Decentralized Supervisory Control With Intersection-Based Architecture

Xiang Yin and Stéphane Lafortune

***Abstract*—We investigate a new decentralized control architecture, called *intersection-based architecture*. We propose a decentralized control protocol under this architecture, called *state-estimator-intersection-based protocol* (SEI-protocol), where each local supervisor sends its state estimate to the fusion sites and these sites take the intersection of these estimates in order to make a control decision. The necessary and sufficient conditions for the achievability of the specification under this protocol is provided. This condition is termed as *state-estimator-intersection-based coobservability* (SEI-coobservability). A polynomial-time algorithm for the verification of SEI-coobservability is provided. We show that the languages that can be achieved under the SEI-protocol are *incomparable* with languages that can be achieved under existing architectures.**

***Index Terms*—Decentralized control, discrete-event systems (DES), intersection-based architecture, observability, supervisory control.**

I. Introduction

Decentralized control is an efficient way of tackling the control of large-scale networked systems, where the information structure is decentralized and the physical components of the system are distributed. In this technical note, we consider the decentralized supervisory control problem of Discrete Event Systems (DES). In this problem, the plant is controlled by a set of local supervisors acting on the plant based on their own observations. The supervisors cooperate with each other in order to achieve some *global* specification.

One of the key ingredients of the decentralized control problem is the *architecture* adopted; see, e.g., [1]–[11]. An architecture consists of the following two ingredients: (i) what information local supervisors can send to the fusion sites; and (ii) what rule the fusion sites use in order to fuse the information received from local supervisors into a global decision. Note that the fusion sites are distributed, since each of them corresponds to a local actuator (controllable event). The first architecture for the decentralized supervisory control problem was proposed in [1] and [2], where each supervisor makes a binary decision, "disable" or "enable"; the fusion rule is "*disable if one disables*." This architecture is referred to as the *conjunctive architecture*. In [3], the *disjunctive architecture* was proposed, where the fusion rule is "*enable if one enables*." In [5], a *general architecture* was proposed by combining the conjunctive architecture and the disjunctive architecture together. The above-mentioned architectures only allow supervisors to make *unconditional* decisions. In order to overcome this drawback, in [6], the *conditional architecture*, where supervisors are allowed to make conditional decisions, e.g., "disable if nobody enables" and "enable if nobody disables," was proposed. In [8], the *inference-based architecture* was proposed, where higher order inferences are allowed in order to achieve a richer class of languages. In [9], a framework that allows several architectures running in parallel was proposed. Note that the goal of using decentralized control is to handle the decentralized information structure rather than to provide a computationally efficient approach to find the centralized solution. This is different from the notion of *distributed control* in the literature, where one wants to compute the solution of the centralized problem more efficiently in a distributed way.

Manuscript received June 18, 2015; revised September 24, 2015 and January 21, 2016; accepted January 27, 2016. Date of publication January 28, 2016; date of current version October 25, 2016. This work was supported in part by NSF grants CCF-1138860 (Expeditions in Computing project ExCAPE: Expeditions in Computer Augmented Program Engineering) and CNS-1446298. Recommended by Associate Editor C. Hadjicostis.

The authors are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: xiangyin@umich.edu; stephane@umich.edu).

Digital Object Identifier 10.1109/TAC.2016.2523424

Similar to the decentralized control problem, many architectures and protocols have been proposed in the literature on decentralized diagnosis; see, e.g., [12]–[14]. In particular, in [12], a decentralized protocol (Protocol 2) was presented, where each local diagnoser sends its state estimate associated with fault labels to the coordinator and the coordinator takes the intersection of the state estimates in order to calculate the global decision. This idea was recently revisited in [15], where a polynomial test for diagnosability under this protocol is provided.

Motivated by Protocol 2 in [12], in this technical note, we propose a new decentralized control architecture, called *intersection-based architecture*. In particular, we investigate a particular protocol under this architecture, called *state-estimator-intersection-based protocol* (SEI-protocol). More specifically, under this protocol, each local supervisor sends its state estimate to the fusion sites and these fusion sites take the intersection of these state estimates in order to issue a global decision. Necessary and sufficient conditions for the achievability of a given specification language under the SEI-protocol are provided. One condition is the well-know controllability condition and the other condition is termed as *state-estimator-intersection-based coobservability* (SEI-coobservability). A polynomial time algorithm for the verification of SEI-coovservability is also provided. More interestingly, we show that the languages that can be achieved under the SEI-protocol are *incomparable* with the languages that can be achieved under any of the existing architectures discussed above. Consistent with the assumptions of previous works on decentralized supervisory control in the literature [1]–[3], [5], [6], [8], [16], the intersection-based architecture also has the following features. First, each fusion site need not have the model of the system and its decision only relies on the emptiness of the intersection, which can be easily implemented. In other words, most of the information is processed locally and only certain "useful" information is sent to the fusion sites. Second, the fusion rule for the intersection-based architecture is local to each actuator (controllable event) and it is memoryless, since it only takes the intersection of sets of states it receives *currently* and it does not depend on past information. These features are desirable for decentralized control architectures, since we want the fusion sites to be "as simple as possible." Finally, although a similar information structure was used in [12] and [15], the diagnosis problem and the control problem are inherently incomparable. Consequently, different architecture formulation and different verification method are required.

