of process i, denoted by α_i, needs to be counted modulo 2, X_{ii} reduces to a singleton, whose only variable x_{ii} is toggled each time α_i occurs:

$$a_i(x_{ii}, \alpha_i) = \overline{x}_{ii} \tag{3}$$

With the updating functions fixed, a solution to DECCN consists of finding guard formulas $g_i(\alpha_i)$, for each $i \in I$.

A. ABP: Problem formulation in the EFSM framework

Alternating Bit Protocol (ABP) [18], [19] is used for reliable transmission of files over half-duplex channels. As shown in Fig. 3, two processes $\mathbf{P_1}$ and $\mathbf{P_2}$ communicate over a channel $\mathbf{ch}$. Process $\mathbf{P_1}$ fetches a message and sends it to the channel. Then process $\mathbf{P_2}$ receives the message from the channel, and accepts it if it is error-free. The control objective requires that every message fetched by $\mathbf{P_1}$ be accepted by $\mathbf{P_2}$ exactly once. When a transmission error occurs, $\mathbf{P_1}$ should resend its message until it is received error-free and is accepted by $\mathbf{P_2}$.

A schematic of the plant is shown in Fig. 4, where a transmission error is denoted by a broken arrow. The system events are defined in Table I. Figure 5 shows FSM models for sender $\mathbf{P_1}$, receiver $\mathbf{P_2}$ and channel $\mathbf{ch}$ as well as the specification E of the desired behavior defined as an ordering of events in $\{\alpha_1, \alpha_2\}$. The following is a short description of each FSM in Fig. 5.

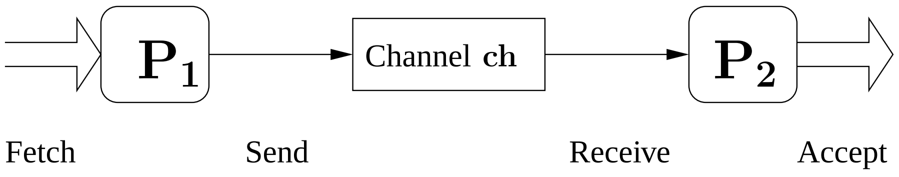


Fig. 3. Two processes $\mathbf{P_1}$ and $\mathbf{P_2}$ communicating over a channel.

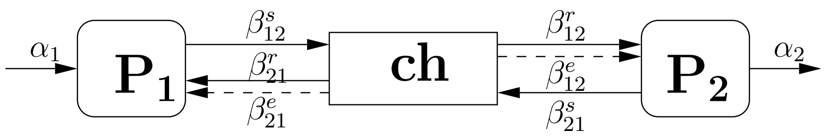


Fig. 4. Schematic of the plant.

TABLE I
SYSTEM EVENTS.

Event	Description	Event	Description
α_1	data fetched by $\mathbf{P_1}$	α_2	data accepted by $\mathbf{P_2}$
β_{12}^s	data sent by $\mathbf{P_1}$	β_{21}^s	acknowledgement sent by $\mathbf{P_2}$
β_{12}^r	data received by $\mathbf{P_2}$	β_{21}^r	acknowledgement received by $\mathbf{P_1}$
β_{12}^e	data received by $\mathbf{P_2}$ erroneous	β_{21}^e	acknowledgement received by $\mathbf{P_1}$ erroneous

1) **Sender $\mathbf{P_1}$**. At the initial state, sender $\mathbf{P_1}$ nondeterministically does one of the following:
    a) It sends a data message to the channel (message could be empty if nothing is yet fetched).
    b) It fetches a data message and sends it to the channel.

   After receiving acknowledgement from the channel (possibly erroneous), sender $\mathbf{P_1}$ returns to its initial state.
2) **Channel $\mathbf{ch}$**. Any type of message received by the channel from one party (data β_{12}^s or acknowledgement β_{21}^s) will be delivered to the other party (β_{12}^r or β_{21}^r, respectively), or it will get lost or corrupted (β_{12}^e or β_{21}^e, respectively). Note that $\mathbf{ch}$ is the composition of C_{12} and C_{21}, as defined in the previous section.
3) **Receiver $\mathbf{P_2}$**. After receiving a data message from the channel (possibly erroneous), receiver $\mathbf{P_2}$ nondeterministically does one of the following:
    a) It sends an acknowledgement to the channel.
    b) It accepts the message, and sends an acknowledgement to the channel.

To make our models simpler we allow slightly more permissive behavior than that of an actual data transmission system. For example, we allow an empty message to be transmitted indefinitely.

It turns out that the plant in Fig. 5 violates the specification in two fundamental ways. The following two strings are accepted by the plant but not by the specification:

$$\alpha_{\mathbf{1}}; \beta_{12}^s; \beta_{12}^r; \alpha_{\mathbf{2}}; \beta_{21}^s; \beta_{21}^e; \beta_{12}^s; \beta_{12}^r; \alpha_{\mathbf{2}} \quad \text{and} \quad \alpha_{\mathbf{1}}; \beta_{12}^s; \beta_{12}^e; \beta_{21}^s; \beta_{21}^e; \alpha_{\mathbf{1}} \tag{4}$$

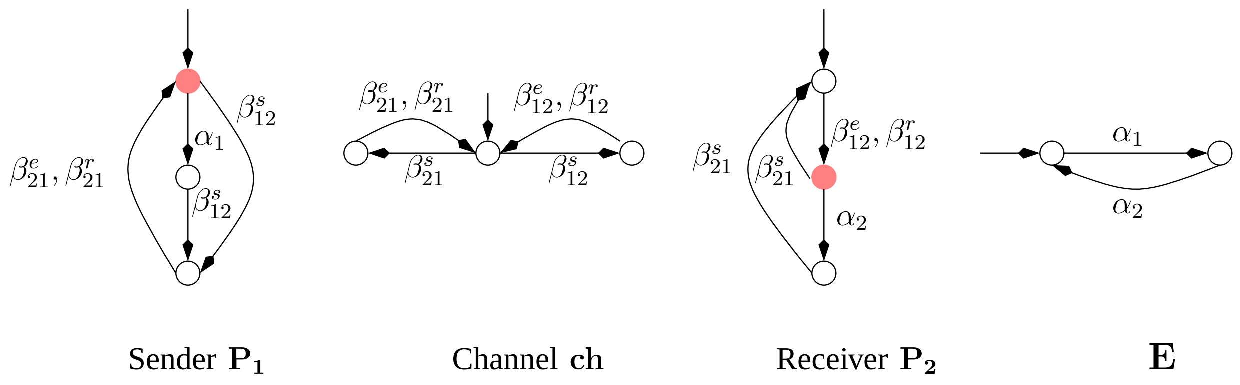


Fig. 5. Plant FSMs and the requirement specification. $\Sigma_1 = \{\alpha_1, \beta^s_{12}, \beta^e_{21}, \beta^r_{21}\}$ and $\Sigma_2 = \{\alpha_2, \beta^s_{21}, \beta^e_{12}, \beta^r_{12}\}$.

The well-known ABP [18], [19] provides a standard solution to this control problem. To find a solution in our framework we extend the two processes to $\mathbf{P_{ix}} = (-, X_i, g_i, a_i)$, $i = 1, 2$, where $X_i = \{x_{ii}, x_{ij}\}$ and updating functions are identity except $a_i(x_{ii}, \alpha_i) = \overline{x_{ii}}$. Note that the assumption of ideal channels allows us to use x_{jj} instead of x_{ij}. The control problem is to find guard formulas g_1 and g_2 such that the projection of $P_{1x}||P_{2x}$ onto $\{\alpha_1, \alpha_2\}$ is equal to E.

B. Solution

In supervisory control theory of DES [1], if a given specification is controllable and observable with respect to the plant, there always exists a centralized supervisor which enforces the legal language. In case of distributed DES where each agent has partial observation of the plant behavior, such a controllable global specification is enforceable if and only if it is coobservable with respect to the plant and agents' corresponding observational natural projections [2]. In simple words, coobservability requires that for every two observationally equivalent plant strings, and every event which extends one to a legal string while the other to an illegal string, there exists at least one agent which can disambiguate the strings and inhibit the illegal behavior. The set of decentralized supervisors synthesized in this case need not communicate amongst themselves.

Therefore, if the controllable global specification were coobservable, the solution to DECCN would simply be obtained by separately implementing the supervisory control maps [14] of the computed decentralized supervisors using only their private variables [15]. This case has been discussed in [6] using the FSMs of the plant components for the ABP example where the authors have shown that if the sender model is enriched by incorporating two events associated with the $0/1$ status of the ABP's attached bit, the specification will become coobservable with respect to the plant. The same can be said about other defined notions of coobservability with other fusion rules [20]. Thus, in this case no control information need be communicated over the network to implement the rules of data exchange (i.e. the protocol).

Unfortunately, the specification E in DECCN is *not* in general coobservable. For example, in ABP, $E = (\alpha_1\alpha_2)^*(\epsilon + \alpha_1)$ is not coobservable. To see why, let $s = \alpha_1\beta^s_{12}\beta^r_{12}$ and $s' = \beta^s_{12}\beta^r_{12}$. Note that α_2 is eligible to occur at both s and s', $s\alpha_2$ is legal while $s'\alpha_2$ is illegal, and finally $\pi_2(\pi(s)) = \pi_2(\pi(s'))$. Since process 2 is the only process that can disable α_2, E is not coobservable. In the rest of this paper we will show how a controllable and observable but non-coobservable specification may be satisfied by communicating information among local processes.

