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_mannani@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 a distributed DES, geographic separation of sites restricts the agents’ observation of the whole system’s 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 Laforetune, 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 Laforetune, 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], 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
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 DSDES

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, modeled by the natural projection \( \pi_i : \Sigma^* \rightarrow \Sigma_{o,i}^* \), and exercises control on events in \( \Sigma_{c,i} \). Thus, from the viewpoint of the \( i \)’th supervisor we have \( \Sigma_{o,i} \subseteq \Sigma_{o} \subseteq \Sigma \), and \( \Sigma_{o} = \Sigma \setminus \Sigma_{c} \). Define \( \Sigma_{o,i} \cup \Sigma_{c,i} \). Associated with each event \( \sigma \) denote by \( L_o(\sigma) \) and \( L_s(\sigma) \) the sets of all sensors (respectively, actuators) which can observe (respectively, control) \( \sigma \), i.e. \( L_o(\sigma) = \{ i \mid \sigma \in \Sigma_{o,i} \} \) and \( L_s(\sigma) = \{ i \mid \sigma \in \Sigma_{c,i} \} \). 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 \( \Sigma_{o} = \bigcup_{i \in I} \Sigma_{o,i} \), \( \Sigma_{o} = \bigcup_{i \in I} \Sigma_{o,i} \), and \( \Sigma_{o} \) is a finite set of events (alphabet); \( \Sigma_{c} = \Sigma \setminus \Sigma_{o} \). Let  be a tuple of \( n \) natural numbers which is sometimes written as \( (v_1, \ldots, v_n) \) to emphasize its \( i \)’th component, \( v_i \), where \( v_i \in I \) if \( i \) is the \( (n - 1) \)-tuple obtained by removing \( v_i \) from \( \sigma \). Define a map \( \pi_i : N^n \rightarrow N \) such that \( \pi_i(\sigma) = v_i \) (i.e. it picks the \( i \)’th component of \( \sigma \)), and extend \( \pi_i \) to a map \( \pi_i : N^* \rightarrow N \). Finally, the prefix closures of \( L \subseteq \Sigma^* \) and \( \{ \sigma \} \), \( s \in \Sigma^* \), are shown by \( \overline{L} \) and \( \overline{\sigma} \), 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 \( \Lambda_i = \Lambda_{c,i} \cup \Lambda_{o,i} \), we notice that for component \( i \), events in \( \Lambda_i \) are neither controllable nor observable. Therefore, we might redefine \( \Sigma_{o,i} \), by setting \( \Sigma_{o,i} = \Lambda_{c,i} \cup \Lambda_{o,i} \), and \( \Sigma_i = \Lambda_{c,i} \cup \Lambda_{o,i} \). Language \( L \) is defined as the synchronous product of \( L_i \), i.e. \( L = \bigcup_{i \in I} \overline{L_i} \). The following definition of the distributed supervisor is denoted by \( D = \{ D_i \}_{i \in I} \), where each quadruple \( D_i = (\Sigma, L, A_i, G_i) \) is defined as follows.

- \( \Sigma \) is a finite set of events (alphabet);
- \( L \) is a (regular) language defined over \( \Sigma \), i.e. \( L \subseteq \Sigma^* \);
- \( A_i : \Sigma^* \rightarrow N \) is an updating function;
- \( G_i : \Sigma \rightarrow \text{pwr}(\Sigma^*) \) is a guard function.

For convenience we extend the domain of \( A_i \) and \( G_i \) to the alphabet \( \Sigma \). Define \( \hat{A}_i : \Sigma \times N^n \rightarrow N \) and \( \hat{G}_i : \Sigma \rightarrow \text{pwr}(N^n) \) according to: for \( \sigma \in \Sigma \) and \( \nu \in N^n \),

\[
\hat{A}_i(\sigma, \nu) = \begin{cases} A_i(\sigma, \nu) : \sigma \in \Sigma_i, \\
\pi_i(\nu) : \sigma \notin \Sigma_i \end{cases} \quad \hat{G}_i(\sigma) = \begin{cases} G_i(\sigma) : \sigma \in \Sigma_i, \\
N^n : \sigma \notin \Sigma_i \end{cases}
\]