II. Preliminaries

We assume basic knowledge of DES and common notations (see, e.g., [17]). Let Σ be a finite set of events and Σ^* be the set of all finite strings over Σ, including the empty string ϵ. A language $L \subseteq \Sigma^*$ is a subset of Σ^* and $\overline{L}$ is the prefix-closure of language L defined by $\overline{L} = \{t \in \Sigma^* : \exists u \in \Sigma^* \text{ s.t. } tu \in L\}$. A DES is modeled as a deterministic finite-state automaton $G = (X, \Sigma, f, x_0, X_m)$, where X is the finite set of states, Σ is the finite set of events, $f : X \times \Sigma \to X$ is the partial transition function, $x_0 \in X$ is the initial state and X_m is the set of marked states. The transition function f is extended to $X \times \Sigma^*$ in the usual manner (see, e.g., [17]). The language generated by G is defined by $\mathcal{L}(G) = \{s \in \Sigma^* : f(x_0, s)!\}$, where ! means "is defined"; the marked language is $\mathcal{L}_m(G) = \{s \in \mathcal{L}(G) : f(x_0, s) \in X_m\}$.

In the framework of supervisory control [18], it is assumed that event set Σ is partitioned into two disjoint subsets: $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$, where Σ_c is the set of controllable events and Σ_{uc} is the set of uncontrollable events. Let $K \subseteq \mathcal{L}(G)$ be a specification language that represents the desired behavior to be achieved under control. We say that K is: (i) *controllable* (w.r.t. G and Σ_{uc}) if $\overline{K}\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \overline{K}$; and (ii) $\mathcal{L}_m(G)$-closed (w.r.t. G) if $\overline{K} \cap \mathcal{L}_m(G) = K$. Under the full observation setting, it is well-known that there exists a centralized non-blocking supervisor that achieves K, if and only if, K is both controllable and $\mathcal{L}_m(G)$-closed.

Let $H = (X_H, \Sigma, f_H, x_{0,H}, X_{m,H})$ be a trim automaton (see, e.g., [17], p.76) that marks the specification language K, i.e., $K = \mathcal{L}_m(H) \subseteq \mathcal{L}(G)$. We assume that H is a sub-automaton of G, denoted by $H \sqsubseteq G$ (see, e.g., [17]). Therefore, we have that $X_H \subseteq X$ and $(\forall x \in X_H)(\forall \sigma \in \Sigma)[f_H(x, \sigma)! \Rightarrow f_H(x, \sigma) = f(x, \sigma)]$. Note that this assumption is w.l.o.g., since we can always refine the state spaces of G and H such that this assumption holds; see, e.g., [19].

In the decentralized control problem, there are n local supervisors that work as a team in order to achieve the specification K. We denote by $\mathcal{I} = \{1, \ldots, n\}$ the index set of the local supervisors. Each supervisor $i \in \mathcal{I}$ has its own sets of controllable events and observable events. We denote by $\Sigma_{c,i}$ and $\Sigma_{o,i}$, the set of locally controllable events and the set of locally observable events, respectively, for supervisor $i \in \mathcal{I}$. We define $\Sigma_c = \cup_{i \in \mathcal{I}} \Sigma_{c,i}$, $\Sigma_o = \cup_{i \in \mathcal{I}} \Sigma_{o,i}$, $\Sigma_{uc} = \Sigma \setminus \Sigma_c$ and $\Sigma_{uo} = \Sigma \setminus \Sigma_o$. For each controllable event $\sigma \in \Sigma_c$, we denote by $I^c(\sigma) = \{i \in \mathcal{I} : \sigma \in \Sigma_{c,i}\}$ the set of supervisors that can disable σ. Also, for any $i \in \mathcal{I}$, $P_i : \Sigma^* \to \Sigma_{o,i}^*$ is the natural projection defined in the usual manner [17].