To begin with, we note that under the assumption of ideal channels, one can work with the variable set $X = \{x_{11}, x_{22}, \ldots, x_{nn}\}$. The function $V : \Sigma^* \to \mathbb{B}^n$ of Section II is $V(s) = \big(v(s,x)\big)_{x\in X}$, where:

$$\forall i \in I. \quad v(\epsilon, x_{ii}) = 0 \;\text{ and }\; v(s\alpha_i, x_{ii}) = \begin{cases} 1 & ; \; v(s,x_{ii}) = 0 \\ 0 & ; \; v(s,x_{ii}) = 1 \end{cases}.$$

It turns out that a solution can be found only for a restricted class of problems. Towards this end, let $\mathbf{E} = (R, \Sigma, \eta, r_0)$ be the *centralized supervisor*'s FSM, and $\mathcal{L}$ denote the set of all labeling maps $l : R \to pwr(\mathbb{B}^n)$. For $1 \le i \le n$ we write a member of $\mathbb{B}^n$ as $\underline{v} = (v_i, v_{-i})$, where v_i is the i^{th} element of the n-tuple $\underline{v}$, and v_{-i} denotes the $(n-1)$-tuple formed by the remaining elements of $\underline{v}$. Define a partial ordering $\preceq$ on $\mathcal{L}$ as follows:

$$\forall l_1, l_2 \in \mathcal{L}. \quad l_1 \preceq l_2 \iff \forall r \in R.\; l_1(r) \subseteq l_2(r) \tag{5}$$

It can be verified that $(\mathcal{L}, \preceq)$ is a complete lattice. Let ℓ be the smallest labeling map satisfying the following properties:

1) $\underline{0} \in \ell(r_0)$,

2) $\forall r, r' \in R, \alpha_i \in \Sigma, \underline{v} \in \mathbb{B}^n.\; \underline{v} \in \ell(r) \wedge r' = \eta(r, \alpha_i) \implies (\overline{v_i}, v_{-i}) \in \ell(r').$ (6)

The labeling map ℓ is chosen so that a transition labeled with α_i toggles the i^{th} element of each vector in the state's label. We show by induction that the label of a state reached by s includes the vector of values $V(s)$.

Lemma 1 We have $\forall s \in E, r \in R.\ r = \eta(r_0, s) \implies V(s) \in \ell(r)$.

Proof: We prove this lemma by induction on the length of s.

- Base: Let $s = \epsilon$. Then $r_0 = \eta(r_0, s)$, and by definition $V(s) = \underline{0} \in \ell(r_0)$.
- Inductive step: For $s \in \Sigma^*$ and $\alpha_i \in \Sigma$ let $s\alpha_i \in E$. Denote $r := \eta(r_0, s)$ and $r' := \eta(r_0, s\alpha_i)$. It follows from the induction assumption that $V(s) \in \ell(r)$. Let $V(s) := (v_i, v_{-i})$. We have:
$$V(s\alpha_i) = (\overline{v_i}, v_{-i}) \in \ell(r') \qquad \text{(by definition of } \ell\text{)} \qquad \blacksquare$$

Under certain conditions the labeling map ℓ can in effect encode the states of $\mathbf{E}$: knowing the current value $\underline{v} \in \mathbb{B}^n$ of boolean variables, it is possible to know which state r the centralized supervisor is in by checking whether $\underline{v} \in \ell(r)$, as long as $\underline{v}$ does not appear in the label of any other state. This idea is formalized in the following definition.

Definition 3 Let $\mathbf{E} = (R, \Sigma, \eta, r_0)$ be a centralized supervisor and $\ell : R \to pwr(\mathbb{B}^n)$ be as defined above. Then $\mathbf{E}$ is said to be *state-independent* with respect to ℓ if
$$\forall r, r' \in R.\ r \neq r' \implies \ell(r) \cap \ell(r') = \emptyset. \qquad \square$$

In other words, in a state-independent centralized supervisor the labels of a pair of distinct states are disjoint. When a centralized supervisor is state-independent, it is possible to uniquely determine its state by knowing the values assumed by the boolean variables after a legal string; in other words, the inverse of the implication in Lemma 1 is true as well.

Lemma 2 When $\mathbf{E}$ is state-independent with respect to ℓ we have:

$$\forall s \in E, r \in R.\ r = \eta(r_0, s) \iff V(s) \in \ell(r).$$

Proof ($\Leftarrow$): By contradiction assume for $s \in \Sigma^*$ and $r \in R$ that $V(s) \in \ell(r)$ but $\eta(r_0, s) = r'$ for some $r' \neq r$ in R. It follows from Lemma 1 that $V(s) \in \ell(r')$, contradicting the fact that $\mathbf{E}$ is state-independent. $\blacksquare$

The following result states that a solution to the control problem exists when the centralized supervisor is state-independent.

Theorem 3 Under the assumption that channels are ideal, DECCN has a solution if $\mathbf{E}$ is state-independent with respect to ℓ.

Proof: Let $\mathcal{L}_i = \bigcup_{r \in R \wedge \eta(r,\alpha_i)!} \ell(r)$ and $g_i(\alpha_i)$ be a boolean formula that is true for $\underline{v} \in \mathbb{B}^n$ iff $\underline{v} \in \mathcal{L}_i$. By induction we show that for all $s \in \Sigma^*$ we have $s \in \pi(P_{1x}||P_{2x}||\cdots||P_{nx})$ iff $s \in E$.

Base is trivial since $\mathbf{E}$ and all $\mathbf{P_{ix}}$ are nonempty. For the inductive step let $s\alpha_i \in \pi(P_{1x}||P_{2x}|| \cdots||P_{nx})$. Since all languages are prefix-closed it follows that $s \in \pi(P_{1x}||P_{2x}||\cdots||P_{nx})$ and hence by the induction assumption $s \in E$. Let $r := \eta(r_0, s)$. We have:

$$s\alpha_i \in \pi(P_{1x}||P_{2x}||\cdots||P_{nx}) \iff g_i(\alpha_i)(V(s)) = 1 \iff V(s) \in \mathcal{L}_i \iff \eta(r,\alpha_i)! \text{ (Lem. 2)}$$

i.e. $s\alpha_i \in E$. ■

The next 2 examples illustrate the idea.

Example 1 Shown in Fig. 6 are two centralized supervisors $\mathbf{E_1}$ and $\mathbf{E_2}$ where $n = 3$ and $\Sigma_{o,i} = \Sigma_{c,i} = \{\alpha_i\},\ i = 1, 2, 3$. A state r is labeled with all values in the set $\ell(r)$. For example, in $\mathbf{E_1}$, we have $\ell(r_1) = \{(1,0,0),(0,1,1)\}$ (for brevity a triple (i,j,k) is written as ijk).

The centralized supervisor $\mathbf{E_1}$ is state-independent as for any pair of distinct states (r, r') we have $\ell(r) \cap \ell(r') = \emptyset$. On the other hand, $\mathbf{E_2}$ is clearly not state-independent: we have $\ell(r_1) \cap \ell(r_2) = \ell(r_1) = \ell(r_2)$. ◊

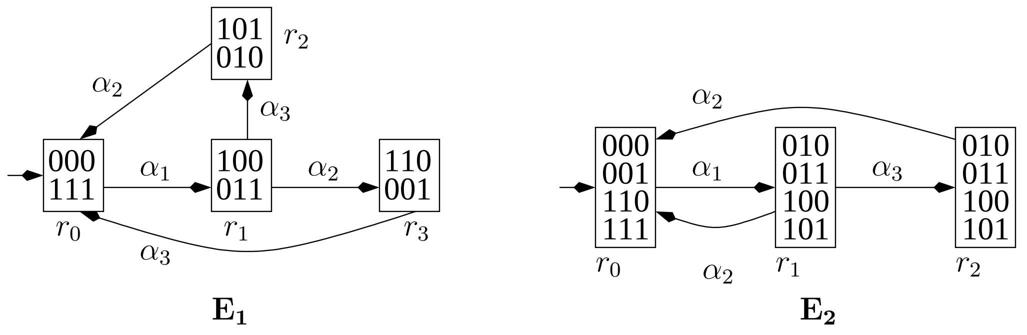


Fig. 6. The centralized supervisor $\mathbf{E_1}$ is state-independent while $\mathbf{E_2}$ is not.

Example 2 As shown in Fig. 7, the specification (centralized supervisor) $\mathbf{E}$ of our running ABP example is state-independent. We have: $\mathcal{L}_1 = \{00, 11\}$ and $\mathcal{L}_2 = \{01, 10\}$. Thus

$$g_1(\alpha_1) = \overline{x_{11} \oplus x_{22}}, \qquad g_2(\alpha_2) = x_{11} \oplus x_{22}. \quad ◊$$

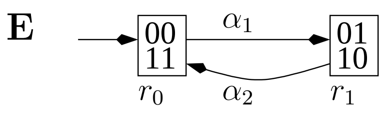


Fig. 7. The centralized supervisor $\mathbf{E}$ of the ABP is state-independent.

Note that if channels were unreliable then, say, the private variable x_{22} in the guard formula $g_1(\alpha_1)$ must be replaced with its local copy x_{12}. The mechanism by which x_{12} is updated with x_{22} is discussed in Section VI.

V. Towards the general problem in the presence of ideal channels

In Section IV we used tuples of booleans to label the states of a centralized supervisor $\mathbf{S} = (R, \Sigma, \eta, r_0)$, and used a fixed updating mechanism in which the occurrence of a significant event α_i toggles the value of the variable x_{ii}, $1 \leq i \leq n$. In general, the class of state-independent centralized supervisors, which can be implemented by communicating decentralized supervisors when channels are ideal, will be widened if one dedicates more bits to count the significant events of processes. The next example illustrates the point.

