A Framework for Modeling Communication Among Decentralized Supervisors for Discrete-Event Systems

A. Mannani and P. Gohari

Abstract

Following the author’s work on Extended Finite-State Machines (EFSMs), we propose a framework to study communication among decentralized supervisors for a (distributed) Discrete-Event System (DES). Equipped with Agent-wise Labeling Maps (ALMs), this framework serves to explore the information structure of the system naturally while enjoying an abstract viewpoint to rigorously express the desired properties. After its development in the centralized case, we extend the framework to the case of decentralized supervisors which need to communicate in the absence of coobservability. Examples illustrate the applicability of the approach.

Index Terms

decentralized supervisory control, discrete-event systems, partial observation.

I. INTRODUCTION

Supervisory Control Theory (SCT) seeks to (minimally) restrict the behavior of a DES, called plant, within the language of a given (controllable and observable) specification by designing a (centralized) supervisor which disables some of the plant’s events [1]. In the absence of global observation of the plant’s behavior, a finite set of decentralized supervisors, each observing the plant’s behavior locally, have to be designed whose synchronous supervision implements the centralized supervisor’s control, in the sense that the resulting closed-loop behavior is the same...
as the one enforced by the centralized supervisor. Given a controllable specification, such a set with non-communicating supervisors exists if and only if the specification is coobservable [2]. Coobservability, which was later extended in [3], simply requires that every illegal move be observed with no ambiguity by at least one local supervisor which can disable it.

In the absence of coobservability, the local supervisors need to communicate amongst themselves to disambiguate their observations, and thus prevent the violation of the specification. Studying such communicating supervisors is inspired by the problem of control over networks such as traffic or communication networks [4]. The rules of communication specifying “who sends what data to whom and when” together with the problem of “minimal” communication have been considered in the abstract contexts of information structures [5], “nonsequential systems” [6], and as refinement relations between control and observation maps [7]. Later works proposed to find more concrete answers to the above questions in terms of “state estimates” [8], “possible worlds” [9], and “observed events” [10]. Further attempts include the formalization of “inference” to make the best use out of the system information made available via observation and communication [11], [12].

However, the above works are are not well equipped to address practical issues such as minimality of communication content, which is measured in bits rather than “events” or “state estimates.” Inspired by such considerations, the authors first proposed the EFSM framework to study communication among supervisors [13]. Equipped with guards and updating functions, EFSMs can model control and thus may serve well to model the closed-loop systems, hence bridging the gap between ad hoc designs for discrete systems and those synthesized systematically using SCT [14]. It was then observed that introducing variables at an early stage hinders further development of a theory for communicating supervisors by making the notations unnecessarily cumbersome. This led the authors to propose a more abstract mathematical framework for coding control-related information, which readily lends itself to the concrete implementation of decentralized supervisors by (boolean) variables at a later stage.

In this paper we assume that a centralized supervisor is already designed using SCT. An Agent-wise Labeling Map (ALM) is then introduced to label each state of the centralized supervisor with disjoint sets of integer tuples. Component \( i \) of the tuples encode the \( i \)th decentralized supervisor’s observation of the centralized supervisor’s states. For each event updating and guard functions are defined to specify how tuples should be updated by its occurrence, and to identify
tuples of integers at which it should be enabled, respectively. We refer to a DES equipped with guard and updating maps as a Supervised DES (SDES). The SDES framework is first developed for centralized supervisory control, and then extended to the decentralized case, where each decentralized supervisor assigns a label to encode the states of a centralized supervisor. A transition is then controlled by guarding its event with a set of tuples of labels, where the $i$th component of a tuple is the label assigned by the $i$th decentralized supervisor. Communication is needed to inform a supervisor of the labels assigned by other supervisors, which it uses to reevaluate its guards, and to update its own labels.

The rest of this paper is organized as follows. Section II introduces the SDESs and employs them to implement centralized supervisors for DESs. Section III then generalizes the ideas developed in Section II to the decentralized case, and through the introduction of ALMs tackles the problem of communication among decentralized supervisors. Section IV concludes the paper and makes suggestions for future work.

II. Supervised DESs

In this section Supervised Discrete-Event Systems (SDESs) are introduced as more abstract versions of EFSMs.

A. The formalism of Supervised DESs

**Notation 1** Let a given alphabet $\Sigma$ be partitioned into controllable $\Sigma_c$ and uncontrollable $\Sigma_{uc}$ subalphabets, i.e. $\Sigma = \Sigma_c \cup \Sigma_{uc}$. Also assume a second partitioning into observable $\Sigma_o$ and unobservable $\Sigma_{uo}$ subalphabets is given, i.e. $\Sigma = \Sigma_o \cup \Sigma_{uo}$. Associated with the latter partitioning a natural projection $P$ is defined as $P : \Sigma^* \rightarrow \Sigma^*_o$ which simply erases from $s \in \Sigma^*$ all events in $\Sigma_{uo}$. Assume that a language $L \subseteq \Sigma^*$ is given and denote its prefix-closure by $\overline{L}$.

**Definition 2** A Supervised DES (SDES) $D$ is denoted by a quadruple $D = (\Sigma, L, A, G)$ where
- $\Sigma$ is a finite set of events (alphabet);
- $L$ is a (regular) language defined over $\Sigma$, i.e. $L \subseteq \Sigma^*$;
- $A : \Sigma \times \mathbb{N} \rightarrow \mathbb{N}$ is called an updating function;
- $G : \Sigma \rightarrow \text{pwr}(\mathbb{N})$ is a guard function.
Map \( A \) is extended to \( A : \Sigma^* \times \mathbb{N} \to \mathbb{N} \) according to: for \( v \in \mathbb{N} \), \( A(\epsilon, v) = v \), and for \( s \in \Sigma^* \) and \( \sigma \in \Sigma \), \( A(s\sigma, v) = A(\sigma, A(s, v)) \). An SDES is thus a language equipped with two mappings. The closed and marked languages associated with an SDES are defined below.