In order to control the system, supervisors make local decisions and send them to the *fusion sites* based on their own observations; a final decision is made at the fusion sites according to some fusion rule. We refer to the set of local decisions the supervisors send together with the fusion rule the fusion sites adopt as the *architecture* of the decentralized system. Here, we briefly review some existing decentralized architectures in the literature. The reader is referred to [1]–[3], [5], [6], [8] for more details.

In the unconditional architecture [1]–[3], [5], each supervisor makes an unconditional decision, i.e., "disable" or "enable." It was shown that the notion of *coobservability* together with controllability and $\mathcal{L}_m(G)$-closure provide the necessary and sufficient conditions for the existence of a non-blocking supervisor that achieves K.

In the conditional architecture [6], each supervisor makes conditional decisions, i.e., "enable," "disable," "enable if nobody disables" and "disable if nobody enables." It was shown that the notion of *conditional coobservability* together with controllability and $\mathcal{L}_m(G)$-closure provide the necessary and sufficient conditions for the existence of a non-blocking supervisor that achieves K.

In [8], an inference-based architecture was proposed. In this architecture, each local supervisor $S_i, i \in \mathcal{I}$ is defined as a map $S_i : P_i(\mathcal{L}(G)) \times \Sigma_{c,i} \to \{0, 1, \phi\} \times \mathbb{N}$, where "0" means "disable," "1" means enable and "ϕ" means "pass." The nonnegative integer component represents the *ambiguity level* of the local control decision. Each fusion site adopts the control decision that has the minimum ambiguity level. An inference-based supervisor is called *N-inferring* if the maximum ambiguity level is N. It was shown that *N-inference observability* together with controllability and $\mathcal{L}_m(G)$-closure provide the necessary and sufficient conditions for the existence of a non-blocking N-inferring supervisor that achieves K. Moreover, the authors in [8] show that coobervability is equivalent to 0-inference observability and conditional coobervability is equivalent to 1-inference observability. By allowing higher-order inference, one may be able to achieve richer behaviors.

III. Intersection-Based Architecture

In this section, we present a new decentralized control architecture called intersection-based architecture. First, we provide an overview of this architecture. In the intersection-based architecture, the default for the local decision is to disable events. However, in addition to this, each local supervisor should also send, for each disabled event, a set of states as the reason for its disablement. Each fusion site will take the intersection of these sets of states received from local supervisors. If the intersection is empty, then it means that there is no common reason for disabling the event and the final decision after fusion is to enable the event, since local supervisors cannot agree with each other. If the intersection is non-empty, then it means that the supervisors have at least one common reason for disabling the event and the final decision after fusion is to disable this event. Next, we present the formal definition of the intersection-based architecture.

In the intersection-based architecture, each local supervisor $i \in \mathcal{I}$ is defined as a function

$$S_i : P_i\left(\mathcal{L}(G)\right) \times \Sigma_{c,i} \to 2^X. \tag{1}$$

We denote by S_{int} the decentralized supervisor combining local supervisors $S_i, i \in \mathcal{I}$. Formally, in the intersection-based architecture, S_{int} is defined as the function $S_{\text{int}} : \mathcal{L}(G) \times \Sigma \to \{0, 1\}$ such that, for any $s \in \mathcal{L}(G)$, for any $\sigma \in \Sigma$,

- If $\sigma \in \Sigma_{uc}$, then $S_{\text{int}}(s, \sigma) = 1$;
- If $\sigma \in \Sigma_c$, then

$$S_{\text{int}}(s, \sigma) = \begin{cases} 1, & \text{if} \bigcap_{i \in I^c(\sigma)} S_i\left(P_i(s), \sigma\right) = \emptyset \\ 0, & \text{if} \bigcap_{i \in I^c(\sigma)} S_i\left(P_i(s), \sigma\right) \neq \emptyset \end{cases} \tag{2}$$

Clearly, S_i can always enable an event in $\Sigma_{c,i}$ by sending $\emptyset$. We denote by $\mathcal{L}(S_{\text{int}}/G)$ the closed-loop behavior under the decentralized supervisor S_{int}, which is defined inductively as follows.

- $\epsilon \in \mathcal{L}(S_{\text{int}}/G)$;
- $[s\sigma \in \mathcal{L}(S_{\text{int}}/G)] \Leftrightarrow [s \in \mathcal{L}(S_{\text{int}}/G) \wedge s\sigma \in \mathcal{L}(G) \wedge S_{\text{int}}(s, \sigma) = 1]$.

