Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Communication and State Realization in Decentralized Supervisory Control of Discrete-Event Systems ⋆

Amin Mannani * **Peyman Gohari** *

* *ECE Dept., Concordia University, S-EV005.139, 1515 St. Catherine West, Montreal, QC, H4G 2W1, Canada, (Tel: 514-848-2424 Ext. 3100; e-mail: amin_man,gohari@ece.concordia.ca).*

Abstract: This paper continues the authors' previous work on studying the communication among decentralized supervisors for a distributed Discrete-Event System (DES) in the framework of Distributed Supervised DES (DSDES). Given an already available centralized supervisor for a distributed DES, it relates a language property of this supervisor, called *weak joint observability*, to a property of the state-based realization of the supervisor, referred to as the existence of the *Independent Updating Functions* (IUFs). The latter property means that the decentralized implementation of the supervisor relies on each agent's independent observation of the DES dynamic evolution and entails simpler, delay-robust, and possibly cheaper communication; issues currently under investigation. Examples illustrate the applicability of the approach.

1. INTRODUCTION

Supervisory Control of distributed DESs has been a subject of interest with many applications such as analysis and design of communication protocols [Rudie and Wonham, 1990]. In s distributed DES, geographic separation of sites restricts the agents' observation of the whole system' dynamic evolution and this, in turn, makes the satisfaction of a *global specification* a difficult task. Such a specification can be enforced by designing a set of supervisors, one for each site with no communication amongst them, if and only if it is controllable and coobservable [Rudie and Wonham, 1992]. The latter property requires that each agent, which can exercise control over an event, can disambiguate the legality of all the lookalike strings which are ended by this event and determine if they are marked. The class of coobservable specifications, though can be enlarged using some variants of the original definition [Yoo and Lafortune, 2002], still is restricted in many applications, where an agent's own observation is inadequate for the sake of control purposes. In such circumstances, agents need to communicate their knowledge of the system's evolution amongst each other.

Communication, thus, appears as the third capability of a distributed DES, which affects and is affected by the other two, namely control and observation [Rudie et al., 2003]. There have been several attempts to model, formalize, classify, and solve the communication by answering different questions as "*who* communicates to *whom* and *when*", "*what* should be communicated and if this is *minimal*" (see [Teneketzis, 1997], [van Schuppen, 2004], and the references therein). The issue of the "content" of the communication has been approached differently, i.e. by exchanging *event* [Rudie et al., 2003] or *state* observation [Barrett and Lafortune, 2000], [Ricker and Rudie, 1999]. Reasoning that these approaches are too abstract in practice, the authors first proposed a framework based on Extended Finite-State Machines (EFSM) which implements the *state* realization of an already designed centralized supervisor's control and observation maps in a *decentralized* manner by designing guard formulas and updating functions, respectively [Mannani et al., 2006]. These are defined on a set of binary variables which is itself a union of disjoint subsets, each assigned to an agent. Communication in this framework naturally arises as reevaluation of guards and updating functions, measured in a *bitwise* manner. This practical advantage is not the only merit of the EFSM framework; finer classification in terms of "communication for control" (i.e. reevaluation of guards) and "communication for observation" (reevaluation of updating functions) is gained, too. Such advantages motivated the authors to generalize the EFSM framework to the Distributed Supervised DES (DSDES) framework in [Mannani and Gohari, 2007b], which, while inheriting the two implementing tools, guard and updating functions, employs a set of integer-valued variables. Such choice of variables which are assigned by Agent-wise Labeling Maps (ALMs), while being flexible and taken from any finite field (including the binary one), makes the proofs more rigorous and insightful.

This paper corresponds, within the DSDES framework, *weak joint observability*, a property of the centralized supervisor's language, to the existence of *independent updating functions*, a property of the state realization of the centralized supervisor. Weak joint observability is a variant of the joint observability, introduced in [Tripakis, 2001], which requires that every illegal move be distinguished from a legal move by (at least) one agent. On the other hand, if all agents possess independent updating functions, then upon the occurrence of an event, the agents which observe it can update their variables, i.e. the indicators of their information of the system evolution, independent

⋆ This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

of each other. The main contributions of the paper are formal definitions of an IUF (within DSDES framework) and weak joint observability, showing the necessity of the latter for the existence of the first, and introducing classes of weakly joint observable supervisor's languages for which IUFs can be computed, as justified by some examples.

The paper is organized as follows. Section 2 introduces the notation and the formalism of DSDES, including the ALMs. Then Section 3 formally defines the notions of an IUF and weak joint observability and explores their correspondence. Finally, conclusions are drawn and suggestions for future research are made in Section 4.