**Definition 3** The closed and marked languages of an SDES \( D = (\Sigma, L, A, G) \) are denoted by \( L(D) \) and \( L_m(D) \), respectively, and are defined recursively as follows: \( \epsilon \in L(D) \) and for all \( s \in \Sigma^*, \sigma \in \Sigma \)

\[
\begin{align*}
&s\sigma \in L(D) \iff s \in L(D) \land s\sigma \in \overline{L} \land A(s, 0) \in G(\sigma), \\
&L_m(D) = L(D) \cap L.
\end{align*}
\]

The semantics of \( D \) is as follows: to each string \( s \in \Sigma^* \) a label \( A(s, 0) \) is attached. Thus, starting recursively from \( \epsilon \), if \( s \) is in the behavior of \( D \) and \( \sigma \in \Sigma \) is eligible in \( L \) after \( s \) (i.e. \( s\sigma \in \overline{L} \)), then \( \sigma \) is “enabled” if the label of \( s \) is in the image of \( \sigma \) under the guard function, i.e. \( A(s, 0) \in G(\sigma) \). When \( \sigma \) is taken, the label of \( s\sigma \) is computed according to \( A(s\sigma, 0) = A(\sigma, A(s, 0)) \).

From Definition 3, the behavior of an SDES \( D = (\Sigma, L, A, G) \) is a subset of the language \( L \). An SDES is obtained by guarding events, i.e. limiting their occurrence, based on the observation of event sequences of the guarded language. Thereby, an SDES is equipped with means to control and observe a given behavior \( L \); in other words, an SDES may be used to implement the control decisions of an already designed supervisor for \( L \), and correspondingly it is suitable to model a closed-loop DES.

**B. SDESs and centralized supervision**

Assume that the plant is modeled by an automaton \( G = (Q, \Sigma, \delta, q_0, Q_m) \), where \( Q \) is the finite set of states, \( q_0 \) is the initial state, \( Q_m \) is the set of marked states, and \( \delta : Q \times \Sigma \to Q \) is the partial transition function. Let \( E \subseteq L_m(G) = L \) be a given specification language. Denote by \( \Gamma = \{ \gamma \in Pwr(\Sigma) \mid \gamma \supseteq \Sigma_w \} \) the set of all control patterns. A map \( W : L(G) \to \Gamma \) such that \( \ker(P \mid L(G)) \leq \ker(W) \) is called a feasible supervisory control for \( G \).

**Theorem 1** ([1], Theorem 6.3.1) Let \( \emptyset \neq E \subseteq L_m(G) \). There exists a nonblocking feasible supervisory control \( W \) for \( G \) such that \( L_m(W/G) = E \) if and only if \( E \) is (i) controllable with respect to \( G \), (ii) observable with respect to \( (G, P) \), and (iii) \( L_m(G) \)-closed. \( \blacksquare \)
We assume that all specifications are subsets of $L_m(G)$, controllable, observable, and $L_m(G)$-closed. These assumptions guarantees the existence of a proper ([11], Chapter 3) supervisor $S = (R, \Sigma, \xi, r_0, R_m)$ for $G$, such that when running in parallel with the plant, the language of the composition is equal to the specification language $E$, i.e. $L_m(G) \cap L_m(S) = E$. The interaction between $S$ and $G$ can be formulated using SDESs in the following way.

**Problem 4 Control Problem for SDESs:** Let $G$ be a plant, $E \subseteq L_m(G)$ be the regular language of a specification satisfying the conditions of Theorem 1, and $S$ be an already designed admissible, feasible and proper supervisor for $G$ such that $L_m(S) \cap L_m(G) = E$. Design updating and guard functions $A$ and $G$ for SDES $D = (\Sigma, L_m(G), A, G)$ such that $L(D) = E$ and $L_m(D) = E$.

□

In other words, we seek to design mappings $A$ and $G$ which implement a given supervisor. Let $S = (R, \Sigma, \xi, r_0, R_m)$. First, a labeling map is introduced to encode the states of $S$.

**Definition 5** A map $\ell : R \rightarrow \mathbb{N}$ is called a Global Labeling Map (GLM) if

1) $\ell(r_0) = 0$, and
2) $\forall r, r' \in R, \ell(r) = \ell(r') \implies r = r'$.

□

Note that a GLM yields a finite image when applied to a finite automaton.

When a GLM is employed to encode the states of a finite deterministic automaton, its transition structure may be equivalently represented by updating functions, as suggested by Lemma 2, the proof of which can be found in the Appendix.

**Lemma 2** Let $S$ be a centralized supervisor whose closed language is denoted by $L(S)$ and $\ell(.)$ be a GLM labeling the states of $S$. Define $A$ according to

$$\forall r, r' \in R, \forall \sigma \in \Sigma. \quad \xi(r, \sigma) = r' \implies A(\sigma, \ell(r)) = \ell(r')$$

Then \hspace{1cm} $\forall s \in L(S), \forall r \in R. \quad r = \xi(r_0, s) \iff \ell(r) = A(s, 0).$

□

A solution to Problem 4 may now be obtained as follows.

**Proposition 3** Let $G$ and $S$ be as before, and $\ell$ be a GLM. Define $D = (L_m(G), \Sigma, A, G)$, where $A$ is defined as in Lemma 2 and $G$ is defined as:
∀σ ∈ Σ, \[ \mathcal{G}(\sigma) = \{ \ell(r) \mid r \in R \land \xi(r, \sigma)! \} \] (2)

Then \( L(D) = \overline{E} \) and \( L_m(D) = E \).