We also define $\mathcal{L}_m(S_{\text{int}}/G) = \mathcal{L}(S_{\text{int}}/G) \cap \mathcal{L}_m(G)$.

IV. State-Estimator-Intersection-Based Protocol

In this section, we first propose a decentralized control protocol under the intersection-based architecture. Then we provide the necessary and sufficient conditions under which the proposed protocol achieves the specification language exactly.

In the intersection-based architecture, there is no restriction on the local decision functions and the supervisor can send any information (as a set of states) it wants (no matter if it achieves the specification or not). Here, we provide a specific local decision rule for each supervisor, which determines the information it sends to the fusion site for each of its controllable events. We refer to these fixed local decision rules as a *protocol* under the architecture.

Before we formally present our decentralized control protocol, we introduce some necessary notations. Recall that the automaton $H = (X_H, \Sigma, f_H, x_{0,H}, X_{m,H})$ that marks the specification language K is assumed to be a sub-automaton of $G = (X, \Sigma, f, x_0, X_m)$. For each controllable event $\sigma \in \Sigma_c$, we define the set of disabled states by

$$D_H(\sigma) := \{x \in X_H : f(x, \sigma)! \wedge f_H(x, \sigma)\neg!\} \tag{3}$$

where $\neg!$ means "is not defined." In other words, $D_H(\sigma)$ is the set of states at which we need to disable σ in order to avoid executing an illegal string in $\mathcal{L}(G) \setminus \overline{K}$. Also, for each local supervisor $i \in \mathcal{I}$, we denote by $\mathcal{E}_i(s)$ the *state estimate* of supervisor i w.r.t. H by observing $P_i(s), s \in \overline{K}$, i.e.,

$$\mathcal{E}_i(P_i(s)) = \{x \in X_H : \exists t \in \overline{K} \text{ s.t. } P_i(t) = P_i(s) \wedge f_H(x_{0,H}, t) = x\}. \tag{4}$$

Now, we are ready to present the decentralized control protocol, called the *State-Estimator-Intersection-Based Protocol* (SEI-protocol). Under this protocol, for any string $s \in \mathcal{L}(G)$ and any controllable event $\sigma \in \Sigma_c$, each local supervisor $i \in I^c(\sigma)$ computes its state estimate $\mathcal{E}_i(P_i(s))$ locally and compares it with the set $D_H(\sigma)$. Then it sends the set of states that it thinks could be the reason for disabling σ, i.e., $\mathcal{E}_i(P_i(s)) \cap D_H(\sigma)$, to the corresponding fusion site. Therefore, the SEI-protocol is realized by the decentralized supervisor S^*_{int} consisting of $S^*_i, i \in \mathcal{I}$ defined as follows. For any $s \in \mathcal{L}(G)$ and any $\sigma \in \Sigma$

$$S^*_i(P_i(s), \sigma) = \begin{cases} \mathcal{E}_i(P_i(s)) \cap D_H(\sigma), & \text{if } \sigma \in \Sigma_{c,i} \\ \emptyset, & \text{if } \sigma \in \Sigma \setminus \Sigma_{c,i}. \end{cases} \tag{5}$$

Definition IV.1: A language $K = \mathcal{L}_m(H) \subseteq \mathcal{L}(G)$ is said to be state-estimator-intersection-based coobservable (SEI-coobservable) w.r.t. $H, G, \Sigma_{c,i}$ and $\Sigma_{o,i}, i \in \mathcal{I}$ if

$$(\forall s \in \overline{K})(\forall \sigma \in \Sigma_c : s\sigma \in \overline{K}) \left[\left(\bigcap_{i \in I^c(\sigma)} \mathcal{E}_i[P_i(s)]\right) \cap D_H(\sigma) = \emptyset\right]. \tag{6}$$

Remark IV.1: SEI-coobservability not only depends on the languages under consideration, but also depends on the automata generating these languages, since it involves sets $D_H(\sigma)$ and $\mathcal{E}_i(P_i(s))$, which both rely on the state space. We will see later that even for the same language, different state-space realizations will result in different SEI-coobservability properties. Hereafter, we assume that the state-space realization of K, i.e., H, is given and fixed, and we want to investigate the SEI-coobservability w.r.t. the fixed H.

The following result states that the SEI-protocol achieves the language K, if and only, K is controllable and SEI-coobservable.

Theorem IV.1: Given automata H and G, where $\mathcal{L}(H) = K \neq \emptyset$, $\mathcal{L}(S^*_{\text{int}}/G) = \overline{K}$ iff K is controllable (w.r.t. $\mathcal{L}(G)$ and Σ_{uc}) and SEI-coobservable (w.r.t. $H, G, \Sigma_{c,i}$ and $\Sigma_{o,i}, i \in \mathcal{I}$).