Example 3 As shown in Fig. 8-a the centralized supervisor $\mathbf{S}$ is not state-independent with respect to ℓ when events are counted modulo 2 as $\ell(r_0) \cap \ell(r_2) \neq \emptyset$. Now, let us use two binary variables x_{11}^1 and x_{11}^2 to count α_1. The first two occurrences of α_1 increment $x_{11}^1 x_{11}^2$ by one, while its next two occurrences decrement $x_{11}^1 x_{11}^2$ by one back to 00, i.e. the updating functions count α_1 modulo 3 (as opposed to modulo 2 counting of the previous section). With the new labeling map $\ell' : R \rightarrow\ pwr(\{0,1,2\} \times \mathbb{B})$, we have $\forall r, r' \in R.\ r \neq r' \Rightarrow \ell'(r) \cap \ell'(r') = \emptyset$, i.e. the centralized supervisor is state-independent with respect to ℓ'. $\Diamond$

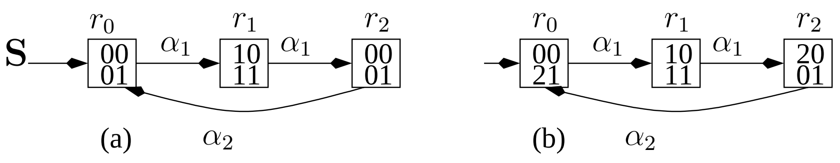


Fig. 8. The centralized supervisor $\mathbf{S}$, with $\Sigma_{o,i} = \Sigma_{c,i} = \{\alpha_i\}$, $\Sigma_{uo,i} = \emptyset$, $i = 1, 2$, is not state-independent when one boolean variable is used to count α_1, while it becomes state-independent when two boolean variables are used to count α_1.

Thus, in general, more elegant coding schemes are required to insure that labels are unique, and that each event changes only the value(s) of the process's own private variable(s). With such coding schemes, which may use more than one private variable, there is no reason to limit to one "significant" event per process, and this assumption can be relaxed, too. The following definition characterizes the labeling maps that have the above desired properties.

Remark 1 Such coding schemes rely on the observation and encoding of *state changes* in a recognizer $\mathbf{S} = (R, \Sigma, \eta, r_0)$ (of the centralized supervisor). Since no state change is observed for events which participate solely in selfloops, i.e. events in $\Sigma_{loop} = \Sigma_{uo} \cup \{\sigma \in \Sigma_o \mid \forall r, r' \in R.\ r' = \eta(r, \sigma) \Rightarrow r = r'\}$, these events might be safely ignored as long as such coding schemes are concerned. However, if an event, say α_i, which is selflooped in one state, say r_1, causes a state change in another state, say r_2, then some provisions should be made to help the coding scheme *observe* all α_i-labeled transitions, including the selfloops. As a remedy, in this case a state $\hat{r}_1$ is added to $\mathbf{S}$ which inherits all the outgoing non-selfloop transitions of r_1, while all selfloop transitions in r_1, which are not labeled by events in Σ_{loop}, are replaced with transitions with the same labels from r_1 to $\hat{r}_1$ and vice versa. By following this procedure, all selfloops in a state that cause state changes in other states are made *observable* to the coding scheme. Note that in the worst case, the state size of the new recognizer (which is still deterministic) would be twice that of the original recognizer. In what follows, the coding schemes are always assumed to be applied to recognizers with possible selfloops formed only by events in Σ_{loop}. Moreover, we assume, without loss of generality, that in the next examples $\Sigma_{uo} = \emptyset$.

Definition 4 Let $\mathbf{S} = (R, \Sigma, \eta, r_0)$ be a centralized supervisor modified if necessary as in Remark 1. An Agent-wise Labeling Map (ALM) is a map $\ell : R \to pwr(\mathbb{N}^n)$ with the following properties:

1) $\underline{0} \in \ell(r_0)$;
2) $\forall r, r' \in R.\ r \neq r' \Rightarrow \ell(r) \cap \ell(r') = \emptyset$ (labels are unique);
3) $\forall r, r' \in R, r \neq r',\ \forall \sigma \in \Sigma_o,\ \forall \underline{v} \in \mathbb{N}^n. \quad v \in \ell(r) \wedge r' = \eta(r, \sigma)$
$$\Longrightarrow\ \exists \underline{v}' \in \mathbb{N}^n.\ \underline{v}' \in \ell(r')\ \wedge\ [\forall i \in I_o(\sigma).\ v_i \neq v'_i]\ \wedge\ [\forall j \in I \setminus I_o(\sigma).\ v_j = v'_j].$$

We call an ALM *finite* if its image is a finite set. □

Remark 2 By the second property $\mathbf{S}$ is state-independent with respect to an ALM. Let $\mathcal{L}$ be the set of all ALMs defined for $\mathbf{S}$ and consider the partial ordering $\preceq$ defined in (5). Since in general $(\mathcal{L}, \preceq)$ is not a complete lattice, there may exist more than one minimal labeling map, each using a different number of variables.

Remark 3 The last property implies that an ALM neither limits the number of events participating in $\mathbf{S}$ from each process, nor makes any distinction between them.

To show the existence of a finite ALM, we need the following definitions.

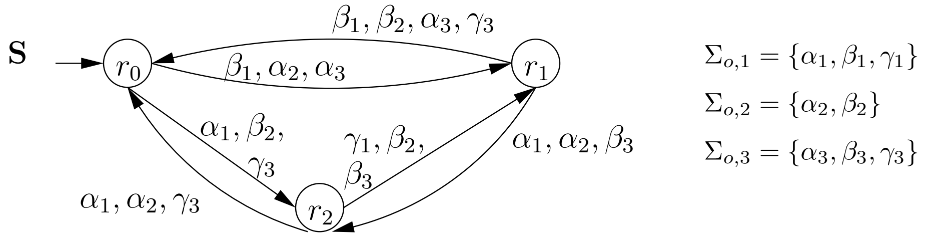
Definition 5 Consider a centralized supervisor $\mathbf{S} = (R, \Sigma, \eta, r_0)$ and an index set I. Two distinct states $r, r' \in R$ are called *I-connected* if for all $i \in I$ there exists a $\sigma \in \Sigma_{o,i}$ such that $r' = \eta(r, \sigma)$. Recognizer $\mathbf{S}$ is I-connected if every pair of distinct states in $\mathbf{S}$ are I-connected. $\square$

Figure 9 illustrates an example of an I-connected recognizer $\mathbf{S}$.

Definition 6 Let $\underline{v}, \underline{v}' \in \mathbb{N}^n$ be labels and $i \in I$. We say $\underline{v}$ is an *i-sibling* of $\underline{v}'$ if $v_i \neq v_i'$ and $v_{-i} = v_{-i}'$. $\square$

Theorem 4 There exists an efficiently computable finite ALM for every centralized supervisor $\mathbf{S} = (R, \Sigma, \eta, r_0)$, where $\mathbf{S}$ is modified if necessary as in Remark 1.

Sketch of the Proof:
The proof is done by establishing a bijection between building an ALM for $\mathbf{S}$ and another problem described below. Assume that $R = \{r_0, r_1, \ldots, r_{m-1}\}$, and define $J = \{0, \ldots, m-1\}$. Notice that since all events in Σ_o are observable, each transition's event in $\mathbf{S}$ belongs to at least one $\Sigma_{o,i}$, $i \in I$.



$\mathbf{S}$
r_0
r_1
r_2
$\beta_1, \beta_2, \alpha_3, \gamma_3$
$\beta_1, \alpha_2, \alpha_3$
$\alpha_1, \beta_2, \gamma_3$
$\gamma_1, \beta_2, \beta_3$
$\alpha_1, \alpha_2, \beta_3$
$\alpha_1, \alpha_2, \gamma_3$
$\Sigma_{o,1} = \{\alpha_1, \beta_1, \gamma_1\}$
$\Sigma_{o,2} = \{\alpha_2, \beta_2\}$
$\Sigma_{o,3} = \{\alpha_3, \beta_3, \gamma_3\}$

Fig. 9. An example of an I-connected recognizer.

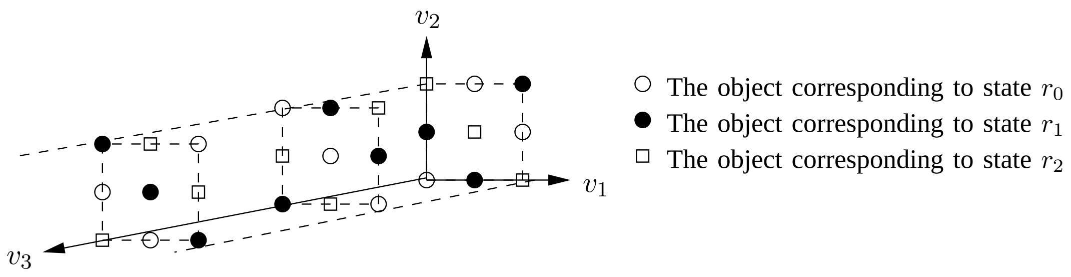

v_2
v_1
v_3
○ The object corresponding to state r_0
● The object corresponding to state r_1
□ The object corresponding to state r_2

Fig. 10. A Latin hypercube with $n = 3$ (the number of axes) and $m = 3$ (the number of objects): There exists exactly one copy of each object in every direction.

We make two assumptions which are relaxed later in the proof: (i) that $\Sigma_{o,i}$'s are mutually disjoint and (ii) that $\mathbf{S}$ is I-connected. By Definition 4, building an ALM for $\mathbf{S}$ is equivalent to finding m ***mutually disjoint*** sets $L_j = \ell(r_j),\ j \in J$, each consisting of labels $\underline{v} \in \mathbb{N}^n$ satisfying Items 1 and 3. Item 1 implies that $\underline{0} \in L_0$. Under assumptions (i) and (ii) mentioned above, since for each i there is a transition from every state r_k to every other state, Item 3 of this definition requires that each tuple $\underline{v} \in L_k$ have an i-sibling in every other state, for a total of $m-1$ distinct i-siblings (since label sets of states must be disjoint by Item 2 of Definition 4).