**Proof:** Please refer to the Appendix. ■

By Proposition 3 the maximal permissiveness of the supervisor \( S \), guaranteed by SCT, is inherited by its implementing SDES, too. Since by SCT no uncontrollable event may be disabled by a supervisor, the image of such an event under guard function may always be taken to be equal to \( \cup_{r \in R} \ell(r) \), i.e. it is always enabled. It is then the plant whose behavior may limit the occurrence of any such event. From now on we implicitly assume that \( \mathcal{G}(\sigma) = \cup_{r \in R} \ell(r) \) for uncontrollable \( \sigma \), and specify \( \mathcal{G}(\sigma) \) for controllable events only. Similarly, a supervisor designed by SCT makes no state change upon the occurrence of an unobservable event. Hence we always assume that \( A(\sigma, \cdot) = id_N \) for an unobservable event \( \sigma \), where \( id_N : \mathbb{N} \to \mathbb{N} : n \mapsto n \) denotes the identity function, and specify \( A(\sigma, \cdot) \) for observable events only. Note that once computed for the SDES, the guard and updating functions can be expressed as boolean formulas and functions by implementing state labels using boolean variables.

**Example 1** Figure 1 shows plant \( G \) which is defined over \( \Sigma = \{ \alpha, \beta, \gamma \} \), where \( \Sigma_o = \{ \alpha \} \) and \( \Sigma_c = \{ \gamma \} \). Also given in the same figure is a specification \( E \) which is \( L_m(G) \)-closed and its controllability and observability with respect to \( G \) can be easily verified. Computed using SCT, \( S \) is a minimally restrictive supervisor which enforces \( L_m(E) \). The states of \( S \) are encoded as 0 and 1, based on which the guard and updating functions (for controllable and observable events, respectively) are computed as follows:

\[
\mathcal{G}(\gamma) = \{ 1 \}, \quad A(\alpha, m) = \begin{cases} 1; & \text{if } m = 0, \\ \text{arbitrary}; & \text{if } m \neq 0. \end{cases}
\]

Note that \( \alpha \) is ineligible in \( S \) when \( m = 1 \). Since values of \( m > 1 \) can never be reached, for \( m \neq 0 \), \( A(\alpha, m) \) can be defined arbitrarily, say, set to 1, in order to simplify the boolean function in its implementing EFSM.

♦
III. DISTRIBUTED SDESs AND DECENTRALIZED SUPERVISORS

This section extends the SDES formalism to the case of decentralized supervisors for controlling distributed plants or concurrent languages.

**Notation 6** We use the following notations. Consider a network consisting of distributed sensors and actuators as means to observe and control, respectively, the plant’s behavior $L \subseteq \Sigma^*$ by $n$ supervisors. Associated with each supervisor $i \in I = \{1, 2, \ldots, n\}$ in the network define observable and controllable event subsets $\Sigma_{o,i}$ and $\Sigma_{c,i}$, respectively, where $\Sigma_{o,i}, \Sigma_{c,i} \subseteq \Sigma$. Thereby, the $i$'th supervisor observes plant’s behavior through its observational window, modelled by the natural projection $P_i : \Sigma^* \rightarrow \Sigma_{o,i}^*$, and exercises control on events in $\Sigma_{c,i}$. Thus, from the viewpoint of the $i$'th supervisor we have $\Sigma_{uo,i} = \Sigma \setminus \Sigma_{o,i}$ and $\Sigma_{uc,i} = \Sigma \setminus \Sigma_{c,i}$. Define $\Sigma_i = \Sigma_{c,i} \cup \Sigma_{o,i}$. Associated with each event $\sigma$ denote by $I_o(\sigma)$ and $I_c(\sigma)$ the sets of all sensors (respectively, actuators) which can observe (respectively, control) $\sigma$, i.e. $I_o(\sigma) = \{i \in I \mid \sigma \in \Sigma_{o,i}\}$ and $I_c(\sigma) = \{i \in I \mid \sigma \in \Sigma_{c,i}\}$. We define a centralized supervisor to be one which has access to all sensors’ observations and can exercise control over all controllable events. For this supervisor we define $\Sigma_c = \bigcup_{i \in I} \Sigma_{c,i}$, $\Sigma_o = \bigcup_{i \in I} \Sigma_{o,i}$, $\Sigma_{uo} = \Sigma \setminus \Sigma_o$, $\Sigma_{uc} = \Sigma \setminus \Sigma_c$, and $P : \Sigma^* \rightarrow \Sigma_o^*$. Denote by $\underline{v} = (v_1, \ldots, v_n) \in \mathbb{N}^n$ a tuple of $n$ natural numbers which is sometimes represented by $(v_i, v_{-i})$ to emphasize on its $i$'th component, $v_i$, where $v_{-i} \in \mathbb{N}^{n-1}$ is the $(n - 1)$-tuple obtained by removing $v_i$ from $\underline{v}$. Let $\underline{0}$ denote a tuple of $n$ zeros. Consider a map $\pi_i : \mathbb{N}^n \rightarrow \mathbb{N}$ such that $\pi_i(v) = v_i$ which picks the $i$'th component of $v$, and extend $\pi_i$ to a map $\text{pwr}(\mathbb{N}^n) \rightarrow \text{pwr}(\mathbb{N})$. For $\underline{v}, \underline{v}' \in \mathbb{N}^n$, we say $\underline{v}$ is an $i$-sibling of $\underline{v}'$ if $v_i \neq v_i'$ and $v_{-i} = v'_{-i}$.

A distributed SDES is defined as follows.