Proof: ($\Leftarrow$) First, we show that $\mathcal{L}(S^*_{\text{int}}/G) \subseteq \overline{K}$ by induction on the length of the strings. Initially, we know that $\epsilon \in \mathcal{L}(S^*_{\text{int}}/G)$ and $\epsilon \in \overline{K}$. Now, let us assume that for any $s \in \mathcal{L}(S^*_{\text{int}}/G)$ such that $|s| = k$, we have $s \in \overline{K}$. For the induction step, suppose that $s\sigma \in \mathcal{L}(S^*_{\text{int}}/G)$ is a string with $|s\sigma| = k + 1$. Since $s\sigma \in \mathcal{L}(S^*_{\text{int}}/G)$, we know that $s\sigma \in \mathcal{L}(G)$ and $S^*_{\text{int}}(s, \sigma) = 1$, i.e., either

(i) $\sigma \in \Sigma_{uc}$; or
(ii) $\sigma \in \Sigma_c$ and $(\cap_{i \in I^c(\sigma)} \mathcal{E}_i(P_i(s))) \cap D_H(\sigma) = \emptyset$.

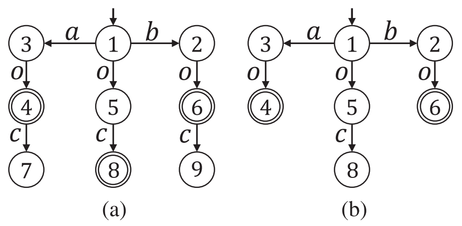


Fig. 1. An example of SEI-coobservable language, where $\Sigma_{o,1} = \{a, o, c\}$, $\Sigma_{o,2} = \{b, o, c\}$, and $\Sigma_{c,1} = \Sigma_{c,2} = \{c\}$. (a) G. (b) H.

If (i) holds, since K is controllable, we know that $s\sigma \in \overline{K}$. If (ii) holds, since $\{f(x_0, s)\} \subseteq (\cap_{i \in I^c(\sigma)} \mathcal{E}_i(P_i(s)))$, we know that $\{f(x_0, s)\} \cap D_H(\sigma) = \emptyset$, which implies that $f_H(x, \sigma)!$, where $x = f(x_0, s)$. Therefore, we have that $s\sigma \in \overline{K}$.

Second, we show that $\overline{K} \subseteq \mathcal{L}(S^*_{\text{int}}/G)$ by induction on the length of the strings. Initially, we know that $\epsilon \in \overline{K}$ and $\epsilon \in \mathcal{L}(S^*_{\text{int}}/G)$. Now, let us assume that for any $s \in \overline{K}$ such that $|s| = k$, we have $s \in \mathcal{L}(S^*_{\text{int}}/G)$. For the induction step, suppose that $s\sigma \in \overline{K}$ is a string with $|s\sigma| = k + 1$. If $\sigma \in \Sigma_{uc}$, we know that $S^*_{\text{int}}(s, \sigma) = 1$, which means $s\sigma \in \mathcal{L}(S^*_{\text{int}}/G)$. If $\sigma \in \Sigma_c$, since K is SEI-coobservable, we know that $[\cap_{i \in I^c(\sigma)} \mathcal{E}_i(P_i(s))] \cap D_H(\sigma) = \emptyset$. Thus

$$\bigcap_{i \in I^c(\sigma)} S^*_i(P_i(s), \sigma) = \bigcap_{i \in I^c(\sigma)} [\mathcal{E}_i(P_i(s)) \cap D_H(\sigma)] = \emptyset \tag{7}$$

which implies that $S^*_{\text{int}}(s, \sigma) = 1$, i.e, $s\sigma \in \mathcal{L}(S^*_{\text{int}}/G)$.