Graphically, each n-tuple label $\underline{v}$ may be considered as a point in $\mathbb{N}^n$. For the ease of representation, a point is marked by one of m distinct *objects*, each corresponding to a state of $\mathbf{S}$; for instance, in Fig. 10, the label $(0, 2, 0)$ is marked by a square, indicating its membership to the label set of state r_2. Note that all i-siblings of $\underline{v}$ are located on a straight line parallel to $i \in I$ axis. As argued before, to have i-siblings of $\underline{v}$ in all other states, along every dimension $i \in I$ there must exists exactly one copy of each object, for a total of m distinct objects. Accordingly, one arrangement would be to construct an m by m hypercube in $\mathbb{N}^n$, one corner of which is located at the origin, and in its every dimension $i \in I$ there exists exactly a copy of each of m distinct objects, i.e. m i-sibling labels, each belonging to one $L_j,\ j \in J$. Such an arrangement is called a *Latin hypercube of side* m, and can be efficiently computed [21], [22]; a simple example is shown in Fig. 10.

The above argument reveals that there exists a finite ALM for a given $\mathbf{S}$ under the assumptions (i) and (ii). Assumption (ii) creates a worst-case scenario; an ALM for $\mathbf{S}$ in which assumption (ii) holds is also an ALM for $\mathbf{S}'$ which is identical to $\mathbf{S}$, except that some transitions are removed, and therefore (ii) may no longer hold.

Let us now assume that assumption (i) is relaxed, i.e. there is an event σ for which $|I_o(\sigma)| > 1$. Item 3 of Definition 4 thus requires that the occurrence of σ move the current point in the Latin hypercube to a point whose every coordinates in $I_o(\sigma)$ changes, while others in $I \setminus I_o(\sigma)$ remain unchanged. Such a point always exists since there is exactly one copy of each of the m distinct objects in each direction of the Latin hypercube, and therefore, there always exists a path which starts from the current point, each time moves along one of the directions specified by $I_o(\sigma)$ in some specific order, and ends up in the required point in the hypercube. Hence the proof remains valid if all the assumptions are lifted. ■

Remark 4 It is interesting to note that, in general, the hypercube of m^n labels, with exactly m copies of each object along each direction, provides an upper bound for the number of labels required by an ALM, in the sense that it is possible to find an ALM with a smaller image size if assumption (ii) is relaxed. On the other hand, it provides the *minimum* number of the required labels in the worst-case scenario where for every pair of recognizer's states and for each i, some events in $\Sigma_{o,i}$ trigger a move from one state of the pair to the other.

The next example illustrates the procedure mentioned in the above proof and Remark 1.

Example 4 Consider the centralized supervisor $\mathbf{S}$ in Fig. 11-a and the subalphabets $\Sigma_{c,1} = \Sigma_{o,1} = \{\alpha, \alpha_1, \beta_1\}$ and $\Sigma_{c,2} = \Sigma_{o,2} = \{\alpha, \alpha_2, \beta_2\}$. Following Remark 1, we examine selfloop transitions in $\mathbf{S}$ and notice that β_1 causes no state change and can thus be safely ignored. On the other hand, α_1 and α_2 cause state change from r_0 to r_1, and therefore they are replaced by transitions between r_1 and the new state $\hat{r}_1$, which inherits from r_1 its outgoing transitions. For the new recognizer $\hat{\mathbf{S}}$ in part (b), which has 3 states r_0, r_1, and $\hat{r}_1$, by the proof of Theorem 4 a finite ALM may be found using a Latin square of side 3. Such an arrangement is shown in Fig. 11-c simply by associating the horizontal and vertical axes with agents 1 and 2, respectively, and placing three objects, each representative of one state, in the first row, and shifting this row one unit to the left each time to create the other rows. By Item 1 of Definition 4, point $(0, 0)$ is assigned to r_0. We notice that state r_0 is connected to r_1 through events $\alpha_1 \in \Sigma_{o,1}$, $\alpha_2 \in \Sigma_{o,2}$, and the common event α. Thus, corresponding to each vector of values in $\ell(r_0)$ (e.g. $(0, 0)$), there are a 1-sibling (e.g. $(2, 0)$), a 2-sibling (e.g. $(0, 2)$), and a vector differing in *both* coordinates (e.g. $(2, 2)$) in $\ell(r_1)$. Similar observations can be made for the other states and their labels. $\Diamond$

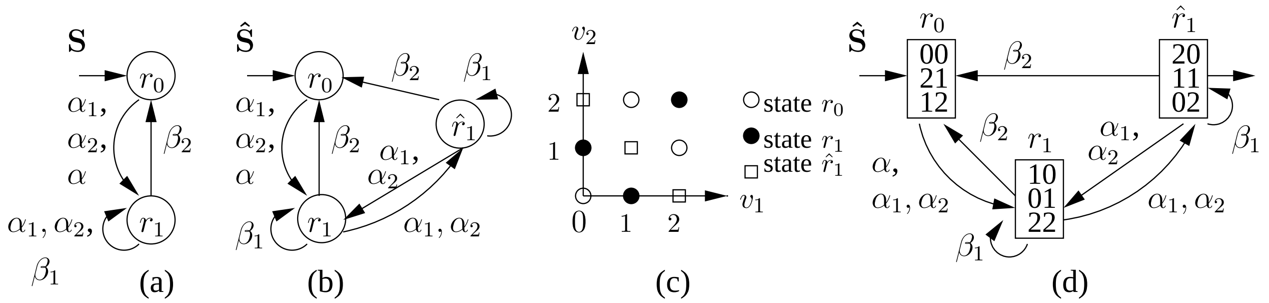


Fig. 11. (a) A centralized supervisor and (b) its unfolded version. (c) Graphical representation of a finite ALM. (d) The encoded supervisor.

Definition 7 Let an ALM be employed for labeling the states of a centralized supervisor $\mathbf{S} = (R, \Sigma, \eta, r_0)$ and denote by V_i the set of numbers used by each agent for labeling; that is,

$$\forall i \in I. \quad V_i = \{v_i \in \mathbb{N} |\ \exists r \in R, v_{-i} \in \mathbb{N}^{n-1}.\ (v_i, v_{-i}) \in \ell(r)\} \tag{7}$$

The set of private boolean variables with which each agent needs to implement its labels is denoted by $X_{ii} = \left\{x_{ii}^k | k \in \{1, \ldots, \lceil log_2 |V_i| \rceil\}\right\}$. □

In general the guard formula of an event α_i is a function of all of agent i's variables, i.e. $g_i(\alpha_i) = h_i(X_{ii}, X_{ci})$. Also, the updating function associated with the private variable x_{ii}^k of process i and an arbitrary event of the process, say α_i, is not *a priori* fixed and is a function of all private and copy variables of process i, i.e. $a_i(x_{ii}^k, \alpha_i) = f_{i,k}(X_{ii}, X_{ci})$. The function $f_{i,k}$ must be designed to implement the desired labeling map as part of the solution to the decentralized control implementation problem. The next example illustrates this point.

Example 5 For the centralized supervisor in Fig. 8-b assume that all channels are ideal. Then using two (one) private variables for process 1 (2) to encode the states as ($x_{11}^1 x_{11}^2, x_{22}^1$), the non-identity updating functions can be calculated as: $a_1(x_{11}^1, \alpha_1) = x_{11}^2 \overline{x_{22}^1}$, $a_1(x_{11}^2, \alpha_1) = \overline{x_{11}^2}$ and $a_2(x_{22}^1, \alpha_2) = \overline{x_{22}^1}$. The guard formulas $g_1(\alpha_1) = \overline{x_{11}^1 \oplus x_{22}^1} + x_{11}^2$ and $g_2(\alpha_2) = x_{11}^1 \overline{x_{22}^1} + \overline{x_{11}^1} x_{22}^1 \overline{x_{11}^2}$ insure that α_1 is enabled only in r_0 and r_1, while α_2 is enabled only in r_2. (Calculation of guards and updating functions are detailed in [14].) ◊

As is evident from the above example, in general, both guards and updating functions depend on the values of (copies of) private variables of other processes. When $g_i(\alpha_i) = h_i(X_{ii}, X_{ci})$, communication is needed to update the copies in X_{ci} to insure that the right control decision is made (“communication for control”). When $a_i(x_{ii}^k, \alpha_i) = f_{i,k}(X_{ii}, X_{ci})$, communication is needed to update the copies in X_{ci} to insure that the variables in X_{ii} are properly updated; in other words, to update an agent’s estimate of the centralized supervisor’s state (“communication for observation”). Thus, given a controllable, observable, but non-coobservable specification and its enforcing centralized supervisor, in a network with ideal channels where local copies of agents’ private variables can be updated instantaneously, the communication protocol is specified by the following entities; The *control decision* of each agent, i.e. guards, and the communications for control and/or observation amongst agents. In this sense, the protocol design is equivalent to SCDS where each decentralized supervisor makes control decisions based on its own observation of the plant behavior and the received communications from other supervisors. Note that in the EFSM formalism supervisors do not exist as separate entities; they are implemented by guards and updating functions of the processes’ EFSMs. As such, communication takes place between the processes themselves.

While in general finding answers to questions about ordering and minimality of communication might be a difficult task, in what follows we restrict EFSM models so that they do not need “communication for observation,” and identify a class of centralized supervisors that can be implemented by such EFSMs.