![Diagram](image)
Definition 7 A Distributed SDES (DSDES) is denoted by $\mathcal{D} = \{\mathcal{D}_i\}_{i \in I}$, where each quadruple $\mathcal{D}_i = (\Sigma, L, \mathcal{A}_i, \mathcal{G}_i)$ is defined as follows.

- $\Sigma$ is a finite set of events (alphabet);
- $L$ is a (regular) language defined over $\Sigma$, i.e. $L \subseteq \Sigma^*$;
- $\mathcal{A}_i : \Sigma_i \times \mathbb{N}^n \rightarrow \mathbb{N}$ is an updating function;
- $\mathcal{G}_i : \Sigma_i \rightarrow \mathbb{P}(\mathbb{N}^n)$ is a guard function.

For convenience we extend the domain of $\mathcal{A}_i$ and $\mathcal{G}_i$ to the alphabet of all events. Define $\hat{\mathcal{A}}_i : \Sigma \times \mathbb{N}^n \rightarrow \mathbb{N}$ and $\hat{\mathcal{G}}_i : \Sigma \rightarrow \mathbb{P}(\mathbb{N}^n)$ according to:

\[
\hat{\mathcal{A}}_i(\sigma, v) = \begin{cases} 
\mathcal{A}_i(\sigma, v) & \sigma \in \Sigma_i \\
\pi_i(\sigma) & \sigma \notin \Sigma_i 
\end{cases}, \quad \hat{\mathcal{G}}_i(\sigma) = \begin{cases} 
\mathcal{G}_i(\sigma) & \sigma \in \Sigma_i \\
\mathbb{N}^n & \sigma \notin \Sigma_i 
\end{cases}
\]  

(3)

With a slight abuse of notation, we shall use $\mathcal{A}_i$ and $\mathcal{G}_i$ to denote $\hat{\mathcal{A}}_i$ and $\hat{\mathcal{G}}_i$, respectively. Define a map $\mathcal{A} : \Sigma^* \times \mathbb{N}^n \rightarrow \mathbb{N}^n$ recursively as

\[
\forall v \in \mathbb{N}^n, \forall s \in \Sigma^*, \forall \sigma \in \Sigma. \quad \mathcal{A}(\epsilon, v) = v; \quad \mathcal{A}(s\sigma, v) = \left(\mathcal{A}_i(\sigma, \mathcal{A}(s, v))\right)_{i \in I}.
\]  

(4)

Definition 8 The closed and marked languages of $\mathcal{D}_i$ are denoted by $L(\mathcal{D}_i)$ and $L_m(\mathcal{D}_i)$, respectively, and are defined recursively as follows: $\epsilon \in L(\mathcal{D}_i)$ and for all $s \in \Sigma^*$ and $\sigma \in \Sigma$,

\[
s\sigma \in L(\mathcal{D}_i) \iff s \in L(\mathcal{D}_i) \wedge s\sigma \in L \wedge \mathcal{A}(s, \sigma) \in \mathcal{G}_i(\sigma),
\]

$L_m(\mathcal{D}_i) = L(\mathcal{D}_i) \cap L$.

The closed and marked languages of a DSDES $\mathcal{D} = \{\mathcal{D}_i\}_{i \in I}$ are denoted by $L(\mathcal{D})$ and $L_m(\mathcal{D})$, respectively, and are defined as follows:

$L(\mathcal{D}) = \cap_{i \in I} L(\mathcal{D}_i), \quad L_m(\mathcal{D}) = \cap_{i \in I} L_m(\mathcal{D}_i)$.

Problem 9 Control problem for DSDESs: Let the plant be modeled by an automaton $G = (Q, \Sigma, \delta, q_0, Q_m)$ and $E$ be a specification satisfying the conditions of Theorem 1, enforced by a
proper, feasible and admissible centralized supervisor $S = (R, \Sigma, \xi, r_0, R_m)$. Design guard and updating functions for each $D_i = (\Sigma, L_m(G_i), A_i, G_i)$ such that $L(\mathcal{D}) = \overline{E}$ and $L_m(\mathcal{D}) = E$. □

In what follows we investigate a solution to Problem 9. To begin with, note that if $E$ can be decomposed into $n$ component specifications, each defined over the subalphabet $\Sigma_i$, then the problem would be reduced to $n$ independent monolithic designs, which can be done using the method of Subsection II-B. Also, if the specification is coobservable, there exist $n$ decentralized supervisors $S_i$, $i \in I$, each partially observing the plant’s behavior directly through $P_i$ and exercising control over events in $\Sigma_{c,i}$, such that their concurrent operation enforces the specification. In this case, separate application of the method of Subsection II-B to each $S_i$ would result in guard and updating functions which depend only on the labels associated with the states of $S_i$. It can be shown that this set of independent guard and updating functions would solve the problem, too. Since the focus of the present work is to develop the SDES framework to model communication among decentralized supervisors, we assume that the specification language is not coobservable either.

**Assumption 10** Specification $E$ in Problem 9 is neither decomposable nor coobservable w.r.t. $G$ and $P_i$, $(i \in I)$ [2].

We start by introducing a labeling map which encodes the states of $S$.