2. DISTRIBUTED SDESs

Notation: Throughout this paper we use the following notations. Consider a language $L \subseteq \Sigma^*$, called plant's behavior, or simply plant, and a network consisting of distributed sensors and actuators. These sensors and actuators are the means to observe and control, respectively, the plant's behavior for 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 $P_i : \Sigma^* \to \Sigma_{o,i}{}^*$, and exercises control on events in $\Sigma_{c,i}$. Thus, from the viewpoint of the i'th supervisor we have $\Sigma_{uo,i} = \Sigma \backslash \Sigma_{o,i}$ and $\Sigma_{uc,i} = \Sigma \setminus \Sigma_{c,i}$. Define $\Sigma_i = \Sigma_{c,i} \cup \Sigma_{o,i}$. Associated with each event σ denote by $I_o(\sigma)$ and $I_c(\sigma)$ the sets of all sensors (respectively, actuators) which can observe (respectively, control) σ, 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 $P : \Sigma^* \to \Sigma_o{}^*$. Denote by $\underline{v} = (v_1, \ldots, v_n) \in \mathbb{N}^n$ a tuple of n natural numbers which is sometimes written as (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. Define a map $\pi_i : \mathbb{N}^n \to \mathbb{N}$ such that $\pi_i(\underline{v}) = v_i$ (i.e. it picks the i'th component of $\underline{v}$), and extend π_i to a map $pwr(\mathbb{N}^n) \to pwr(\mathbb{N})$. Finally, the prefix closures of $L \subseteq \Sigma^*$ and $\{s\}$, $s \in \Sigma^*$, are shown by $\overline{L}$ and $\overline{s}$, respectively. To model the case of a distributed DES consisting of component languages $L_i \subseteq \Lambda_i^*$, $i \in I = \{1, 2, \ldots, n\}$, where $\Lambda_i = \Lambda_{c,i} \dot\cup \Lambda_{uc,i} = \Lambda_{o,i} \dot\cup \Lambda_{uo,i}$, we notice that for component i, events in $\Lambda_i \setminus (\Lambda_{c,i} \cup \Lambda_{o,i})$ are neither controllable nor observable. Therefore, we might redefine new alphabets Σ_i by setting $\Sigma_{c,i} = \Lambda_{c,i}$, $\Sigma_{o,i} = \Lambda_{o,i}$, and $\Sigma_i = \Lambda_{c,i} \cup \Lambda_{o,i}$. Language L is defined as the synchronous product of L_i's, i.e. $L = L_1 \| L_2 \| \cdots \| L_n$.

2.1 DSDESs and agent-wise labeling maps

Definition 1. 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.

- Σ is a finite set of events (alphabet);
- L is a (regular) language defined over Σ, i.e. $L \subseteq \Sigma^*$;
- $\mathcal{A}_i : \Sigma_i \times \mathbb{N}^n \to \mathbb{N}$ is an *updating* function;
- $\mathcal{G}_i : \Sigma_i \to pwr(\mathbb{N}^n)$ 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 $\hat{\mathcal{A}}_i : \Sigma \times \mathbb{N}^n \to \mathbb{N}$ and $\hat{\mathcal{G}}_i : \Sigma \to pwr(\mathbb{N}^n)$ according to: for $\sigma \in \Sigma$ and $\underline{v} \in \mathbb{N}^n$,

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

In the natural recursive way, $\hat{\mathcal{A}}_i$ is extended to $\hat{\mathcal{A}}_i : \Sigma^* \times \mathbb{N}^n \to \mathbb{N}$. 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 $\mathcal{A} : \Sigma^* \times \mathbb{N}^n \to \mathbb{N}^n$ recursively as follows: for all $\underline{v} \in \mathbb{N}^n$, $s \in \Sigma^*$, and $\sigma \in \Sigma$

$$\mathcal{A}(\epsilon, \underline{v}) = \underline{v}; \ \mathcal{A}(s\sigma, \underline{v}) = \Big(\mathcal{A}_i(\sigma, \mathcal{A}(s, \underline{v}))\Big)_{i \in I}. \qquad (1)$$

Definition 2. 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 $\sigma \in \Sigma$,

$s\sigma \in L(\mathcal{D}_i) \iff s \in L(\mathcal{D}_i) \wedge s\sigma \in \overline{L} \wedge \mathcal{A}(s, \underline{0}) \in \mathcal{G}_i(\sigma)$
$L_m(\mathcal{D}_i) = L(\mathcal{D}_i) \cap L$.

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:

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

Associated with each index $i \in I$, a DSDES is equipped with guard and updating functions to capture control and observation, respectively. Control for each $\mathcal{D}_i$ is based upon n-tuples of natural numbers; component i of a tuple is updated with $\mathcal{A}_i$.

Problem 3. Control problem for DSDESs: Let the plant be modeled by an automaton $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$ and $E \subseteq L_m(\mathbf{G})$ be a nonempty specification language which is controllable with respect to $\mathbf{G}$, observable with respect to $(\mathbf{G}, P)$, and $L_m(\mathbf{G})$-closed (see [Wonham, 2007], Theorem 6.3.1). Assume that that E is enforced by a proper, feasible and admissible centralized supervisor $\mathbf{S} = (R, \Sigma, \xi, r_0, R_m)$. Design guard and updating functions for each $\mathcal{D}_i = (\Sigma, L_m(G), \mathcal{A}_i, \mathcal{G}_i)$ such that $L(\mathcal{D}) = \overline{E}$ and $L_m(\mathcal{D}) = E$. □

Since the DSDES framework aims at studying *communication* among supervisors, we exclude the cases in which E may be satisfied without communication and assume that E is neither decomposable nor coobservable w.r.t. $\mathbf{G}$ and P_i, $(i \in I)$ [Rudie and Wonham, 1992].

Definition 4. Let $\mathbf{S} = (R, \Sigma, \xi, r_0, R_m)$ be a centralized supervisor. 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 disjoint);
(3) $\forall r, r' \in R, r \neq r',\ \forall \sigma \in \Sigma_o,\ \forall \underline{v} \in \mathbb{N}^n.\quad \underline{v} \in \ell(r) \wedge r' = \xi(r, \sigma)$
$\quad \Longrightarrow\ \exists! \underline{v}' \in \mathbb{N}^n.\ \underline{v}' \in \ell(r') \ \wedge\ [\forall i \in I_o(\sigma).\ v_i \neq v_i'] \ \wedge\ [\forall j \in I \setminus I_o(\sigma).\ v_j = v_j']$.

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

By Theorem 4 in [Mannani and Gohari, 2007a], there exists an efficiently computable finite ALM for every centralized supervisor $\mathbf{S} = (R, \Sigma, \xi, r_0, R_m)$. Let $\ell : R \to pwr(\mathbb{N}^n)$ be a finite ALM for $\mathbf{S}$. There exists a map $\mu : \Sigma \times \mathbb{N}^n \to \mathbb{N}^n$ that is consistent with the labeling of ℓ, i.e.