($\Rightarrow$) By contradiction. Assume that $\mathcal{L}(S^*_{\text{int}}/G) = \overline{K}$ but K is not controllable or SEI-coobservable. If K is not controllable, then we know that $\exists s \in \overline{K}$, $\exists \sigma \in \Sigma_{uc}$ such that $s\sigma \in \mathcal{L}(G) \setminus \overline{K}$. Since $\sigma \in \Sigma_{uc}$, we know that $S^*_{\text{int}}(s, \sigma) = 1$, i.e., $s\sigma \in \mathcal{L}(S^*_{\text{int}}/G)$. This contradicts the assumption that $\mathcal{L}(S^*_{\text{int}}/G) = \overline{K}$. If K is not SEI-coobservable, we know that

$$(\exists s \in \overline{K})(\exists \sigma \in \Sigma_c : s\sigma \in \overline{K}) \left[\left(\bigcap_{i \in I^c(\sigma)} \mathcal{E}_i[P_i(s)]\right) \cap D_H(\sigma) \neq \emptyset\right]$$

which implies that

$$\bigcap_{i \in I^c(\sigma)} S^*_i(P_i(s), \sigma) = \bigcap_{i \in I^c(\sigma)} (\mathcal{E}_i[P_i(s)] \cap D_H(\sigma)) \neq \emptyset.$$

Therefore, we know that $S^*_{\text{int}}(s, \sigma) = 0$, i.e., $s\sigma \notin \mathcal{L}(S^*_{\text{int}}/G)$. This again contradicts the assumption that $\mathcal{L}(S^*_{\text{int}}/G) = \overline{K}$. □

Remark IV.2: In the "$\Leftarrow$" direction of the above proof, we see that SEI-coobservability is used only in the proof of the second set inclusion. In other words, $\mathcal{L}(S^*_{\text{int}}/G) \subseteq \overline{K}$ always holds if K is controllable. Therefore, we know that even if SEI-coobservability does not hold, using the SEI-protocol still results in a safe solution, in the sense that $\mathcal{L}(S^*_{\text{int}}/G) \subseteq \overline{K}$.

The following result is a corollary of Theorem IV.1 when nonblockingness is of concern.

Collorary IV.1: $\mathcal{L}(S^*_{\text{int}}/G) = \overline{K}$ and $\overline{\mathcal{L}_m(S^*_{\text{int}}/G)} = \mathcal{L}(S^*_{\text{int}}/G)$ iff K is controllable (w.r.t. $\mathcal{L}(G)$ and Σ_{uc}), SEI-coobservable (w.r.t. $H, G, \Sigma_{c,i}$ and $\Sigma_{o,i}, i \in \mathcal{I}$) and $\mathcal{L}_m(G)$-closed.

Example IV.1: Consider the system automaton G and the specification automaton H in Fig. 1. We have that $D_H(c) = \{4, 6\}$. For string o such that $oc \in \overline{K}$, we have

$$\mathcal{E}_1(P_1(o)) \cap \mathcal{E}_2(P_2(o)) \cap D_H(c) = \{5, 6\} \cap \{4, 5\} \cap \{4, 6\} = \emptyset.$$

Therefore, K is SEI-coobservable and can be achieved by S^*_{int}. For example, if o occurs, then we know that $S^*_{\text{int}}(o, c) = 1$, since

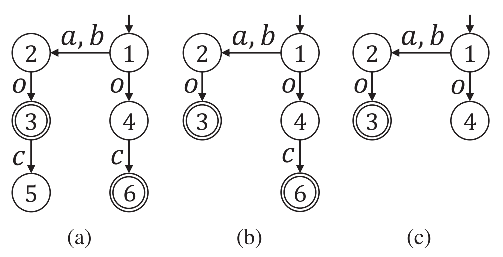


Fig. 2. $\Sigma_{o,1} = \{a, o, c\}$, $\Sigma_{o,2} = \{b, o, c\}$, and $\Sigma_{c,1} = \Sigma_{c,2} = \{c\}$. (a) H. (b) $\mathcal{L}(S^*_{\text{int}}/G)$.

$S^*_1(P_1(o)) \cap S^*_2(P_2(o)) = \emptyset$. Similarly, if ao occurs, we know that $S^*_1(P_1(ao)) \cap S^*_2(P_2(ao)) = \{4\} \cap \{4, 5\} \neq \emptyset$, i.e., $S^*_{\text{int}}(ao, c) = 0$.

Note that the SEI-protocol is a special protocol under the intersection-based architecture. One may ask whether or not SEI-coobservability is also necessary for the existence of supervisor under the intersection-based architecture. Unfortunately, this is not true in general, as is shown in the following example.

Example IV.2: Consider the system automaton G and the specification automaton H in Fig. 2. We know that $D_H(c) = \{3\}$. For string o such that $oc \in \overline{K}$, we have $\mathcal{E}_1(P_1(o)) \cap \mathcal{E}_2(P_2(o)) \cap D_H(c) = \{3, 4\} \cap \{3, 4\} \cap \{3\} = \{3\} \neq \emptyset$. Therefore, K is not SEI-coobservable. By applying supervisor S^*_{int}, we obtain the closed-loop behavior shown in Fig. 2(c). We see that S_{int} is still safe, although it cannot achieve K exactly.