A. Agent-wise labeling maps

In accordance with the distributed nature of the system, the labeling scheme to be employed for the centralized supervisor, $S = (R, \Sigma, \xi, r_0, R_m)$, should reflect the observations of the plant’s behavior from the viewpoint of “component supervisors,” i.e. it should be structured as $n$ local observations. To define such labeling schemes, the selfloops of $S$ should be modified in the following way (see [15], Remark 1): For an event, say $\alpha_i$, which is selflooped in one state, say $r_1$, and causes a state change in another state, say $r_2$, a state $\hat{r}_1$ is added to $S$ which inherits all the outgoing non-selfloop transitions of $r_1$, all selfloops at $r_1$ labeled with events in $\Sigma_{loop} = \Sigma_{uo} \cup \{\sigma \in \Sigma_o | \forall r, r' \in R. r' = \xi(r, \sigma) \Rightarrow r = r'\}$, and $r_1$’s marking, while all selfloop transitions at $r_1$ which are not labeled by events in $\Sigma_{loop}$ are replaced with transitions with the same labels from $r_1$ to $\hat{r}_1$ and vice versa. The labeling maps enjoying the desired properties are characterized below.
Definition 11 Let \( S = (R, \Sigma, \xi, r_0, R_m) \) be a centralized supervisor. An Agent-wise Labeling Map (ALM) is a map \( \ell : R \rightarrow pwr(\mathbb{N}^n) \) with the following properties:

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

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

Theorem 4 in [15] provides a constructive proof for the existence of an efficiently computable finite ALM for every centralized supervisor based on the structure of a Latin hypercube of dimension \( n \) and side \( m \) (the size of \( R \)). Employing other structural information of \( S \), such as coobservability of the specification with respect to a partitioning of supervisors, might help reduce the side of the Latin hypercube (see Example 2). This issue is currently under investigation.

B. Updating functions

The existence of a finite ALM paves the way for defining the updating functions associated with a DSDES \( \{D_i\}_{i \in I} \). The key point is the existence of a unique \( i \)-sibling in \( \ell(r') \) of a \( \nu \in \ell(r) \) whenever there is a \( \sigma \in \Sigma_o,i \) such that \( r' = \xi(r, \sigma) \). This is formalized in the following lemma whose proof directly follows from Definition 11.

Lemma 4 Let \( \ell : R \rightarrow pwr(\mathbb{N}^n) \) be a finite ALM for \( S = (R, \Sigma, \xi, r_0, R_m) \). Then there exists a map \( \mu : \Sigma \times \mathbb{N}^n \rightarrow \mathbb{N}^n \) that is consistent with the labeling of \( \ell \), i.e. for all \( r, r' \in R, \sigma \in \Sigma, \nu \in \mathbb{N}^n \) we have:

\[
\nu \in \ell(r) \land r' = \xi(r, \sigma) \Rightarrow \mu(\sigma, \nu) \in \ell(r') \land [\forall i \in I_o(\sigma), \pi_i(\mu(\sigma, \nu)) \neq \pi_i(\nu)] \\
\land [\forall j \in I \setminus I_o(\sigma), \pi_j(\mu(\sigma, \nu)) = \pi_j(\nu)].
\]

Now, the updating functions can be defined using the map \( \mu \) so that each supervisor only updates the label components it can observe:

\[
\forall r, r' \in R, \forall \sigma \in \Sigma. r' = \xi(r, \sigma) \land \nu \in \ell(r) \Rightarrow A_i(\sigma, \nu) = \pi_i(\mu(\sigma, \nu)). \tag{5}
\]

This formula relates \( A \) in (4) to the transition structure of \( S \) as shown next.
Lemma 5 Let \( S = (R, \Sigma, \xi, r_0, R_m) \) be a centralized supervisor and \( \ell : R \rightarrow \text{pwr}(N^n) \) be an ALM. We have:

\[
\forall s \in L(S), \forall r \in R. \quad A(s, 0) \in \ell(r) \iff r = \xi(r_0, s).
\]

Proof: Please refer to the Appendix.

The following provides a solution to Problem 9.

Proposition 6 Let \( G \) be a plant, \( S = (R, \Sigma, \xi, r_0, R_m) \) be a centralized supervisor enforcing some specification \( E \), and \( \ell : R \rightarrow \text{pwr}(N^n) \) be a finite ALM for \( S \). Define \( D = \{D_i\}_{i \in I}, D_i = (L_m(G), \Sigma, G_i, A_i) \) for all \( i \in I \), where the maps \( A_i \) are defined as in (5), and the maps \( G_i \) are defined as\(^1\):

\[
\forall \sigma \in \Sigma, \quad G_i(\sigma) = \begin{cases} \{\ell(r) \mid r \in R \land \xi(r, \sigma)\} & \text{if } \sigma \in \Sigma_{c,i} \\ \cup_{r \in R} \ell(r) & \text{if } \sigma \in \Sigma_{uc,i}. \end{cases}
\]

Then \( L(D) = \overline{E} \) and \( L_m(D) = E \).

Proof: Please refer to the Appendix.

C. Communication as reevaluation of guard and updating functions

In DSDES framework, communication among decentralized supervisors is needed for reevaluation of their guard and updating functions. Assume that the tuple of values after a string \( s \) is observed is \( v := A(s, 0) \). Then \( \sigma \in \Sigma_i \) is enabled at \( s \) if and only if \( v \in G_i(\sigma) \). To determine if this is the case, supervisor \( i \) needs to receive the value \( v_j \), for all \( j \neq i \), from supervisor \( j \). When \( \sigma \) is taken, supervisor \( i \) updates \( v_i \) with the value \( A_i(\sigma, v) \). Again, to correctly evaluate \( A_i(\sigma, v) \), supervisor \( i \) needs to receive the value \( v_j \), for all \( j \neq i \), from supervisor \( j \). If the labels are implemented by boolean variables as in the EFSM framework, the exchange of label values is reduced to the communication of bits (see [15] and [13]). Example 2 illustrates the procedure.

Example 2 Figure 2-a shows the model of a distributed network consisting of three plant components and four events \( \alpha_1, \alpha_2, \alpha_3 \) and \( \beta \), where \( \Sigma_{c,1} = \Sigma_{o,1} = \{\alpha_1, \beta\}, \Sigma_{c,2} = \Sigma_{o,2} = \)

\(^1\)The maps \( \pi_i \) and \( A_i(\sigma, \cdot) \) are extended to \( \text{pwr}(N^n) \) in the obvious way.