Definition 8 [15] We say we have *independent updating functions* when

$$\forall i \in I,\ \forall k \in \mathbb{N},\ \forall x_{ii}^k \in X_{ii},\ \forall \alpha_i \in \Sigma_{o,i}.\ a_i(x_{ii}^k, \alpha_i) = f_{i,k}(X_{ii}). \qquad \square$$

The following Lemma identifies ALMs that yield independent updating functions.

Lemma 5 An ALM results in independent updating functions for the centralized supervisor **S** iff it assigns the same component labels to the states of **S** which are reached by strings that are observationally equivalent to that component.

Proof (Only if): Choose any two strings $s, s' \in \Sigma^*$ which are observationally equivalent to agent i, i.e. $\pi_i(s) = \pi_i(s') = \alpha_i^1\alpha_i^2 \ldots \alpha_i^m$, for some $m \in \mathbb{N}$, where $\alpha_i^j \in \Sigma_{o,i}$ for all $j \in \{1, 2, \ldots, m\}$. By Definition 8, we have

$$\forall k \in \mathbb{N},\ \forall x_{ii}^k \in X_{ii}, \forall \alpha_i \in \Sigma_{o,i}.\ a_i(x_{ii}^k, \alpha_i) = f_{i,k}(X_{ii}).$$

Thus, starting from the initial state where $\underline{0} \in \ell(r_0)$ (Item 1, Definition 4), every x_{ii}^k may be affected only by the current values of the variables in X_{ii} as a result of the occurrence of an event $\alpha_i^j \in \Sigma_{o,i},\ j \in \{1, 2, \dots, m\}$. Since both s and s' include the same ordering of such events, their i'th labels become equal.

(If): If the updating functions are not independent, then there exists a variable $x_{ii}^k \in X_{ii}$ and an event α_i such that the corresponding updating function depends on sets of variables other than X_{ii}, i.e. $a_i(x_{ii}^k, \alpha_i) = f_{i,k}(X_{ii}, X_{ci})$. As a result, valuation of this variable by agent i depends on the values of other agents' private variables, too. This, in turn, implies that the assignment of the labels (which are actually implemented using the variables in X_{ii}, including x_{ii}^k) by agent i would depend on the other agents' observations. Thus, two strings which are observationally equivalent to agent i may be assigned different labels by agent i, which is a contradiction. ■

It turns out that EFSMs with independent updating functions can meet a specification *only if* the specification satisfies a weak version of "joint observability" property [16]. We show this point next.

Definition 9 [16] S is *jointly observable* with respect to $\pi(L)$ and $(\Sigma_{o,1}, \dots, \Sigma_{o,n})$ iff

$$\forall \rho \in S,\ \forall \rho' \in \pi(L) \setminus S,\ \exists i \in I.\ \ \pi_i(\rho) \neq \pi_i(\rho'). \qquad ■$$

In words, joint observability requires that for every two lookalike legal-illegal sequences in the plant's behavior, there exists at least one supervisor which can tell them apart. However, in control problems one always cares about the first instance at which the legal behavior is violated, and any subsequent evolution of illegal behavior is not of interest (as it is to be prevented by a controller). From this viewpoint joint observability is too strong a property for control applications, and therefore below we introduce a weaker notion which requires the existence of a supervisor which can distinguish two legal strings when an event extends one to a legal string while extends the other to an illegal string.

Definition 10 [15] S is *weakly jointly observable* with respect to $\pi(L)$ and $(\Sigma_{o,1}, \dots, \Sigma_{o,n})$ iff

$$\forall s, s' \in S,\ \forall \sigma \in \Sigma.\ \ s\sigma \in S \wedge s'\sigma \in \pi(L) \setminus S\ \Rightarrow\ \exists i \in I.\ \ \pi_i(s) \neq \pi_i(s'). \qquad \square$$

Lemma 6 [15] Joint observability implies weak joint observability.

Proof: Choose any $s, s' \in S$, and $\sigma \in \Sigma$ such that $s\sigma \in S \ \wedge \ s'\sigma \in \pi(L) \setminus S$. Take $\rho = s\sigma$ and $\rho' = s'\sigma$. By joint observability we know that there exists $i \in I$ such that

$$\begin{aligned}
&\pi_i(\rho) \neq \pi_i(\rho') \\
&\Rightarrow \pi_i(s\sigma) \neq \pi_i(s'\sigma) \\
&\Rightarrow \pi_i(s)\pi_i(\sigma) \neq \pi_i(s')\pi_i(\sigma) \\
&\Rightarrow \pi_i(s) \neq \pi_i(s').
\end{aligned}$$

■

Lemma 7 [15] A language S is weakly jointly observable with respect to $\pi(L)$ and $(\Sigma_{o,1}, \ldots, \Sigma_{o,n})$ if there exists an ALM for **S** such that the associated updating functions are independent.

Proof: Assume that there exist independent updating functions and let $s, s' \in S$ and $\sigma \in \Sigma$ be such that $s\sigma \in S$ and $s'\sigma \in \pi(L) \setminus S$. Write the states reached by s and s' as r and r', respectively, so that there exist $\underline{v}, \underline{v}' \in \mathbb{N}^n$ such that $\underline{v} = (v_i, v_{-i}) \in \ell(r)$ and $\underline{v}' = (v'_i, v'_{-i}) \in \ell(r')$ as in Definition 4. If S is not weakly joint observable, then:

$\forall i \in I.\ \pi_i(s) = \pi_i(s')$

$\Rightarrow \forall i \in I.\ v_i = v'_i$ (Defn. 4)

$\Rightarrow \underline{v} = \underline{v}'$

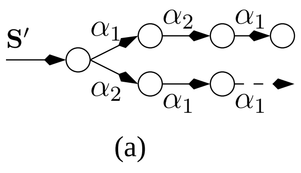
$\Rightarrow r = r'$ (Only if part of Lem. 5)

which is a contradiction. ■

The above result states a structural property for the language of the centralized supervisor without which no independent updating functions may be derived regardless of the choice of ALM. However, for an updating function to be independent of other agents' variables, it is necessary that its corresponding component labels assigned to states by an ALM be such that any changes in their values depend only on the current values of its own component labels. In simple words, the choice of the ALM should be such that updating the labels of every agent is a *function* of its own values. The next example illustrates these points.

Example 6 It can be verified that S' in Fig. 12-a is not weakly jointly observable. As a counterexample, let $s = \alpha_1\alpha_2$, $s' = \alpha_2\alpha_1$ and the dashed arrow represents the plant's illegal move. Then while $s\alpha_1$ is legal and $s'\alpha_1$ is illegal, we have $\pi_i(s) = \pi_i(s') = \alpha_i$ for $i = 1, 2$. Therefore, by the previous lemma a set of independent updating functions cannot be found to implement $\mathbf{S}'$ regardless of the choice of ALM.

For the weakly jointly observable S in Fig. 8-b, the labeling map ℓ' used in Example 5 does not yield independent updating functions: for agent 1, the component label 1 in state r_1 is mapped sometimes to 2 and sometimes to 0, depending on the label assigned by agent 2, so that its updating action cannot be expressed as a function on its set of labels $\{0, 1, 2\}$, but as a function on the cartesian product of both agents' labels, i.e. $\{0, 1, 2\} \times \{0, 1\}$, which makes the updating functions dependent. Now, let us apply the ALM ℓ'' of Fig. 12-b to the same specification; note that the specification remains state-independent with respect to ℓ''. Observe that under the new labeling every component label in the set $\{0, 1, 2, 3\}$ for agent 1 is uniquely mapped to an element in the same set. In this case the set of boolean variables and the last two updating functions remain as in Example 5, while the first updating function becomes $a_1(x^1_{11}, \alpha_1) = x^1_{11} \oplus x^2_{11}$, hence independent updating functions are achieved. $\diamondsuit$



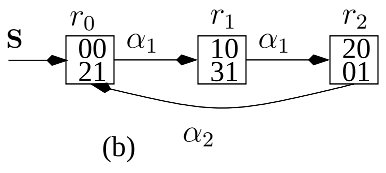


Fig. 12. $\Sigma_{o,i} = \Sigma_{c,i} = \{\alpha_i\},\ i = 1, 2$. (a) A language that is not weakly jointly observable. (b) A state-independent centralized supervisor yielding independent updating functions.

When updating functions are independent, as in the ABP example, the solution of SCDS enjoys the following property. We first need to define “minimality” of boolean functions.

Definition 11 We say a boolean formula is in a *reduced form* if it contains a minimal number of boolean variables after possibly utilizing *don't care* conditions [23] . $\square$

Remark 5 Notice that when computing reduced forms for guards and updating functions, one should take into account the fact that in the end it is desired to have minimal exchange of information among the supervisors. As a result, whenever there are more than one reduced form for a boolean formula or function, the one(s) which share more common variables with other formulas and functions are selected. This issue is outside the scope of the present work.

Lemma 8 Let E be a global controllable, observable, but non-coobservable specification and $\mathbf{S}$ be the centralized supervisor enforcing E, whose associated updating functions are independent. Then E can be implemented over a network of ideal channels if a number of bits are communicated in order to reevaluate guards, while no communication is needed for reevaluating the updating functions. Moreover, this number may be chosen minimally, in the sense of Definition 11, up to the ALM used to label the states of $\mathbf{S}$.

Proof: Similar to the proof of Theorem 3, by the state-independency of $\mathbf{S}$ with respect to the ALM (Item 3, Definition 4), the formulas representing the guards can be computed as functions of the private and copy variables, i.e.:

$$\forall i \in I,\ \forall \alpha_i \in \Sigma_{o,i}.\ g_i(\alpha_i) = h_i(X_{ii}, X_{ci}).$$

Thus, to apply control over its corresponding event α_i, agent i needs to receive only the updated values of the copy variables in X_{ci} (i.e. communication for control). Following the fact that the image of the ALM is finite, only a *finite* number $|X_{ci}|$ of bits must be received (instantaneously, under the assumption of ideal channels) in order to make the right control decisions. On the other hand, the independency of updating functions implies that every such agent updates its private variables in X_{ii} based on its own observation of the plant behavior (Lemma 5), and therefore no communication for observation is required.