However, there does exist an intersection-based supervisor that achieves K. Let us consider the supervisor defined as follows:

$$S_1(P_1(s), c) = \begin{cases} \{1\}, & \text{if } P_1(s) = o \\ \{1, 2\}, & \text{otherwise} \end{cases}$$

$$S_2(P_2(s), c) = \begin{cases} \{2\}, & \text{if } P_2(s) = o \\ \{1, 2\}, & \text{otherwise.} \end{cases}$$

One can easily verify that $S_{\text{int}}(ao, c) = S_{\text{int}}(bo, c) = 0$ and $S_{\text{int}}(o, c) = 1$, i.e., K can be achieved under S_{int}.

Remark IV.3: The system language and specification language in Figs. 1 and 2 are exactly the same. The only difference is that some states in Fig. 1 are merged as a single state in Fig. 2. The above example also illustrates the interesting phenomenon that SEI-coobservability not only depends on the system and specification languages under consideration, but it also depends on the automata generating these languages. Intuitively, this can be interpreted as follows: in Fig. 2, state 4 carries the information that "string ao or bo has occurred"; however, in Fig. 1, state 4 carries more the precise information that "only string ao has occurred." In other words, by refining the state space of the system, states can carry more information.

V. Verification of SEI-Coobservability

In this section, we provide a polynomial time algorithm for the verification of SEI-coobservability by constructing a new automaton that we call the I-Verifier. The I-Verifier is similar to the $\mathcal{M}$-machine studied by [5], [6], and [20] for the verification of unconditional and conditional coobservability. For the sake of simplicity, we state the results for two supervisors, i.e., $\mathcal{I} = \{1, 2\}$; the extension to the case of n supervisors is straightforward.

Given the system automaton $G = (X, \Sigma, f, x_0, X_m)$, the specification automaton $H = (X_H, \Sigma, f_H, x_{0,H}, X_{m,H})$, where $H \sqsubseteq G$, and local supervisors with controllable events and observable events $\Sigma_{c,i}, \Sigma_{o,i}$, $i = 1, 2$, the I-Verifier V is defined as the deterministic automaton

$$V = (X_V, \Sigma_V, f_V, x_{0,V}, X_{V,m}) \tag{8}$$

where:

- $X_V = X \times X \times X$ is the set of states;
- $\Sigma_V = (\Sigma \cup \{\epsilon\}) \times (\Sigma \cup \{\epsilon\}) \times (\Sigma \cup \{\epsilon\})$ is the event set;
- $x_{0,V} = (x_0, x_0, x_0)$ is the initial state;
- $X_{V,m}$ is the set of marked states;
- $f_V : X_V \times \Sigma_V \to X_V$ is the partial (deterministic) transition function defined below.

In the definition of f_V, we refer to the following violation condition (VIO) stated in terms of the controllability properties of the event $\sigma \in \Sigma$ under consideration at state triple $(x_1, x_2, x_3) \in X_V$:

$$\begin{aligned}
(\text{VIO}) &\Leftrightarrow (\text{VIO}-1) \vee (\text{VIO}-2) \vee (\text{VIO}-12) \\
(\text{VIO}-1) &\Leftrightarrow \sigma \in \Sigma_{c,1} \setminus \Sigma_{c,2} \wedge f_H(x_1, \sigma)! \wedge f_H(x_2, \sigma)\neg! \\
&\quad \wedge f(x_2, \sigma)! \\
(\text{VIO}-2) &\Leftrightarrow \sigma \in \Sigma_{c,2} \setminus \Sigma_{c,1} \wedge f_H(x_1, \sigma)! \wedge f_H(x_3, \sigma)\neg! \\
&\quad \wedge f(x_3, \sigma)! \\
(\text{VIO}-12) &\Leftrightarrow \sigma \in \Sigma_{c,1} \cap \Sigma_{c,2} \wedge f_H(x_1, \sigma)! \wedge x_2 = x_3 \\
&\quad \wedge f_H(x_2, \sigma)\neg! \wedge f(x_2, \sigma)!.
\end{aligned}$$

The transition function f_V is specified as follows

(a) For $\sigma \in \Sigma_{o,1} \cap \Sigma_{o,2}$

$$f_V((x_1, x_2, x_3), (\sigma, \sigma, \sigma)) = (f_H(x_1, \sigma), f_H(x_2, \sigma), f_H(x_3, \sigma)).$$