$\forall r, r' \in R,\ \sigma \in \Sigma,\ \underline{v} \in \mathbb{N}^n, \underline{v} \in \ell(r) \wedge\ r' = \xi(r, \sigma) \Longrightarrow$
$\mu(\sigma, \underline{v}) \in \ell(r')\ \wedge [\forall i \in I_o(\sigma).\ \pi_i(\mu(\sigma, \underline{v})) \neq \pi_i(\underline{v})]$
$\wedge\ [\forall j \in I \setminus I_o(\sigma).\ \pi_j(\mu(\sigma, \underline{v})) = \pi_j(\underline{v})]$.

The updating functions can be defined using the map μ so that each supervisor only updates the label components it can observe, i.e. for all $r, r' \in R$ and $\sigma \in \Sigma$ we have:

$$r' = \xi(r,\sigma) \ \wedge \ \underline{v} \in \ell(r) \Longrightarrow \mathcal{A}_i(\sigma,\underline{v}) = \pi_i(\mu(\sigma,\underline{v})). \quad (2)$$

Upon occurrence of σ, μ does not change the value of any v_j with $j \in I \setminus I_o(\sigma)$. This fact and (2) imply the following.

$$\forall i \in I, \underline{v} \in \mathbb{N}^n, \sigma \in \Sigma \setminus \Sigma_{o,i}. \mathcal{A}_i(\sigma,\underline{v}) = \pi_i(\underline{v}). \quad (3)$$

Formula (2) relates $\mathcal{A}$ to the transition structure of $\mathbf{S}$, i.e.

$$\forall s \in L(\mathbf{S}), \forall r \in R. \quad \mathcal{A}(s,\underline{0}) \in \ell(r) \Longleftrightarrow r = \xi(r_0, s) \quad (4)$$

A solution to Problem 3: Define $\mathcal{D} = \{\mathcal{D}_i\}_{i\in I}$, $\mathcal{D}_i = (L_m(\mathbf{G}), \Sigma, \mathcal{G}_i, \mathcal{A}_i)$ for all $i \in I$, where the maps $\mathcal{A}_i$ are defined as in (2), and the maps $\mathcal{G}_i$ are defined for as follows:

$$\forall \sigma \in \Sigma.\ \mathcal{G}_i(\sigma) = \begin{cases} \{\ell(r) \mid r \in R \wedge \xi(r,\sigma)!\}; & \text{if } \sigma \in \Sigma_{c,i}, \\ \mathbf{N}^n; & \text{if } \sigma \in \Sigma_{uc,i}. \end{cases} \quad (5)$$

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

2.2 Communication among supervisors

The DSDES framework employs *updating* and *guard* functions as two means to capture the observation- and control-related information of a centralized supervisor, respectively. Communication naturally arises to evaluate either the former or the latter which are referred to as "communication for observation" and "communication for control," respectively. To begin with, let the vector of values after a string s is observed be denoted by $\underline{v} := \mathcal{A}(s,\underline{0})$ and denote supervisor i by $\mathbf{S}_i$.

Communication for observation: Upon observing the occurrence of an event σ, supervisor i must update v_i with the value $\mathcal{A}_i(\sigma,\underline{v})$. However, to correctly evaluate $\mathcal{A}_i(\sigma,\underline{v})$, $\mathbf{S}_i$ needs to know the value of $\underline{v}$, and thus may need to receive the value v_j, for all $j \neq i$, from $\mathbf{S}_j$. There are two methods to communicate the required values. According to the first approach, $\mathbf{S}_i$ requests all such supervisors to send it their v_j's. According to the second approach, after $\mathbf{S}_j$ updates the value of v_j, it immediately sends the new value to all the supervisors whose updating functions depend on v_j. In both cases, communication for observation is initiated by a supervisor which *observes* the occurrence of an event and may wait until a need for an updated value arises (the first approach) or choose to communicate immediately after an update (the second approach). Assuming that communication is *instantaneous*, i.e. the system does not execute any transitions while the queries are being answered, once the required information is received, $\mathbf{S}_i$ can unambiguously determine its new value for v_i. This information may be used to update its estimate of the states of $\mathbf{S}$ and find out what control decisions have to be made over events in $\Sigma_{c,i}$.

Communication for control: The objective of *communication for control* is to update the vector of values to determine if it is included in the image of guard functions. Let event α be in $\Sigma_{c,l}$. Then α is enabled at s if and only if $\underline{v} \in \mathcal{G}_l(\alpha)$. To determine if this is the case, $\mathbf{S}_l$ may need to receive the value v_j, for all $j \neq l$, from $\mathbf{S}_j$. Since $\mathbf{S}_l$ is uncertain about the state of the centralized supervisor, and therefore about whether or not α should be disabled, out of the two policies mentioned for "communication for observation," only the second one is applicable in this case; that is, a "communication for control" should always be initiated by supervisors which have updated their label values upon such updates.

According to the above classification, the "order" of communication is as follows. Upon the occurrence of an event, first all supervisors observing it should employ communication in one of the two ways described above to update their values and determine their state estimates. Next, all of them should send their updated values to those whose guards require them. Once labels, and guard and updating functions are specified in DSDES framework, they can be implemented in the EFSM framework by employing binary variables for each agent (see [Yang and Gohari, 2005] and [Mannani et al., 2006]). Thereby, the exchange of label values is reduced to the communication of bits.