In the natural recursive way, \( \hat{A}_i \) is extended to \( \hat{A}_i : \Sigma^* \times N^n \rightarrow N \). With a slight abuse of notation, we shall use \( A_i \) and \( G_i \) to denote \( \hat{A}_i \) and \( \hat{G}_i \), respectively. Define a map \( A : \Sigma^* \times N^n \rightarrow N \) recursively as follows: for all \( \nu \in N^n \), \( s \in \Sigma^* \), and \( \sigma \in \Sigma \),

\[
A(\epsilon, \nu) = \nu; \quad A(\sigma, \nu) = (A_i(\sigma, A(s, \nu)))_{i \in I}
\]

(1) Definition 2. The closed and marked languages of \( D \) are denoted by \( L(D_i) \) and \( L_m(D_i) \), respectively, and are defined recursively as follows: \( \epsilon \in L(D_i) \) and for all \( s \in \Sigma^* \) and \( \sigma \in \Sigma \),

\[
s \in L(D_i) \iff s \in L(D) \land \sigma \in \overline{L} \land A(s, \nu) \in (G_i(\sigma))_{i \in I}
\]

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

\[
L(D) = \cap_{i \in I} L(D_i); \quad L_m(D) = \cap_{i \in I} L_m(D_i).
\]

3 Problem 3. Control problem for DSDES: Let the plant be modeled by an automaton \( G = (Q, \Sigma, \delta, q_0, Q_m) \) and \( E \subseteq L_m(G) \) be a nonempty specification language which is controllable with respect to \( G \), observable with respect to \( (G, P) \), and \( L_m(G) \)-closed (see [Wonham, 2007], Theorem 6.3.11). Assume that \( E \) is enforced by a proper, feasible and admissible centralized supervisor \( S = (R, \Sigma, \xi, r_0, R_m) \). Design guard and updating functions for each \( D_i \) based upon \( n \)-tuples of natural numbers; component \( i \) of a tuple is updated with \( A_i \).

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. \( G \) and \( P_i \), \( i \in I \) [Radje and Wonham, 1992].

Definition 4. Let \( S = (R, \Sigma, \xi, r_0, R_m) \) be a centralized supervisor. An Agent-wise Labeling Mapping (ALM) is a map \( \ell : R \rightarrow \text{pwr}(N^n) \) with the following properties:

1. \( \emptyset \in \ell(r_0); \)
2. \( \forall r, r' \in R. \; r \neq r' \rightarrow \ell(r) \cap \ell(r') = \emptyset \) (labels are disjoint);
3. \( \forall r, r' \in R. \; r \neq r' \rightarrow \forall \sigma \in \Sigma_o, \forall \nu \in N^n. \; \nu \in \ell(r) \land \ell(r') \rightarrow \exists \nu' \in N^n. \; \nu' \notin \ell(r) \land \nu' \notin \ell(r') \land \forall v \in I \ell(\sigma). \; \nu_i \neq \nu_i' \land \forall v \in I \ell(\sigma). \; \nu_i = \nu_i' \)

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

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

\[
\forall r, r' \in R, \; \sigma \in \Sigma, \; \nu \in N^n. \; \ell(\sigma, r) \land \ell(\sigma, r') \rightarrow \mu(\sigma, \nu) \in \ell(\sigma) \land \forall v \in I \ell(\sigma). \; \mu(\sigma, \nu) \neq \pi_i(\nu) \land \forall v \in I \ell(\sigma). \; \mu(\sigma, \nu) = \pi_i(\nu)
\]

The updating functions can be defined using the map \( \mu \) 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:
\( r' = \xi(r, \sigma) \land v \in \ell(r) \Rightarrow A_i(\sigma, v) = \pi_i(\mu(\sigma, v)) \). (2)

Upon occurrence of \( \sigma, \mu \) does not change the value of any \( v_j \) with \( j \in I \setminus L_0(\sigma) \). This fact and (2) imply the following.