Upon computing one of the (possibly several) reduced forms of the guard formulas (see Definition 11), a minimal number of copy variables in X_{ci} are needed for communication. We notice that there might exist more than one ALM to label the states of $\mathbf{S}$, each using $|X_{ii}|$ private variables for agent i. As a result, the minimality is up to the ALM used in labeling the states of $\mathbf{S}$. ■

In conclusion, over an ideal channel, a protocol for a non-coobservable specification with associated independent updating functions of the centralized supervisor, simply requires the communication of a (minimal) number of bits for agents' control purpose of reevaluating their guard formulas.

Example 7 For $\mathbf{S}$ in Fig. 12-b, we have $g_1(\alpha_1) = \overline{x_{11}^1 \oplus x_{22}^1} + x_{11}^2$ and $g_2(\alpha_2) = x_{11}^1 \oplus x_{22}^1$. Therefore, the *protocol* requires process 1 (2) to attach to each data message it sends the value of x_{11}^1 (x_{22}^1, respectively). Notice that value of x_{11}^2 needs not be communicated. $\Diamond$

Remark 6 It is worth comparing our ALM-based approach to the *estimator structure* of [10] and *possible worlds* of [11]. The following observations can be made about our approach versus those of [10] and [11].

- While an ALM can be found for any deterministic automaton of a centralized supervisor (after a possible modification as explained in Remark 1), the other two approaches have been used for reachability trees only, and their applicability to general automata containing loops is not claimed and does not seem obvious.
- An ALM labels the states of a centralized supervisor using an agent-wise viewpoint, while the other two approaches rely on a global labeling for the states and then gathering the lookalike state labels for each agent as a set of state estimates [10] or possible worlds [11]. Since in decentralized control the supervisors view the plant's behavior subject to their partial observations, the ALM labeling provides a natural formulation for the distribution of information within the network. Moreover, the ALM approach views the labels as an integral part of the implementation of supervisor's commands, while in the other two approaches labeling is only a simplifying tool and the viewpoint is quite abstract.
- The final rules for communication in the other two approaches are always translated in terms of communicating the state estimates (or possible worlds), while in the ALM approach (more specifically, in the EFSM framework) everything is done with respect to bits of information used by each local supervisor to encode the states of a global supervisor. As a result, the latter serves to define a practical measure, especially when issues such as minimal communication are studied.
- Another advantage of the EFSM formalism is its compact representation of the supervisors' commands and observations using boolean formulas and functions, while the other two approaches make use of the supervisors' automata.
- The works in [10] and [11] adopt "the latest safe point" and "as early as possible" policies, respectively, to deal with the issue of "when" to communicate. Although this issue is not explicitly addressed in our work, where the focus is on the logical aspects of protocol design, it is implicit that communication takes place whenever necessary, in other words, when guard and updating functions need to be reevaluated.
- Moreover, the case of unreliable channels, which is the subject of the last section, is not

studied in the aforementioned papers.

Noting the similarities between [10] and [11], where either state estimates or possible worlds are communicated, through the following example, taken from [10], we illustrate our formulation and solution and that of [10] for a simple problem.

Example 8 Consider the centralized supervisor $\mathbf{S}$ in Fig. 13-a where $\Sigma_{o,1} = \{\alpha_1, \beta_1, \gamma_1\}$, $\Sigma_{o,2} = \{\alpha_2\}$ and event γ_1 is controllable by the first supervisor. Part (b) shows the labels assigned to the states by an ALM. Representing the component labels $\{0, 1, 2, 3\}$ and $\{0, 1, 2\}$ of, respectively, the first and the second supervisors, using binary variables $x_{11}^1 x_{11}^2$ and $x_{22}^1 x_{22}^2$, the guard associated with γ_1 would be $g_1(\gamma_1) = x_{22}^2$ and the updating functions may be computed as $a_1(x_{11}^1, \alpha_1) = 0$, $a_1(x_{11}^2, \alpha_1) = 1$, $a_1(x_{11}^1, \beta_1) = 1$, $a_1(x_{11}^2, \beta_1) = 0$, $a_1(x_{11}^1, \gamma_1) = 1$, $a_1(x_{11}^2, \gamma_1) = 1$, $a_1(x_{22}^1, \alpha_2) = x_{11}^2$, and $a_1(x_{22}^2, \alpha_2) = \overline{x_{11}^2}$. Therefore, by the time supervisor 1 wants to make its control decision for γ_1 at state 7, it should have received the updated value of x_{22}^2 from supervisor 2 (i.e. *communication for control*). However, the last time x_{22}^2 is updated is upon the occurrence of α_2, for which supervisor 2 needs to receive the most recent value of x_{11}^2 (i.e. *communication for observation*). This latter variable is updated solely based on the observation of supervisor 1, so no more communication is required. As a result, our solution requires that a) whenever α_2 occurs, supervisor 2 receive 1 bit to reevaluates its updating function(s) and b) before making a decision on disabling γ_1, supervisor 1 receive 1 bit to reevaluate its guard.

The solution in [10] relies on first a global labeling of states of $\mathbf{S}$ as in part (a) of the figure, and second on the estimator structure in part (c). Every state of the estimator structure consists of a quadruple whose top and bottom elements correspond to the event occurred and the state it leads to in $\mathbf{S}$. The second and the third elements are, respectively, the state estimates made by supervisors 1 and 2 after the occurrence of events. Computing the latest safe point as state 5, the authors in [10] come up with the communication policy which prescribes that supervisor 2 communicate its state estimate $\{2, 5\}$ at the latest safe point, and as supervisor 2 cannot tell apart state 5 from state 2, it does the same communication at state 2 as well.

Accordingly, the following observations can be made: a) The content of communication consists of 2 bits in our formulation and 2 states (or their labels) in the formulation in [10], which, in general, consists of more than two bits (especially since labels are global). b) Also, our formulation provides a more detailed treatment of the (qualitative) time of each communication.

However, we would like to point out that this example is not an exhaustive comparison between the two methods.

Notice that while our approach is capable of handling any *arbitrary* finite automaton $\mathbf{S}$ with equal ease, this simple example serves to illustrate how naturally the purpose of communication (observation v. control) manifests itself in the designed protocol. $\Diamond$

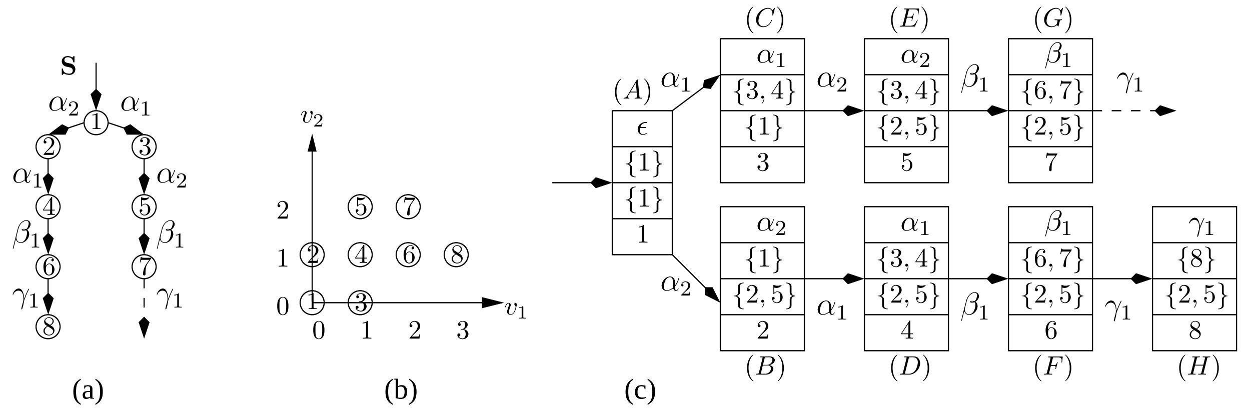


Fig. 13. (a) A centralized supervisor and (b) its labels assigned by an ALM. (c) The estimator structure (without communication) for part (a) (reprinted from Fig. 3 in [10]).

VI. DECCN SOLUTION—UNRELIABLE CHANNELS

This section studies the effects of unreliable channels on implementation of a centralized supervisor. To simplify the study of such effects, we keep assumptions b) and c) of Section IV. However, the results can be generalized to the case of Section V in an appropriate manner.

When process $\mathbf{P_{ix}}$ is connected to process $\mathbf{P_{jx}}$ through an unreliable channel, we assume that process $\mathbf{P_{ix}}$ sends the values of its variables to the channel infinitely often (event β^s_{ij}). Although the transmission could fail several times (event β^e_{ij}), we assume that the channel is *weakly fair*, in the sense that the control information is received error-free by process $\mathbf{P_{jx}}$ (event β^r_{ij}) infinitely often. Thus, the copies of variables in $\mathbf{P_{jx}}$ are updated with the corresponding values in $\mathbf{P_{ix}}$ infinitely often, but as a result of possible transmission errors there is unbounded delay before the eventual update of copies in $\mathbf{P_{jx}}$ takes place. Unfortunately delay in a communication network makes it nearly impossible to implement any specification in which nondeterminism exists. To see this, suppose at a state of a specification both α_i and α_j are enabled, and the occurrence of one entails disabling the other. Then, say, if α_i occurs first, α_j remains enabled until process $\mathbf{P_{jx}}$ is informed that α_i has occurred in $\mathbf{P_{ix}}$ (in our proposed framework, this means that x_{ji} is updated with the value of x_{ii}). Until then, if α_j occurs the specification will clearly be violated. The following example further illustrates the problem.