3. INDEPENDENT UPDATING FUNCTIONS

3.1 Motivation

In general, behavioral and structural properties of a centralized supervisor as well as the ALM used to label its states determine which type(s) of communication is (are) needed by a solution to Problem 3. The above classification of communication lets us distinguish the class of solutions in which only communication for control is required. We first show, using an illustrative example, the advantage of this class of solutions over solutions that require both types of communication and then characterize such solutions.

Example 1. Consider a centralized supervisor $\mathbf{S}$ which is implemented by a network of 2 supervisors, i.e. $|I| = 2$, by using an ALM ℓ_1 and then in the EFSM framework using sets of boolean variables $X_1 = \{x_1^1, x_1^2\}$ and $X_2 = \{x_2^1, x_2^2\}$. Let $\Sigma_{o,1} = \Sigma_{c,1} = \{\alpha_1\}$, and the updating functions and guard formula associated with supervisor 1 be as follows: $g_1(\alpha_1) = h_{\alpha_1}(x_1^1, x_2^1)$, $a_1(x_1^1, \alpha_1) = f_{\alpha_1,1}(x_1^1, x_1^2)$, and $a_1(x_1^2, \alpha_1) = f_{\alpha_1,2}(x_1^1, x_1^2, x_2^1, x_2^2)$ [1]. Accordingly, upon observing α_1, supervisor 1 ($\mathbf{S}_1$) should update x_1^1 and x_1^2 for which it should have received the values of x_2^1 and x_2^2 from supervisor 2 ($\mathbf{S}_2$). Knowing that $g_1(\alpha_1)$ requires the updated value of x_2^1, $\mathbf{S}_2$ then sends the updated value of x_2^1 to $\mathbf{S}_1$. In summary, communication for observation (respectively, control) would require the transfer of 2 bits (respectively, 1 bit) from $\mathbf{S}_2$ to $\mathbf{S}_1$.

Assume now that there exists another ALM for $\mathbf{S}$ with the following EFSM implementation: $X_1 = \{x_1^1, x_1^2, x_1^3\}$, $X_2 = \{x_2^1, x_2^2\}$, $g_1(\alpha_1) = h'_{\alpha_1}(x_1^1, x_1^3, x_2^1, x_2^2)$, $a_1(x_1^1, \alpha_1) = f_{\alpha_1,1}(x_1^1, x_1^2)$, $a_1(x_1^2, \alpha_1) = f_{\alpha_1,2}(x_1^2, x_1^3)$, and $a_1(x_1^3, \alpha_1) = f_{\alpha_1,3}(x_1^1, x_1^2, x_1^3)$. This new labeling employs a third variable, x_1^3, and its guard depends on two (as opposed to one) variables of $\mathbf{S}_2$, but its updating functions do not depend on $\mathbf{S}_2$'s variables, i.e. it needs no communication for observation. As a result, $\mathbf{S}_1$ can access all information it needs, captured by x_2^1 and x_2^2, though one single communication for control. Moreover, in the first case if the values of x_2^1 and x_2^2 are received in error, x_1^1 and x_1^2 may be updated to incorrect values, thus $\mathbf{S}_1$ cannot trust the values of its private variables thereafter. Retransmission of x_2^1 and x_2^2 would not help $\mathbf{S}_1$ unless it keeps a record of the evolution of its private variables. However, a corrupted communicated for control only affects $g_1(\alpha_1)$ which can be reevaluated upon a retransmission of the correct values. $\Diamond$

The above example, though not being a rigorous analysis, motivates studying of IUFs and state structures and behaviors for which they can be computed.

Definition 5. An updating function $\mathcal{A}_i$ associated with the i'th agent is called an *Independent Updating Function*

[1] Detailed information of supervisor 2 is not important for the purpose of this example and hence is not mentioned.

(IUF) if for all $\sigma \in \Sigma_i$ and $\underline{v}, \underline{v}' \in \mathbb{N}^n$ the followings holds.

$$v_i = v_i' \implies \mathcal{A}_i(\sigma, \underline{v}) = \mathcal{A}_i(\sigma, \underline{v}'). \quad \blacksquare$$

In simple words an IUF $\mathcal{A}_i$ reevaluates the agent i's associated component of $\underline{v}$, i.e. v_i, based on its own current value, v_i, only and it need not know the value of v_{-i}. This fact together with (4) and (3) imply that when updating functions are independent, the current state of the recognizer may be computed by forming an n-tuple of agents' independent observations of the system's behavior.

3.2 Weak joint observability

It turns out that the existence of a DSDES possessing only independent updating functions, depends directly on properties of the closed and marked language of **S**. In particular, this is related to a property of the language of supervisor **S** called *weak joint observability*. This property was originally defined in [Mannani et al., 2006] motivated by a definition of *joint observability* in [Tripakis, 2001] and is extended to the case of marking now. In simple words joint observability requires that within the closed behavior (respectively, marked behavior) of the plant, any legal-illegal pair of strings (respectively, any marked-unmarked pair of strings) can be told apart by at least one agent. To simplify the notation, let us first define two equivalence relations.

Definition 6. Two sequences $s, s' \in \Sigma^*$ are called *observationally equivalent*, denoted by $s \equiv_I s'$, if for all $i \in I$ it holds that $P_i(s) = P_i(s')$. Also define $s \equiv_U s'$ if $P_{uo}(s) = P_{uo}(s')$. □

Denote the equivalence class of $s \in \Sigma^*$ by $[s] = \{s' \mid s' \equiv_I s\}$. Notice that two observationally equivalent sequences are not necessarily *trace equivalent* [Mazurkiewics, 1995] as their projections onto events in Σ_{uo} might be different. A *jointly observable* language is characterized as follows [2].