11
\( \{\alpha_2, \beta\} \), \( \Sigma_{c,3} = \Sigma_{o,3} = \{\alpha_3\} \). The specification \( S \) is shown in part (b) of the same figure, where all states are marked. Since \( \Sigma = \Sigma_o = \Sigma_c \) and \( S \) is \( L_m(G) \)-closed, the specification meets the conditions of Theorem 1 and thus \( S \) is a proper centralized supervisor, too. Following the explanations provided in Subsection III-A, to define an ALM for \( S \) we unfold the selfloop at state \( r_2 \) since its event (\( \alpha_2 \)) causes a state change at \( r_1 \) and \( r_3 \), resulting in the new finite deterministic automaton \( \hat{S} \) shown in part (c).

While the proof of Theorem 4 in [15] suggests looking up the labels within a Latin hypercube of side 5 and dimension 3, we notice that, in the transition structure of \( \hat{S} \), event \( \alpha_3 \) merely takes the system from a state in the set \( A = \{r_0, r_1\} \) to a state in the set \( B = \{r_2, r_3, r_4\} \); this can be represented by a function \( f : A \rightarrow B \) where \( f(r_0) = r_2 \) and \( f(r_1) = r_3 \). Moreover, within each set the state transitions are labeled with events in \( \Sigma_{o,1} \cup \Sigma_{o,2} \) such that for \( r, r' \in A, \hat{r} \in B \) and \( \sigma \in \Sigma_{o,1} \cup \Sigma_{o,2} \), if \( \hat{r} = f(r), r' = \xi(r, \sigma) \) and \( \xi(\hat{r}, \sigma)! \), then \( \xi(\hat{r}, \sigma) = f(r') \). This symmetry between the two sets of states allows us to choose the required labels from two Latin squares of side 3 whose elements are 3-siblings. This arrangement is shown in part (d), based on which the states’ labels are as follows (A point \((a, b, c) \in \mathbb{N}^3\) is denoted by ‘abc’):

\[
\begin{align*}
\ell(r_0) &= \{000, 210, 120\}, \ell(r_1) = \{100, 010, 220\}, \ell(r_2) = \{001, 211, 121\}, \\
\ell(r_3) &= \{101, 011, 221\}, \ell(r_4) = \{201, 021, 111\}.
\end{align*}
\]

Correspondingly, guards are computed as follows:

Correspondingly, guards are computed as follows:

Fig. 2. (a) The plant’s model. (b) The specification. (c) The modified centralized supervisor. (d) The ALM-assigned labels; a point \( v \in \mathbb{N}^3 \) is labeled with \( r \in R \) iff \( v \in \ell(r) \); thus, for example, \( \ell(r_4) = \{(201), (111), (021)\} \).
### TABLE I

Updating functions \((a, b \in \{0, 1, 2\}, c \in \{0,1\})\).

<table>
<thead>
<tr>
<th>(v)</th>
<th>00c</th>
<th>21c</th>
<th>12c</th>
<th>201</th>
<th>111</th>
<th>021</th>
<th>else</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mathcal{A}(\alpha_1, v))</td>
<td>10c</td>
<td>01c</td>
<td>22c</td>
<td>101</td>
<td>011</td>
<td>221</td>
<td>arbit.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(v)</th>
<th>01c</th>
<th>10c</th>
<th>22c</th>
<th>021</th>
<th>111</th>
<th>201</th>
<th>001</th>
<th>121</th>
<th>211</th>
<th>else</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mathcal{A}(\alpha_2, v))</td>
<td>00c</td>
<td>12c</td>
<td>21c</td>
<td>001</td>
<td>121</td>
<td>211</td>
<td>021</td>
<td>111</td>
<td>201</td>
<td>arbit.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(v)</th>
<th>10c</th>
<th>01c</th>
<th>22c</th>
<th>else</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mathcal{A}(\beta, v))</td>
<td>21c</td>
<td>12c</td>
<td>00c</td>
<td>arbit.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(v)</th>
<th>ab0</th>
<th>else</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mathcal{A}(\alpha_3, v))</td>
<td>ab1</td>
<td>arbit.</td>
</tr>
</tbody>
</table>

\[
\mathcal{G}_1(\alpha_1) = \ell(r_0) \cup \ell(r_2) \cup \ell(r_4), \quad \mathcal{G}_2(\alpha_2) = \ell(r_1) \cup \ell(r_2) \cup \ell(r_3) \cup \ell(r_4), \\
\mathcal{G}_1(\beta) = \mathcal{G}_2(\beta) = \ell(r_1) \cup \ell(r_3), \quad \mathcal{G}_3(\alpha_3) = \ell(r_0) \cup \ell(r_1).
\]

The updating functions are listed in Table I. “Arbitrary” cases are chosen later on so as to simplify the derivation of guards and updating functions of the EFSM implementation.

To implement the guards and updating functions in the EFSM framework, we observe that the first, second and third components of a label can be \(\{0, 1, 2\}\), \(\{0, 1, 2\}\) and \(\{0, 1\}\), respectively. Thus, supervisors 1, 2, and 3 would require respectively \(\lceil \log_2(3) \rceil = 2\), \(\lceil \log_2(3) \rceil = 2\), and \(\lceil \log_2(2) \rceil = 1\) boolean variables to implement their labels. Following the notations used in [15], denote by \(X_i\) the set of private variables of the \(i\)'th supervisor, and let \(X_1 = \{x_1, y_1\}\), \(X_2 = \{x_2, y_2\}\) and \(X_3 = \{y_3\}\), where a label \((l_1, l_2, l_3)\) in the range of interest is represented by the value of \((x_1y_1, x_2y_2, y_3)\) (for instance, \(211\) is represented by \((x_1y_1, x_2y_2, y_3) = (10, 01, 1)\)).