Example 9 Assume that we would like to implement the centralized supervisor shown in Figure 14, where $\Sigma_{o,i} = \Sigma_{c,i} = \{\alpha_i\}$, $i \in I = \{1, 2, 3\}$. When channels are ideal this could be achieved by introducing boolean variables x_{ii}, $i \in I$, where x_{ii} is toggled upon the occurrence of α_i, i.e. $a(x_{ii}, \alpha_i) = \overline{x_{ii}}$, while guard formulas are found to be $g_1(\alpha_1) = \overline{x_{11} \oplus x_{22} \oplus x_{33}}$ and $g_2(\alpha_2) = g_3(\alpha_3) = x_{11} \oplus x_{22} \oplus x_{33}$.

In the presence of unreliable channels, process i keeps local copies of private variables of processes j and k, denoted respectively by x_{ij} and x_{ik}, which are updated with the values of variables x_{jj} and x_{kk} whenever an error-free communication from the corresponding process is received ($i, j, k \in I,\ i \neq j,\ i \neq k,\ j \neq k$). Thus, $X_i = \{x_{ii}, x_{ij}, x_{ik}\}$. Accordingly, the guard formulas are evaluated "locally," i.e.: $g_1(\alpha_1) = \overline{x_{11} \oplus x_{12} \oplus x_{13}}$, $g_2(\alpha_2) = x_{21} \oplus x_{22} \oplus x_{23}$ and $g_3(\alpha_3) = x_{31} \oplus x_{32} \oplus x_{33}$.

Initially, all variables are zero; thus α_1 is enabled while α_2 and α_3 are disabled, as required at the initial state of the centralized supervisor. Assume that α_1 is taken, and the values of x_{21} and x_{31} are updated with the new value of $x_{11}(= 1)$. At this point, $g_2(\alpha_2) = g_3(\alpha_3) = 1$ while $g_1(\alpha_1) = 0$, as required at state 'b' of the centralized supervisor. Next, assume that α_2 is taken and thus the value of x_{22} is toggled to 1. As a result, $g_2(\alpha_2) = 0$, as required at state 'a' of the centralized supervisor. However, α_3 remains enabled (i.e. $g_3(\alpha_3) = 1$) until the value of x_{32} is updated with the new value of x_{22} by a successful communication from process $\mathbf{P_{2x}}$ to process $\mathbf{P_{3x}}$. Until then, α_3 may be taken, and thus our attempt to implement the centralized supervisor fails. Intuitively, for decentralized supervisory control to work, processes $\mathbf{P_{2x}}$ and $\mathbf{P_{3x}}$ must be

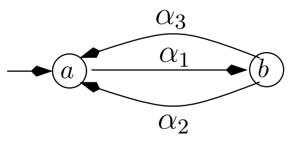


Fig. 14. The centralized supervisor of Example 9.

immediately notified of the occurrence of the other process's significant event. $\Diamond$

The problem is further complicated when the network itself is *nondeterministic*, i.e. there are two or more paths from one process to another. Suppose, for instance, that the specification requires α_j to happen after α_i, and that there are two paths $\wp_a$ and $\wp_b$ from $\mathbf{P_{ix}}$ to $\mathbf{P_{jx}}$. Assume that $\mathbf{P_{jx}}$ enables α_j after it is informed through $\wp_a$ that α_i has occurred. After α_j is taken it should be disabled by $\mathbf{P_{jx}}$ until the next time α_i occurs. Now assume that process $\mathbf{P_{jx}}$ is informed through $\wp_b$ that α_i occurred 0 times modulo 2 (note that counting is performed modulo $N = 2$; more elaborate examples can be devised for arbitrary finite N). Then $\mathbf{P_{jx}}$ does not know for certain what to make of the information just received: if α_i occurred 0 times, then the information is outdated (i.e. the communication was initiated by $\mathbf{P_{ix}}$ before α_i was taken) and must be ignored. In this case, α_j should remain disabled. On the other hand, process $\mathbf{P_{jx}}$ needs to re-enable α_j if it is informed that α_i has occurred for the second time.

We conclude that the class of specifications satisfiable over unreliable communication channels is severely restricted. One can hope for a solution to DECCN when the network is deterministic in the sense defined above, and the centralized supervisor enables a *single* event in its every state. In particular, the following result offers a solution when the specification requires a linear ordering among significant events. First we define a deterministic network.

Definition 12 Let $\mathcal{N}$ be a system consisting of n communicating parallel processes which are connected through a strongly connected network of potentially unreliable channels. $\mathcal{N}$ is *deterministic* if for every i and j, $i \neq j$, there is a unique path from $\mathbf{P}_i$ to $\mathbf{P}_j$. $\Box$

Theorem 9 Let $\{k_1, k_2, \ldots, k_n\}$ be a permutation of $\{1, 2, \ldots, n\}$. If $\mathcal{N}$ is deterministic, the controllable specification $E = \overline{(\alpha_{k_1}\alpha_{k_2}\ldots\alpha_{k_n})^*}$, with $\Sigma_{o,i} = \Sigma_{c,i} = \{\alpha_i\}$, can be satisfied by guarding α_{k_i} with $g_{k_i}(\alpha_{k_i})$, where:

$$g_{k_i}(\alpha_{k_i}) = \begin{cases} \overline{x_{k_1k_1} \oplus x_{k_1k_n}} & ; \quad i = 1 \\ x_{k_ik_i} \oplus x_{k_ik_{i-1}} & ; \quad 2 \leq i \leq n \end{cases}$$

Proof. Since E is controllable and is defined over an observable alphabet, $\mathbf{E}$ can be used as a centralized supervisor enforcing E. Without loss of generality assume that $k_i = i$. We name the states of $\mathbf{E}$ from r_1 to r_n, so that α_i is enabled in state r_i. We show by an inductive argument that in state r_i of $\mathbf{E}$ we have $\forall j.\ g_j(\alpha_j) = 0$, until $g_i(\alpha_i) = 1$ and $\forall j \neq i,\ g_j(\alpha_j) = 0$, at which point α_i can be taken and thus E is satisfied.

- $i = 1$. Since all variables are initialized to 0 we have $g_1(\alpha_1) = 1$ and $\forall j \neq 1.\ g_j(\alpha_j) = 0$.
- $i = k,\ 1 \leq k \leq n$ (we let $n+1 \equiv_n 1$). In state r_k of $\mathbf{E}$ let $g_k(\alpha_k) = 1 \wedge \forall j \neq k.\ g_j(\alpha_j) = 0$, i.e. α_k is the only event enabled in r_k. When α_k is taken, it sets $x_{kk} := \overline{x_{kk}}$ and moves $\mathbf{E}$ to state r_{k+1}. Let $k < n$ (the argument for $k = n$ is similar). Since $g_k(\alpha_k) = x_{kk} \oplus x_{k,k-1}$ was previously 1, after the assignment $x_{kk} := \overline{x_{kk}}$ the guard formula $g_k(\alpha_k)$ evaluates to 0. Thus, temporarily we have $\forall j.\ g_j(\alpha_j) = 0$.

  Observe that when the value of the private variable of $\mathbf{P_k}$ is changed, communication eventually updates all copies $x_{jk},\ j \neq k$, with x_{kk}. Since $g_j(\alpha_j)$ is only a function of x_{jj} and $x_{j,j-1}$, the only guard formula that will be affected by such communications is $g_{k+1}(\alpha_{k+1}) = x_{k+1,k+1} \oplus x_{k+1,k}$, which evaluates to 1 after $x_{k+1,k}$ is updated with x_{kk}. Thus, we have established that in state r_{k+1} eventually $g_{k+1}(\alpha_{k+1}) = 1$ and $\forall j \neq 1.\ g_j(\alpha_j) = 0$. The proof is complete. ■

Remark 7 The restriction on the network can be relaxed if there is a dedicated communication channel between each pair of processes, that is, we have $\forall i, j.\ \mathbf{P_{ix}} \rightarrow \mathbf{P_{jx}}$. In this case, the copy of the private variable of $\mathbf{P_{ix}}$ in $\mathbf{P_{jx}}$ is updated only when a *direct* communication from $\mathbf{P_{ix}}$ to $\mathbf{P_{jx}}$ is received error-free: $a_j(x_{ji}, \beta_{ij}^r) = x_{ii}$, while for $k \notin \{i, j\}$ we have $a_j(x_{jk}, \beta_{ij}^r) = x_{jk}$.

In the next examples Theorem 9 is used to design decentralized communicating supervisors.