Lemma 7. (Lemma 3.1, [Tripakis, 2004]). Let L and K be two languages such that $K \subseteq L \subseteq \Sigma^*$. K is called Jointly Observable (JO) with respect to L and $(\Sigma_{o,1}, \cdots, \Sigma_{o,n})$ iff

$$\forall s \in K,\ \forall s' \in L \setminus K,\ \exists i \in I.\ P_i(s) \neq P_i(s'). \quad \blacksquare$$

For control purposes one should differentiate between the closed and marked languages. Having this in mind, Lemma 7 can be equivalently stated as follows.

Lemma 8. Let L and K be two languages such that $K \subseteq L \subseteq \Sigma^*$. K is called Jointly Observable (JO) with respect to L and $(\Sigma_{o,1}, \cdots, \Sigma_{o,n})$ iff

$$\forall s, s' \in \overline{L}.\ s \in \overline{K} \ \wedge\ s \equiv_I s' \implies s' \in \overline{K}. \quad (6)$$

$$\forall s, s' \in L.\ s \in K \ \wedge\ s \equiv_I s' \implies s' \in K. \quad (7)$$

$\blacksquare$

Weak joint observability weakens the requirements of joint observability by asking for the distinguishing property between legal strings and the *minimal-length* illegal strings as defined next. The reason is that, from a control perspective, one cares about the *first* illegal move regardless of any of its feasible future behavior in the plant.

Definition 9. Let L and K be two languages such that $K \subseteq L \subseteq \Sigma^*$. K is called Weakly Jointly Observable (WJO) with respect to L and $(\Sigma_{o,1}, \cdots, \Sigma_{o,n})$ if the following conditions hold.

$$\forall s, s' \in \overline{L},\ \forall \sigma \in \Sigma.$$

$$s\sigma \in \overline{K} \wedge s'\sigma \in \overline{L} \wedge s' \in \overline{K} \wedge s \equiv_I s' \implies s'\sigma \in \overline{K} \quad (8)$$

$$\forall s, s' \in L.\ s \in K \ \wedge\ s \equiv_I s' \implies s' \in K \quad (9)$$

□

It can be readily shown that joint observability implies weak joint observability [Mannani and Gohari, 2007a]. When it is clear from the context, $(\Sigma_{o,1}, \cdots, \Sigma_{o,n})$ will not be mentioned. We finish this subsection by proving a result which is used in Subsection 3.4.

Definition 10. A language $A \subseteq \Sigma^*$ is called *trace-closed* if

$$\forall s, s' \in \Sigma^*.\ s \equiv_I s' \wedge s \equiv_U s' \wedge s \in A \implies s' \in A \quad (10)$$

□

It is shown next that trace-closedness is inherited by K when it is jointly observable with respect to L.

Lemma 11. Let $K \subseteq L \subseteq \Sigma^*$, L be trace-closed, and K be JO with respect to L. Then K is trace-closed.

Proof: Let $s, s' \in \Sigma^*$ be two strings such that $s \equiv_I s'$, $s \equiv_U s'$, and $s \in K$. Then we have

$$[s \in K \implies s \in L] \qquad [K \subseteq L]$$

$$\implies [s \in L \wedge s \equiv_I s' \wedge s \equiv_U s' \implies s' \in L] \qquad [\text{Defn. 10}]$$

$$\implies [s, s' \in L \wedge s \equiv_I s' \wedge s \in K \implies s' \in K], \qquad [(7)]$$

i.e. K is trace-closed. $\blacksquare$

Remark 12. As stated in [Tripakis, 2005], checking joint observability of K is decidable if L is trace-closed, otherwise it is undecidable in general.

3.3 A necessary condition for the existence of IUFs

This subsection establishes one side of the relationship between the language of a centralized supervisor, i.e. a behavior, and its state realization. The result was first shown in [Mannani and Gohari, 2007a] in the EFSM framework and is proved here in DSDES framework and using the improved definitions for the sake of completeness.

Lemma 13. Let the updating functions $\mathcal{A}_i$ be all independent. Then the following holds.

$$\forall i \in I,\ \forall s \in L(\mathbf{S}),\ \forall v \in \mathbb{N}^n.\ \mathcal{A}_i(s, \underline{v}) = \mathcal{A}_i(P_i(s), \underline{v})$$

Sketch of the Proof: By (3), starting from any label $\underline{v}$ of an arbitrary state, the occurrence of any event $\sigma \in \Sigma \setminus \Sigma_{o,i}$ may not change the value of the i'th component of $\underline{v}$, i.e. $\mathcal{A}_i(\sigma, \underline{v}) = \pi_i(\underline{v})$. By Definition 5, any change in the value of this component is independent of the values of other components. This completes the proof. $\blacksquare$

Corollary 14. If the updating functions $\mathcal{A}_i$ are all independent, then all observationally equivalent strings lead to the same state.

Proof: Let $s, s' \in L(\mathbf{S})$ be two observationally equivalent strings, i.e. $\forall i \in I.\ P_i(s) = P_i(s')$. Lemma 13 then implies that $\forall i \in I.\ \mathcal{A}_i(s, \underline{0}) = \mathcal{A}_i(P_i(s), \underline{0}) = \mathcal{A}_i(s', \underline{0})$. Thus by (1) we have $\mathcal{A}(s, \underline{0}) = \mathcal{A}(s', \underline{0})$ and by the uniqueness of labels in Definition 4, the proof is complete. $\blacksquare$

Assumption 1. As can be seen, weak joint observability assumes that $K \subseteq L$. In general, when K and L are respectively taken to be $L_m(\mathbf{S})$ and $L_m(\mathbf{G})$, where **S** and **G** are the supervisor, designed using supervisory control theory, and plant, respectively, this does not hold. However, such a supervisor **S** implements the supervisory control map in the following sense:

$$L_m(\mathbf{S}) \cap L_m(\mathbf{G}) = K \ \wedge\ L(\mathbf{S}) \cap L(\mathbf{G}) = \overline{K}. \quad (11)$$

[2] Lemma 7 was originally introduced as the definition of a jointly observable language in [Tripakis, 2001], but was shown later to be equivalent to a new definition for—what is then called—observable languages in [Tripakis, 2004]. Here, for notational convenience we choose the original definition. While the original definition is for any two languages K and L, we also assume, without loss of generality, that $K \subseteq L$.