\[ \forall \sigma \in \Sigma, \forall v \in \Sigma \setminus \Sigma_1, A_i(\sigma, v) = \pi_i(\mu(\sigma, v)). \] (3)

Formula (2) relates \( A \) to the transition structure of \( S \), i.e.

\[ \forall s \in L(S), \forall r \in R. A(s, \emptyset) \in \ell(r) \leftarrow r = \xi(r_0, s) \] (4)

A solution to Problem 3: Define \( D = \{ D_i \}_{i \in I} \), \( D_i = (L_m(G_i), \Sigma_i, G_i, A_i) \) for all \( i \in I \), where the maps \( A_i \) are defined as in (2), and the maps \( G_i \) are defined for as follows:

\[ \forall \sigma \in \Sigma, G_i(\sigma) = \left\{ \ell(v) | r \in R \land \xi(r, \sigma) \right\}; \text{ if } \sigma \in \Sigma_{c,i}, \text{ if } \sigma \in \Sigma_{u,i}. \] (5)

Then \( L(D) = E \) and \( L_m(D) = E \). 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 \( \bar{v} := A(s, \emptyset) \) and denote supervisor \( i \) by \( S_i \).

Communication for observation: Upon observing the occurrence of an event \( \sigma \), supervisor \( i \) must update \( v_i \) with the value \( A_i(\sigma, v) \). However, to correctly evaluate \( A_i(\sigma, v) \), \( S_i \) needs to know the value of \( v \) and thus may need to receive the value \( v_j \) for all \( j \neq i \), from \( S_j \). There are two methods to communicate the required values. According to the first approach, \( S_i \) requests all such supervisors to send it their \( v_j \)’s. According to the second approach, after \( S_i \) updates the value of \( v_i \), it immediately sends the new value to all the supervisors whose updating functions depend on \( v_i \). 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, \( S_i \) can unambiguously determine its new value for \( v_i \). This information may be used to update its estimate of the states of \( S \) 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 \( \alpha \) be in \( \Sigma_{c,i} \). Then \( \alpha \) is enabled at \( s \) if and only if \( v \in G(\alpha) \). To determine if this is the case, \( S_i \) may need to receive the value \( v_j \) for all \( j \neq i \), from \( S_j \). Since \( S_i \) is uncertain about the state of the centralized supervisor, and therefore about whether or not \( \alpha \) 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 \( S \) which is implemented by a network of 2 supervisors, i.e. \( |I| = 2 \), by using an ALM \( \ell_1 \) 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 \( \Sigma_{c,1} = \Sigma_{c,2} = \{\alpha_1\} \), and the updating functions and guard formula associated with supervisor 1 be as follows: \( g_1(\alpha_1) = h_{\alpha_1}(x_1^1, x_1^2), a_1(x_1^1, \alpha_1) = f_{\alpha_1,1}(x_1^1, x_1^2) \), and \( a_1(x_2^1, \alpha_1) = f_{\alpha_1,2}(x_2^1, x_2^2) \). Accordingly, upon observing \( \alpha_1 \), supervisor 1 (\( S_1 \)) should update \( x_1^1 \) and \( x_1^2 \) for which it should have received the values of \( x_1^1 \) and \( x_1^2 \) from supervisor 2 (\( S_2 \)). Knowing that \( g_1(\alpha_1) \) requires the updated value of \( x_1^2 \), \( S_2 \) then sends the updated value of \( x_2^1 \) to \( S_1 \). In summary, communication for observation (respectively, control) would require the transfer of 2 bits (respectively, 1 bit) from \( S_2 \) to \( S_1 \).