Example 10 Consider a system consisting of 4 processes in Fig. 15. The dynamics of each process is unimportant and is thus abstracted as self-loops. Shown in the same figure is a centralized supervisor S enforcing an ordering between events, which we would like to implement by decentralized supervisors embedded in each process. Note that conditions of Theorem 9 are satisfied. The complete design is shown in Fig. 16. ◊

Example 11 The complete model of ABP in EFSM framework is shown in Fig. 17. ◊

VII. Conclusions and future works

Our formulation of the class of protocol synthesis problems (including ABP) makes it plausible to think that over ideal channels the problem of "protocol design" for communication processes

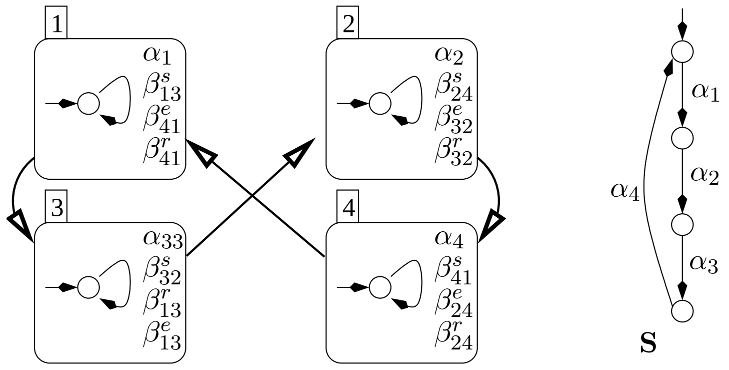
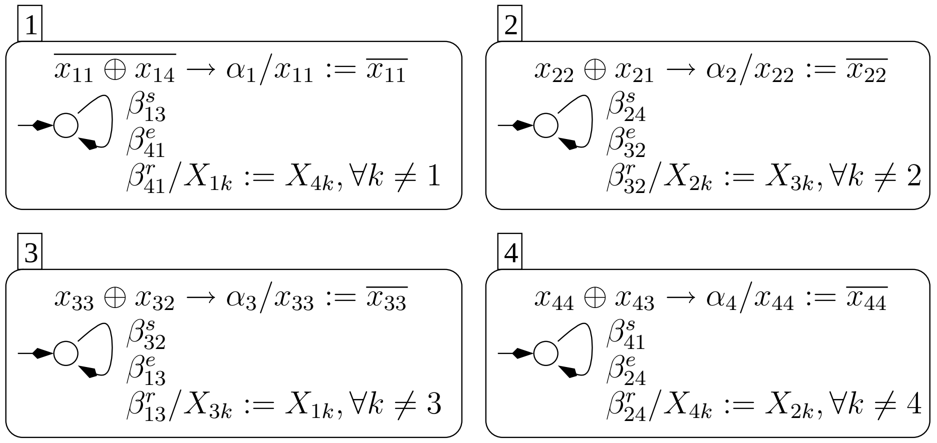


Fig. 15. Four processes in a deterministic network and the centralized supervisor $\mathbf{S}$.



1
$\overline{x_{11} \oplus x_{14}} \to \alpha_1/x_{11} := \overline{x_{11}}$
β^s_{13}
β^e_{41}
$\beta^r_{41}/X_{1k} := X_{4k}, \forall k \neq 1$
2
$x_{22} \oplus x_{21} \to \alpha_2/x_{22} := \overline{x_{22}}$
β^s_{24}
β^e_{32}
$\beta^r_{32}/X_{2k} := X_{3k}, \forall k \neq 2$
3
$x_{33} \oplus x_{32} \to \alpha_3/x_{33} := \overline{x_{33}}$
β^s_{32}
β^e_{13}
$\beta^r_{13}/X_{3k} := X_{1k}, \forall k \neq 3$
4
$x_{44} \oplus x_{43} \to \alpha_4/x_{44} := \overline{x_{44}}$
β^s_{41}
β^e_{24}
$\beta^r_{24}/X_{4k} := X_{2k}, \forall k \neq 4$

Fig. 16. The complete design for the system of Fig. 15.

with non-coobservable specifications can be reduced to the synthesis of communicating decentralized supervisors. Solutions for a special class of problems are presented when the processes need to communicate amongst themselves only for control, and a couple of positive results are stated when channels are unreliable. We seek to extend the results to more general specifications, and study the effect of communication delays in our future work.

One of the important contributions of this paper is that the crucial role the communication network plays in solvability of the decentralized control problem is investigated. With the exception of [13], most works in this area leave one with the impression that generalization from the case where $n = 2$ to arbitrary $n > 2$ is straightforward. Interestingly enough, when $n = 2$ the network is always deterministic. As discussed in this paper, for $n > 2$, one has

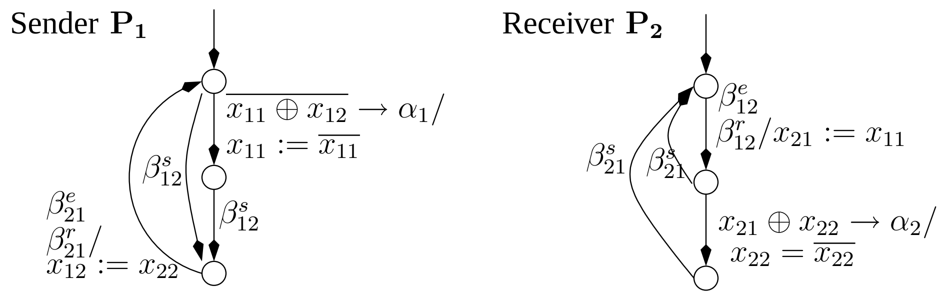

Sender $\mathbf{P_1}$
$\overline{x_{11} \oplus x_{12}} \to \alpha_1/$
$x_{11} := \overline{x_{11}}$
β^s_{12}
β^s_{12}
β^e_{21}
$\beta^r_{21}/$
$x_{12} := x_{22}$
Receiver $\mathbf{P_2}$
β^e_{12}
$\beta^r_{12}/x_{21} := x_{11}$
β^s_{21}
β^s_{21}
$x_{21} \oplus x_{22} \to \alpha_2/$
$x_{22} = \overline{x_{22}}$

Fig. 17. ABP design in EFSM framework.

to require that the network be deterministic, or that every process be connected to every other process through dedicated channels.

REFERENCES

[1] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[2] K. Rudie and W. M. Wonham, "Think globally, act locally: Decentralized supervisory control," *IEEE Trans. Automat. Contr.*, vol. 37, pp. 1692–1708, Nov. 1992.

[3] R. L. Probert and K. Saleh, "Synthesis of communication protocols: Survey and assessment," *IEEE Transactions on Computers*, vol. 40, pp. 468–476, Apr. 1991.

[4] G. Holzmann, "Protocol design: Redefining the state of the art," *IEEE Software*, vol. 9, pp. 17–22, Jan. 1992.

[5] R. Cieslak, C. Deslaux, A. S. Fawaz, and P. Varaiya, "Supervisory control of discrete-event processes with partial observations," *IEEE Trans. Automat. Contr.*, vol. 33, pp. 249–260, Mar. 1988.

[6] K. Rudie and W. M. Wonham, "Supervisory control of communicating processes," in *Protocol Specification, Testing, and Verification*, L. Logrippo, R. L. Probert, and H. Ural, Eds. B. V., North Holland: Elsevier Science Publishers, 1990.

[7] J. H. van Schuppen, "Decentralized control with communication between controllers," in *Unsolved Problems in Mathematical Systems and Control Theory*, V. D. Blondel and A. Megretski, Eds. Princeton, USA: Princeton University Press, 2004.

[8] K. C. Wong and J. H. van Schuppen, "Decentralized supervisory control of discrete-event systems with communication," in *Proc. Workshop on Discrete-Event Systems (WODES'96)*, Edinburgh, UK, 1996, pp. 284–289.

[9] J. H. van Schuppen, "Decentralized supervisory control with information structures," in *Proc. Workshop on Discrete-Event Systems (WODES'98)*, Cagliari, Italy, Aug 1998, pp. 36–41.

[10] G. Barrett and S. Lafortune, "Decentralized supervisory control with communicating controllers," *IEEE Trans. Automat. Contr.*, vol. 45, pp. 1620–1638, Sept. 2000.

[11] S. L. Ricker and K. Rudie, "Incorporating communication and knowledge into decentralized discrete-event systems," in *Proc. IEEE Conference on Decision and Control (CDC'99)*, Phoenix, USA, Dec. 1999, pp. 1326–1332.

[12] K. Rudie, S. Lafortune, and F. Lin, "Minimal communication in a distributed discrete-event system," *IEEE Trans. Automat. Contr.*, vol. 48, pp. 957–975, June 2003.

[13] S. Tripakis, “Decentralized control of discrete-event systems with bounded or unbounded delay communication,” *IEEE Trans. Automat. Contr.*, vol. 49, pp. 1489–1501, Sept. 2004.

[14] Y. Yang and P. Gohari, “Embedded supervisory control of discrete-event systems,” in *IEEE Conference on Automation Science and Engineering (ASE)*, August 2005, pp. 410–415.

[15] A. Mannani, Y. Yang, and P. Gohari, “Distributed extended finite-state machines: Communication and control,” in *Proc. of the IEEE 8th International Workshop on Discrete-Event Systems (WODES’06)*, Ann Arbor, USA, July 2006, pp. 161–167.

[16] S. Tripakis, “Decentralized observation problems,” in *Proc. IEEE Conference on Decision and Control-European Control Conference (CDC-ECC’05)*, Seville, Spain, Dec. 2005, pp. 6–11.

[17] F. Lin and W. M. Wonham, “On observability of discrete-event systems,” *Information Sciences*, vol. 44, no. 2, pp. 173–198, 1988.

[18] W. C. Lynch, “Computer systems: Reliable full-duplex file transmission over half-duplex telephone line,” *Communications of the ACM*, vol. 11, no. 6, pp. 407–410, 1968.

[19] K. A. Bartlett, R. A. Scantlebury, and P. T. Wilkinson, “A note on reliable full-duplex transmission over half-duplex links,” *Communications of the ACM*, vol. 12, pp. 260–261, 1969.

[20] S. Takai, R. Kumar, and T. Ushio, “Characterization of co-observable languages and formulas for their super/sublanguages,” *IEEE Trans. Automat. Contr.*, vol. 50, pp. 434–447, Apr. 2005.

[21] J. Denes and A. D. Keedwell, *Latin suares and their applications*, 1st ed. Academic Press, 1974.

[22] C. J. Colbourn and J. H. Dinitz, Eds., *The CRC handbook of combinatorial designs*, 1st ed. New York, USA: CRC Press, 1996.

[23] M. M. Mano, *Digital design*, 3rd ed. Prentice-Hall, 2002.