Therefore, in the subsequent discussion, we may safely assume that $L_m(\mathbf{S}) \subseteq L_m(\mathbf{G})$. □

Proposition 15. Let $\mathbf{G}$, E, and $\mathbf{S} = (R, \Sigma, \xi, r_0, R_m)$, be as before and $\ell(.)$ be an ALM which labels the states of $\mathbf{S}$ yielding independent updating functions. Then $L_m(\mathbf{S})$ is WJO with respect to $L_m(G)$.

Proof: Let $s, s' \in L(\mathbf{G})$ and $\sigma \in \Sigma$ such that $s\sigma \in L(\mathbf{S})$, $s'\sigma \in L(\mathbf{G})$, $s' \in L(\mathbf{S})$, and $s \equiv_I s'$. Then we have:

$$\begin{aligned} & [s, s' \in L(\mathbf{S}) \ \wedge \ s \equiv_I s' \implies \\ & \xi(r_0, s) = \xi(r_0, s')] \qquad \text{[Cor. 14]} \\ \implies & [s, s' \in L(\mathbf{S}) \ \wedge \ s\sigma \in L(\mathbf{S}) \wedge \\ & \xi(r_0, s) = \xi(r_0, s') \implies s'\sigma \in L(\mathbf{S})] \end{aligned}$$

Similarly for any two strings $s, s' \in L_m(\mathbf{G})$ such that $s \in L_m(\mathbf{S})$ and $s \equiv_I s'$ the following holds.

$$\begin{aligned} & [s, s' \in L_m(\mathbf{G}) \ \wedge \ s \in L_m(\mathbf{S}) \implies s, s' \in L(\mathbf{G}) \\ & \wedge \ s \in L(\mathbf{S}) \implies s' \in L(\mathbf{S})] \qquad \text{[Part 1 of proof]} \\ \wedge \quad & [s, s' \in L(\mathbf{G}) \ \wedge \ s \in L(\mathbf{S}) \wedge \ s \equiv_I s' \\ & \implies \xi(r_0, s) = \xi(r_0, s')] \qquad \text{[Cor. 14]} \\ \implies & [s \in L_m(\mathbf{S}) \wedge \xi(r_0, s) = \xi(r_0, s') \implies s' \in L_m(\mathbf{S})] \ \blacksquare \end{aligned}$$

3.4 Construction of IUF-yielding ALMs

Proposition 15 establishes one side of the relationship between the existence of IUFs and weak joint observability. The other side of this relationship, i.e. the fact that for a WJO behavior there exists a representation and an ALM which yield IUFs, seems more challenging to prove. First of all, there are different possible state representations for a given behavior. Second, it can be shown through examples that the existence of IUFs depends on the choice of the associated ALM (see Mannani and Gohari [2007a]), too. Moreover, a general constructive procedure for finding a suitable ALM for a specific representation of the behavior that yields IUFs is not yet known to exist. These facts have led the authors to investigate specific state representations for which IUFs can be computed. In what follows, we distinguish one such class of problems.

Lemma 16. For $i \in I$ let $A_i \subseteq \Sigma_i^*$ be a language recognized by $\mathbf{A}_i = (Q_i, \Sigma_i, \eta_i, q_{0,i}, Q_{m,i})$ such that

$$\forall q, q' \in Q_i, \ \forall \sigma \in \Sigma_i \setminus \Sigma_{o,i}. \ q' = \eta_i(q, \sigma) \implies q' = q, \quad (12)$$

i.e. all transitions unobservable to agent i are selfloops. Let $A = A_1 \| \cdots \| A_n \subseteq \Sigma^*$ be the synchronous product of A_i's recognized by $\mathbf{A} = \mathbf{A}_1 \| \cdots \| \mathbf{A}_n$. There exists an ALM yielding IUFs for (almost) any such A. ■

The proof is omitted here due to space considerations. However, we notice that to define an ALM $\ell_i(.)$ for each recognizer $\mathbf{A}_i$, its observable selfloops should be modified in the following cases. Example 2 illustrates the idea.

- C1 For an event, say $\alpha_i \in \Sigma_{o,i}$, which is selflooped in one state, say $q_1 \in Q_i$, and causes a state change in another state, say $q_2 \in Q_i$ (see Mannani and Gohari [2007a], Remark 1),
- C2 For an event, say $\alpha \in \Sigma_{o,i} \cap \Sigma_{o,j}, j \in I, j \neq i$, which is selflooped in one state, say $q_3 \in Q_i$, and causes a state change in an state, say $q_4 \in Q_j$, of $\mathbf{A}_j$,

a state $\hat{q}_1$ (respectively $\hat{q}_3$) is added to $\mathbf{A}_i$ which inherits all the outgoing non-selfloop transitions of q_1, all selfloops at q_1 labeled with events in $\Sigma_{loop,i} = \Sigma_{uo,i} \cup \{\sigma \in \Sigma_{o,i} \mid \forall q, q' \in Q_i. \ q' = \eta_i(q, \sigma) \Rightarrow q = q'\}$, and q_1's (respectively q_3's) marking, while all selfloop transitions at q_1 (respectively q_3's) which are not labeled by events in $\Sigma_{loop,i}$ are replaced with transitions with the same labels from q_1 (respectively q_3's) to $\hat{q}_1$ (respectively $\hat{q}_3$) and vice versa. Notice that in case $q_1 = q_3$ one duplicated state $\hat{q}_1$ takes care of the both cases.