Next, the guard and updating functions are implemented as boolean formulas and functions over these five variables. In compliance with the notations used in the EFSM formalism (see [14] and [15]), for the \(i\)'th supervisor \(g_i(\sigma)\) denotes the guard formula associated with event \(\sigma\), and \(a_i(x, \sigma)\) denotes the updating function associated with private variable \(x\) upon the occurrence of event \(\sigma\). We notice that the ultimate goal of such a representation is to formulate communication as the exchange of the values of private variables of supervisors amongst themselves to reevaluate the guards and updating functions. Hence, supervisor \(i\) needs to receive less communication if a
smaller number of private variables of other supervisors appear in the expressions of its guards and updating functions. To arrive at the most simplified forms (in the above sense), we make the following observations:

- Two variables in general can represent four integers. However, in the case of supervisors 1 and 2, only three labels, i.e. 0, 1, and 2, are used. The unused label 3 may be used arbitrarily to simplify the functions (see “don’t care” conditions in [16]).

- The table of updating functions can be simplified by proper use of arbitrary cases. For example, if the (now arbitrary) images of 200, 110 and 020 under $A_1(\alpha_1, \cdot)$ are chosen to be equal to those of 201, 111 and 021, respectively, then $A_1(\alpha_1, \cdot)$ will become independent of the label assigned by the 3rd supervisor. In a similar way, $A_2(\alpha_2, \cdot)$ can be made independent of the label assigned by the 3rd supervisor. Also, clearly the updates of supervisor 3 are independent of the labels assigned by supervisors 1 and 2 (see Table I), and hence supervisor 3’s updating function is independent of the their four private variables.

- The guard associated with $\alpha_1$ can be simplified by using the symmetry induced by adding labels 020, 110, and 200 to $G_1(\alpha_1)$. This is possible because any such tuple of values, $\bar{v}$, encode a non-existing state, which is the counterpart of $r_4$ in set $A$, but is not “reachable” (i.e. there is no $\sigma \in \Sigma$, $s \in L(S)$, $i \in \{1, 2, 3\}$ such that $A(\sigma, A(s, 0)) = \bar{v}$ $\wedge$ $A(s, 0) \in G_i(\sigma)$); thus it can be safely added to a guard set to simplify the expression for guard formula. Doing so, $g_1(\alpha_1)$ becomes independent of the supervisor 3’s private variable.

Accordingly the guards and updating functions are obtained as in Table II.

With guards and updating functions defined as above, the following observations can be made about communication among the decentralized supervisors:
• Supervisor 3’s guard and updating function depend only on its own private variable, $y_3$, hence it does not require other supervisors’ information.

• On the other hand, supervisor 1’s guards and updating functions depend on $x_2$ and $y_2$ in addition to its own private variables in $X_1$. Thus, supervisor 2 has to communicate the values of $x_2$ and $y_2$ to supervisor 1, so that supervisor 1 can correctly reevaluate its guards and updating functions.

• While supervisor 2’s updating functions depend on the variables in $X_1$ (in addition to its own private variables), $\alpha_2$ is guarded with a formula that depends on $y_3$ as well. This implies the need for communication from supervisors 3 and 1 to supervisor 2.

• For a common event such as $\beta$, the guards associated with all supervisors which can exercise control over it are the same, i.e. all such supervisors make consistent control decisions. As a result, it is sufficient to supply the updated values just to one via communication. Clearly, this supervisor is chosen so that the overall communication content is minimal.

Although no explicit mention of the “time” of communication is made, it is implicit that communication takes place when guards and updating functions need to be reevaluated. Analysis of guards and updating functions can lead to the logical order and instant of receiving the updated values of the required variables. This issue is currently under investigation.

In [15], a comparison is made between the EFSM approach to communication among decentralized supervisors and approaches based on state estimates [8] and possible worlds [9]. Among the advantages of the approach followed in this paper are applicability to every centralized supervisor designed using supervisory control theory, use of (agent-wise) labeling as an integral part of the design, a practical bit-wise approach to the communication content, and a compact implementation of the supervisor by encoding control and observation using integer labels. As a high-level abstraction of EFSMs, (D)SDESs enjoy the same advantages, and moreover, provide a rigorous framework for establishing the properties of communicating decentralized supervisors.

IV. CONCLUSION AND FUTURE WORK

We propose a framework for studying communication among decentralized supervisors of a DES using SDESs. Being an abstract version of an EFSM, an SDES is equipped with guard and updating functions to implement the (already designed) centralized supervisor and model
the closed-loop system. The central tool employed by an SDES is a labeling map, called GLM and ALM in centralized and decentralized cases, respectively, which is shown to exist and can be efficiently computed. Communication among decentralized supervisors is used to reevaluate guard and updating functions. Exploring more properties of the ALMs, finding a finer partitioning of the class of such decentralized supervisors based on the need for reevaluating guard functions, updating functions, or both, and including inference in this framework are among the future extensions of this work.

APPENDIX

Proof of Lemma 2: First note that the function \( A \) is well-defined. Proof is done by induction on the length of strings.