Assume now that there exists another ALM for \( S \) with the following EFSM implementation: \( X_1 = \{x_1^1, x_1^2, x_1^3\} \), \( X_2 = \{x_2^1, x_2^2\} \); \( g_1(\alpha_1) = h_{\alpha_1}(x_1^1, x_1^2, x_1^3, x_2^2), a_1(x_1^1, \alpha_1) = f_{\alpha_1,1}(x_1^1, x_1^2, x_1^3, x_2^2) \), \( a_1(x_2^1, \alpha_1) = f_{\alpha_1,2}(x_2^1, x_2^2) \) and \( a_1(x_2^2, \alpha_1) = f_{\alpha_1,3}(x_2^1, x_2^2) \). This new labeling employs a third variable, \( x_1^3 \), and its guard depends on two (as opposed to one) variables of \( S_2 \), but its updating functions do not depend on \( S_2 \)’s variables, i.e. it needs no communication for observation. As a result, \( S_1 \) 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 \( x_1^1 \) and \( x_1^2 \) are received in error, \( x_1^3 \) may be updated to incorrect values, thus \( S_1 \) cannot trust the values of its private variables thereafter. Retransmission of \( x_1^1 \) and \( x_1^2 \) would not help \( S_1 \) unless it keeps a record of the evolution of its private variables. However, a corrupted communication for control only affects \( g_1(\alpha_1) \) which can be reevaluated upon a retransmission of the correct values.

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 \( A_i \) 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 $v, v' \in \mathbb{N}_n$ the followings holds.
$$v_i = v'_i \implies A_i(\sigma, v) = A_i(\sigma, v').$$

In simple words an IUF $A_i$ reevaluates the agent $i$’s associated component of $g$, i.e. $v_i$, only if it need not know the value of $v_{i\neq j}$. 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 $P_u(s) = P_u(s')$.

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

**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 $(\Sigma_{o,1}, \ldots, \Sigma_{o,n})$ if
$$\forall s, s' \in L \setminus K \setminus \{\varepsilon\}, s \equiv_I s' \implies P_i(s) = P_i(s').$$

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 $(\Sigma_{o,1}, \ldots, \Sigma_{o,n})$ if
$$\forall s, s' \in L \setminus K \setminus \{\varepsilon\}, s \equiv_I s' \implies P_i(s) = P_i(s').$$

**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

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 $K \subseteq L$.

(WJO) with respect to $L$ and $(\Sigma_{o,1}, \ldots, \Sigma_{o,n})$ if the following conditions hold.
$$\forall s, s' \in L, \forall \sigma \in \Sigma \implies A_i(\sigma, v) = A_i(\sigma, v').$$

**Definition 10.** A language $A \subseteq \Sigma^*$ is called trace-closed if
$$\forall s, s' \in \Sigma^*, s \equiv_I s' \implies s' \in K.$$

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
$$\{s \in K \implies s \subseteq L\} \quad \text{[K \subseteq L]}$$
$$\Rightarrow \{s \in L \wedge s \equiv_I s' \wedge s \equiv_U s' \implies s' \subseteq L\} \quad \text{[Defn. 10]}$$
$$\Rightarrow \{s, s' \in L \wedge s \equiv_I s' \wedge s \in K \implies s' \in K\}, \quad \text{[\{7\]}$$
i.e. $K$ is trace-closed.

**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 $A_i$ be all independent. Then the following holds.
$$\forall i \in I, \forall s \in L(S), \forall v \in \mathbb{N}_n. A_i(s, v) = A_i(P_i(s), v)$$

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

**Corollary 14.** If the updating functions $A_i$ are all independent, then all observationally equivalent strings lead to the same state.

**Proof:** Let $s, s' \in L(S)$ be two observationally equivalent strings, i.e. $\forall i \in I. P_i(s) = P_i(s')$. Lemma 13 then implies that $\forall i \in I. A_i(s, v) = A_i(P_i(s), v) = A_i(s', v).$ Thus by (1) we have $A_i(s, v) = A_i(s', v)$ and by the uniqueness of labels in Definition 4, the proof is complete.