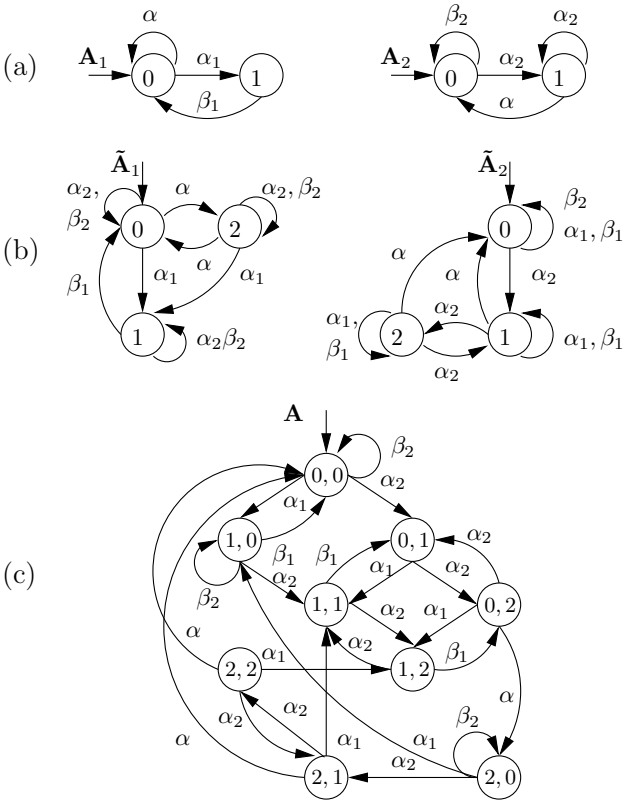

Fig. 1. (a) The recognizers $\mathbf{A}_1$ and $\mathbf{A}_2$. (b) The modified recognizers $\tilde{\mathbf{A}}_1$ and $\tilde{\mathbf{A}}_2$ whose states are labeled. (c) The meet of the modified recognizers whose states are labeled by joint labeling of the ALM's for $\tilde{\mathbf{A}}_1$ and $\tilde{\mathbf{A}}_2$.

Example 2. Figure 1-a shows two recognizers $\mathbf{A}_1$ and $\mathbf{A}_2$ where $\Sigma_1 = \Sigma_{o,1} = \{\alpha, \alpha_1, \beta_1\}$, $\Sigma_2 = \{\alpha, \alpha_2, \beta_2\}$, and $\Sigma_{o,2} = \{\alpha, \alpha_2\}$ and all states are assumed to be marked. Clearly (12) is satisfied. To arrive at the recognizers $\tilde{\mathbf{A}}_1$ and $\tilde{\mathbf{A}}_2$ in part (b) of the same figure, we notice that while the common event α makes a selfloop transition at state 0 of $\mathbf{A}_1$ it also makes a state change at state 0 of $\mathbf{A}_2$. Therefore, by C2, state 0 is duplicated yielding state 2 of $\tilde{\mathbf{A}}_1$. Also, event α_2 which moves $\mathbf{A}_2$ from state 0 to state 1 forms a selfloop at state 1 of the same recognizer. Thus, following the modification C1, state 1 is duplicated producing state 2 of $\tilde{\mathbf{A}}_2$. Labeling the states of the two modified recognizers, we would have the corresponding updating functions $\mathcal{A}_1$ and $\mathcal{A}_2$ as in Table 1.

Recognizer $\mathbf{A}$ is then computed as the meet of $\tilde{\mathbf{A}}_1$ and $\tilde{\mathbf{A}}_2$ as can be seen in part (c) where the tuple formed by the two ALM's for $\tilde{\mathbf{A}}_1$ and $\tilde{\mathbf{A}}_2$ is used as the ALM for $\mathbf{A}$ for which the updating functions in Table 1 are the IUFs. ◊

Next we show that Lemma 16 can be applied to the case where a language is trace-closed.

Lemma 17. For a trace-closed language $A \in \Sigma^*$ there exists a state representation for which there exists an ALM yielding IUFs.

Sketch of the proof: Since A is trace-closed, for each string in A all members of the equivalence class of that

string modulo $\equiv_I$ belong to A, i.e. $A = \bigcup_{i\in I} P_i^{-1}P_i(A)$ and $A = A_1\| \cdots \|A_n$. Then each $P_i(A)$ can be recognized by $\mathbf{A}_i = (Q_i, \Sigma_i, \eta_i, q_{0,i}, Q_{m,i})$ satisfying the condition of Lemma 16 by which the required ALM is computed. ■

In light of Lemma 16, any substructure derived from such a recognizer $\mathbf{A}$ in Lemma 16, may be assigned the same ALM. The following definition and corollary makes this point clear.

Definition 18. Let $\mathbf{B} = (Q, \Sigma, \xi, q_0, Q_m)$ be a recognizer. A recognizer $\hat{\mathbf{B}} = (\hat{Q}, \Sigma, \hat{\xi}, q_0, \hat{Q}_m)$ is called a *subrecognizer* of $\mathbf{B}$ if $\hat{Q} \subseteq Q$, $\hat{Q}_m = \hat{Q} \cap Q_m$, and

$\forall \sigma \in \Sigma^*,\ \forall q, q' \in \hat{Q}.\ q' = \hat{\xi}(q,\sigma) \implies q' = \xi(q,\sigma)$ □

Corollary 19. Let $\mathbf{A} = (Q, \Sigma, \xi, q_0, Q_m)$ be a recognizer for which there exists an ALM ℓ yielding IUFs and $\hat{\mathbf{A}} = (\hat{Q}, \Sigma, \hat{\xi}, q_0, \hat{Q}_m)$ be a subrecognizer of $\mathbf{A}$. Then ℓ, restricted to the states of $\hat{\mathbf{A}}$, is an ALM for $\hat{\mathbf{A}}$ yielding IUFs.