- **Base:**

  \[ A(\epsilon, 0) = \ell(r) \iff 0 = \ell(r) \]

  \[ \iff r = r_0 \quad \text{[Item 1, Defn. 5]} \]

  \[ \iff r = \xi(r_0, \epsilon) \]

- **Inductive step:** Assume that \( s\sigma \in L(S) \). Then \( s \in L(S) \), i.e. there exists \( r \in R \) such that \( r = \xi(r_0, s) \). It follows from the induction assumption that \( A(s, 0) = \ell(r) \iff r = \xi(r_0, s) \). Let \( r' = \xi(r, \sigma) \). We have:

  \[ A(s\sigma, 0) = \ell(r') \]

  \[ \iff A(\sigma, A(s, 0)) = \ell(r') \quad \text{[Defn. of} \ A\text{]} \]

  \[ \iff A(\sigma, \ell(r)) = \ell(r') \quad \text{[Induction assumption]} \]

  \[ \iff r' = \xi(r, \sigma) \quad \text{[Eq. (1), Item 2 of Defn. 5]} \]

  \[ \iff r' = \xi(r_0, s\sigma). \]

Proof of Proposition 3: In the nontrivial case where \( E \) is nonempty the proof is by induction on the length of strings, where the base is trivial since \( \epsilon \in \overline{E} \) and \( \epsilon \in L(D) \). For the inductive
step let $\sigma \in \Sigma$, $s \in \Sigma^*$ and $r = \xi(r_0, s)$. By the induction assumption,

$$ s \in \overline{E} \iff s \in L(D). \quad (7) $$

Then we have

$$ s\sigma \in \overline{E} $$

$$ \iff s \in \overline{E} \land s\sigma \in \overline{L} \land [\exists r \in R. r = \xi(r_0, s) \land \xi(r, \sigma)!] $$

$$ \iff s \in \overline{E} \land s\sigma \in \overline{L} \land [\exists r \in R. \text{A}(s, 0) = \ell(r) \land \xi(r, \sigma)!] \quad \text{[Lem. 5]} $$

$$ \iff s \in L(D) \land s\sigma \in \overline{L} \land \ell(r) \in \mathcal{G}(\sigma) \quad \text{[Eq. (7), Eq. (2)]} $$

$$ \iff s\sigma \in L(D). \quad \text{[Defn. 3]} $$

This proves that $L(D) = \overline{E}$. For the next part by Definition 3 and the fact that $E$ is $L_m(G)$-closed, we have:

$$ L_m(D) = L(D) \cap L_m(G) = \overline{E} \cap L_m(G) = E. $$

\[ \text{\blacksquare} \]

**Proof of Lemma 5:** Proof is done by induction on the length of strings.

- **Base:**

$$ \text{A}(\epsilon, 0) \in \ell(r) $$

$$ \iff 0 \in \ell(r) \quad \text{[Eq. (4)]} $$

$$ \iff r = r_0 \quad \text{[Item 1, Defn. 11]} $$

$$ \iff r = \xi(r_0, \epsilon) $$

- **Inductive step:** Assume that $s\sigma \in L(S)$. Then $s \in L(S)$, i.e. there exists $r \in R$ such that $r = \xi(r_0, s)$. It follows from the induction assumption that $\text{A}(s, 0) \in \ell(r) \iff r = \xi(r_0, s)$. Let $r' = \xi(r, \sigma)$. We have:
\[ A(s\sigma, 0) \in \ell(r') \]
\[ \iff \ \Bigl( A_i(\sigma, A(s, 0)) \Bigr)_{i \in I} \in \ell(r') \] [Eq. (4)]
\[ \iff \ \Bigl( \pi_i \mu(\sigma, A(s, 0)) \Bigr)_{i \in I} \in \ell(r') \] [Eq. (5)]
\[ \iff \ \mu(\sigma, A(s, 0)) \in \ell(r') \] [Lem. 4]
\[ \iff \ r' = \xi(r, \sigma) \]
\[ \iff \ r' = \xi(r_0, s\sigma) \]

**Proof of Proposition 6:** In the nontrivial case where \( E \) is nonempty the proof is by induction on the length of strings, where the base is trivial since \( \epsilon \in \overline{E} \) and \( \epsilon \in L(D) \). For the inductive step let \( \sigma \in \Sigma, \ s \in \Sigma^* \) and \( r = \xi(r_0, s) \). By the induction assumption,

\[ s \in \overline{E} \iff s \in L(D). \] (8)

We have:

\[ s\sigma \in \overline{E} \]
\[ \iff s \in \overline{E} \land s\sigma \in \overline{L} \land [\exists r \in R. \ r = \xi(r_0, s) \land \xi(r, \sigma)!] \]
\[ \iff s \in \overline{E} \land s\sigma \in \overline{L} \land [\exists r \in R. \ A(s, 0) \in \ell(r) \land \xi(r, \sigma)!] \] [Lem. 5]
\[ \iff s \in L(D) \land s\sigma \in \overline{L} \land [\exists r \in R. \ A(s, 0) \in \ell(r) \land \xi(r, \sigma)!] \] [Eq. (8)]
\[ \iff s \in \cap_{i \in I} L(D_i) \land s\sigma \in \overline{L} \land A(s, 0) \in \cap_{i \in I} G_i(\sigma) \] [Defn. 8]
\[ \iff \forall i \in I. \ s \in L(D_i) \land s\sigma \in \overline{L} \land A(s, 0) \in G_i(\sigma) \] [Defn. 7]
\[ \iff \forall i \in I. \ s\sigma \in L(D_i) \] [Defn. 8]
\[ \iff s\sigma \in \cap_{i \in I} L(D_i) \]
\[ \iff s\sigma \in L(D). \]

This proves that \( L(D) = \overline{E} \). For the next part we have:

\[ L_m(D) = \cap_{i \in I} L_m(D_i) \] [Defn. 8]
\[ = \cap_{i \in I} (L(D_i) \cap L_m(G)) \] [Defn. 3]
\[ = (\cap_{i \in I} L(D_i)) \cap L_m(G) = \overline{E} \cap L_m(G) \] [Part 1]
which, by $L_m(G)$-closure of $E$, is equal to $E$. ■

REFERENCES