**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 $L_m(S)$ and $L_m(G)$, 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:
$$L_m(S) \cap L_m(G) = K \wedge L(S) \cap L(G) = \overline{K}. \quad (11)$$
Therefore, in the subsequent discussion, we may safely assume that $L_m(S) \subseteq L_m(G)$. □

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

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

$$[s, s' \in L(S) \land s \equiv s''] \quad \Rightarrow \quad \xi(r_0, s) = \xi(r_0, s') \quad \text{[Cor. 14]}

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

$$[s, s' \in L_m(G) \land s \in L_m(S) \land s \equiv s'] \quad \Rightarrow \quad \xi'(r_0, s) = \xi'(r_0, s') \quad \text{[Cor. 14]}$$

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 $A_i \subseteq \Sigma^*$ be a language recognized by $A_i = (Q_i, \Sigma_i, \eta_i, q_{i0}, Q_{mi})$ such that

$$(\forall q, q' \in Q_i, \forall \sigma \in \Sigma \setminus \Sigma_{oi}, q' = \eta_i(q, \sigma) \Rightarrow q' = q), \quad (12)$$
i.e. all transitions unobservable to agent $i$ are selfloops.

Let $A = A_1 \parallel \cdots \parallel A_n \subseteq \Sigma^*$ be the synchronous product of $A_i$'s recognized by $A = A_1 \parallel \cdots \parallel A_n$. 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 $f_i(.)$ for each recognizer $A_i$, its observable selfloops should be modified in the following cases. Example 2 illustrates the idea.

Example 2. Figure 1-a shows two recognizers $A_1$ and $A_2$ where $\Sigma_1 = \{\alpha, \beta, \gamma\}$, $\Sigma_2 = \{\alpha, \beta\}$, and $\Sigma_0 = \{\alpha, \beta\}$ and all states are assumed to be marked. Clearly (12) is satisfied. To arrive at the recognizers $\tilde{A}_1$ and $\tilde{A}_2$ in part (b) of the same figure, we notice that while the common event $\alpha$ makes a selfloop transition at state 0 of $A_1$ it also makes a state change at state 0 of $A_2$. Therefore, by C2, state 0 is duplicated yielding state 2 of $A_1$. Also, event $\alpha_2$ which moves $A_2$ 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 $A_2$. Labeling the states of the two modified recognizers, we would have the corresponding updating functions $A_1$ and $A_2$ as in Table 1.

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

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

![Fig. 1](a) The recognizers $A_1$ and $A_2$. (b) The modified recognizers $\tilde{A}_1$ and $\tilde{A}_2$. (c) The meet of the modified recognizers whose states are labeled by joint labeling of the ALM’s for $\tilde{A}_1$ and $\tilde{A}_2$.

3208
string modulo $\equiv_I$ belong to $A$, i.e. $A = \bigcup_{j=1}^{P-1} P_j^{-1} P_j(A)$ and $A = A_1 \sqcup \cdots \sqcup A_m$. Then each $P_j(A)$ can be recognized by $A_i = (Q_i, \Sigma_i, \eta_i, q_{i0}, Q_{im})$ 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 $A$ in Lemma 16, may be assigned the same ALM. The following definition and corollary makes this point clear.

**Definition 18.** Let $B = (Q, \Sigma, \xi, q_0, Q_m)$ be a recognizer. A recognizer $\hat{B} = (\hat{Q}, \Sigma, \hat{\xi}, \hat{q}_0, \hat{Q}_m)$ is called a subrecognizer of $B$ if $\hat{Q} \subseteq Q$, $\hat{Q}_m = \hat{Q} \cap \hat{Q}_m$, and

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

**Corollary 19.** Let $A = (Q, \Sigma, \xi, q_0, Q_m)$ be a recognizer for which there exists an ALM $\ell$ yielding IUFs and $\hat{A} = (\hat{Q}, \Sigma, \xi, \hat{q}_0, \hat{Q}_m)$ be a subrecognizer of $A$. Then $\ell$, restricted to the states of $\hat{A}$, is an ALM for $\hat{A}$ yielding IUFs.

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

$$q' = \xi(q, s) \quad \iff \quad q' = \xi(q, s) \quad \text{[Defn. 18]}$$

Thus the same updating function applies to $\hat{A}$. 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.

**REFERENCES**


![Table 1. Updating functions for $\hat{A}_1$ and $\hat{A}_2$.](http://users.encs.concordia.ca/crg/CRG.html)