Proof: For any two states $q, q' \in Q$ and any string $s \in \Sigma^*$, with $q' = \xi(q,s) \in Q$, and any label $\underline{v} \in \ell(q)$ assigned by the ALM, and the updating function $\mathcal{A}$ it holds that

$$\begin{aligned} q' = \hat{\xi}(q,s) &\implies q' = \xi(q,s) && \text{[Defn. 18]} \\ &\iff \mathcal{A}(s,\underline{v}) \in \ell(q'). && \text{[(4)]} \end{aligned}$$

Thus the same updating function applies to $\hat{\mathbf{A}}$. In particular, this holds for the case of the IUFs, too. ■

Lemma 17 and Corollary 19 imply the following result.

Corollary 20. For any trace-closed supervisor's language there exists a state representation for which IUFs can be obtained. These IUFs can be employed for any subrecognizer of the state representation, too. ■

4. CONCLUSION

This paper explains the importance of independent updating functions (IUFs) in communication among decentralized supervisors of discrete-event systems. It also shows that weak joint observability is a necessary condition for the existence of IUFs and, to establish the other direction, provides constructive procedures to obtain IUFs for some classes of centralized supervisors. We would like to prove the correspondence in both directions in general and study the exact consequences of the existence of IUFs and the computational issues related to their computations in our future work.

Table 1. Updating functions for $\tilde{\mathbf{A}}_1$ and $\tilde{\mathbf{A}}_2$.

σ_1	v_1	$\mathcal{A}_1(\sigma_1, v_1)$	σ_2	v_2	$\mathcal{A}_2(\sigma_2, v_2)$
α_1	0	1	α_2	0	1
α_1	2	1	α_2	1	2
β_1	1	0	α_2	2	1
α	0	2	β_2	0	0
α	2	0	α	1	0
—	—	—	α	2	0

REFERENCES

George Barrett and Stéphane Lafortune. Decentralized supervisory control with communicating controllers. *IEEE Trans. Automat. Contr.*, 45:1620–1638, September 2000.

Amin Mannani and Peyman Gohari. Decentralized supervisory control of discrete-event systems over communication networks. Technical Report CRG-TR0207, Control and Robotics Group, Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada, February 2007a. URL `http://users.encs.concordia.ca/ crg/CRG.html`.

Amin Mannani and Peyman Gohari. A framework for modeling communication among decentralized supervisors for discrete-event systems. Technical Report CRG-TR0407, Control and Robotics Group, Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada, April 2007b. URL `http://users.encs.concordia.ca/ crg/CRG.html`.

Amin Mannani, Yue Yang, and Peyman Gohari. Distributed extended finite-state machines: Communication and control. In *Proc. of the IEEE 8th International Workshop on Discrete-Event Systems (WODES'06)*, pages 161–167, Ann Arbor, USA, July 2006.

A. Mazurkiewics. Introduction to trace theory. In V. Diekert and G. Rozenberg, editors, *The book of traces*. World Scientific, New Jersey, 1995.

S. Laurie Ricker and Karen Rudie. Incorporating communication and knowledge into decentralized discrete-event systems. In *Proc. IEEE Conference on Decision and Control (CDC'99)*, pages 1326–1332, Phoenix, USA, December 1999.

Karen Rudie, Stéphane Lafortune, and Feng Lin. Minimal communication in a distributed discrete-event system. *IEEE Trans. Automat. Contr.*, 48:957–975, June 2003.

Karen Rudie and W. Murray Wonham. Supervisory control of communicating processes. In L. Logrippo, R. L. Probert, and H. Ural, editors, *Protocol Specification, Testing, and Verification*. Elsevier Science Publishers, B. V., North Holland, 1990.

Karen Rudie and W. Murray Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Trans. Automat. Contr.*, 37:1692–1708, November 1992.

D. Teneketzis. On information structures and nonsequential stochastic control. *CWI Quarterly*, 10(2):179–199, 1997.

Stavros Tripakis. Undecidable problems of decentralized observation and control. In *Proc. IEEE Conference on Decision and Control (CDC'01)*, ,, December 2001.

Stavros Tripakis. Undecidable problems of decentralized observation and control on regular languages. *Information Processing Letters*, 90:2128, 2004.

Stavros Tripakis. Decentralized observation problems. In *Proc. IEEE Conference on Decision and Control-European Control Conference (CDC-ECC'05)*, pages 6–11, Seville, Spain, December 2005.

Jan H. van Schuppen. Decentralized control with communication between controllers. In Vincent D. Blondel and Alexandre Megretski, editors, *Unsolved Problems in Mathematical Systems and Control Theory*. Princeton University Press, Princeton, USA, 2004.

Walter Murray Wonham. Supervisory control of discrete-event systems, 2007. URL `http://www.control.utoronto.ca/DES/`.

Yue Yang and Peyman Gohari. Embedded supervisory control of discrete-event systems. In *IEEE Conference on Automation Science and Engineering (ASE)*, pages 410–415, August 2005.

T.-S. Yoo and Stéphane Lafortune. A general architecture for decentralized supervisory control of discrete-event systems. *Discrete-Event Dynamic Systems: Theory and Applications*, 12(3):335–377, July 2002.