(b) For $\sigma \in \Sigma_{o,1} \setminus \Sigma_{o,2}$

$$\begin{aligned}
f_V((x_1, x_2, x_3), (\sigma, \sigma, \epsilon)) &= (f_H(x_1, \sigma), f_H(x_2, \sigma), x_3) \\
f_V((x_1, x_2, x_3), (\epsilon, \epsilon, \sigma)) &= (x_1, x_2, f_H(x_3, \sigma)).
\end{aligned}$$

(c) For $\sigma \in \Sigma_{o,2} \setminus \Sigma_{o,1}$

$$\begin{aligned}
f_V((x_1, x_2, x_3), (\sigma, \epsilon, \sigma)) &= (f_H(x_1, \sigma), x_2, f_H(x_3, \sigma)) \\
f_V((x_1, x_2, x_3), (\epsilon, \sigma, \epsilon)) &= (x_1, f_H(x_2, \sigma), x_3).
\end{aligned}$$

(d) For $\sigma \in \Sigma_{uo}$

$$\begin{aligned}
f_V((x_1, x_2, x_3), (\sigma, \epsilon, \epsilon)) &= (f_H(x_1, \sigma), x_2, x_3) \\
f_V((x_1, x_2, x_3), (\epsilon, \sigma, \epsilon)) &= (x_1, f_H(x_2, \sigma), x_3) \\
f_V((x_1, x_2, x_3), (\epsilon, \epsilon, \sigma)) &= (x_1, x_2, f_H(x_3, \sigma)).
\end{aligned}$$

Finally, we define by the set of marked states by

$$X_{V,m} := \{(x_1, x_2, x_3) \in X_V : \text{(VIO) holds for some } \sigma \in \Sigma \text{ at } (x_1, x_2, x_3)\}. \tag{9}$$

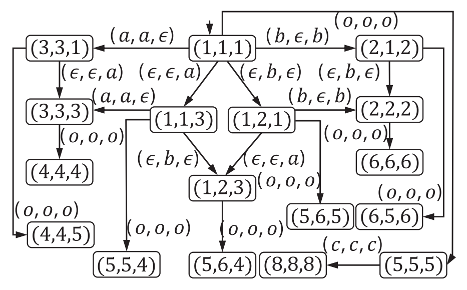
This completes the definition of the I-verifier.

Note that a string in V is a sequence of event triple. Hereafter, with a slight abuse of notation, we also write $(s_1, s_2, s_3) \in \Sigma^*_V$ if $(\sigma^1_1, \sigma^1_2, \sigma^1_3) \ldots (\sigma^n_1, \sigma^n_2, \sigma^n_3) \in \Sigma^*_V$, where $s_i = \sigma^1_i \ldots \sigma^n_i$, $i = 1, 2, 3$. By construction, the I-verifier V has the following property. For any string $s = (s_1, s_2, s_3) \in \Sigma^*_V$, where $s_1, s_2, s_3 \in \mathcal{L}(H)$, we have that $s \in \mathcal{L}(V)$ iff $P_1(s_1) = P_1(s_2)$ and $P_2(s_1) = P_2(s_3)$. The following theorem reveals that K is not SEI-coobservable, if and only if, a marked is reachable in the I-verifier.

Theorem V.1: A language $K = \mathcal{L}_m(H) \subseteq \mathcal{L}(G)$ is SEI-coobservable (w.r.t. $H, G, \Sigma_{c,i}$ and $\Sigma_{o,i}, i \in \mathcal{I}$) if and only if $\mathcal{L}_m(V) = \emptyset$.

Proof: ($\Leftarrow$) By contrapositive. Suppose that K is not SEI-coobservable. We know that

$$(\exists s \in \overline{K})(\exists \sigma \in \Sigma_c : s\sigma \in \overline{K}) \left[\left(\bigcap_{i \in I^c(\sigma)} \mathcal{E}_i(P_i(s)) \right) \cap D_H(\sigma) \neq \emptyset \right]$$


(a)

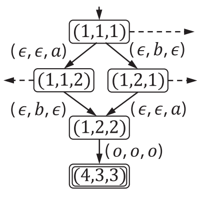

(b)

Fig. 3. Examples of the I-Verifier. (a) The complete I-Verifier for the system in Fig. 1. (b) Part of the I-Verifier for the system in Fig. 2.

which implies

$$(\exists s \in \overline{K})(\exists \sigma \in \Sigma_c : s\sigma \in \overline{K})\,(\exists x \in D_H(\sigma))$$
$$(\forall i \in I^c(\sigma))\,(\exists s_i \in \overline{K})\,[P_i(s) = P_i(s_i) \wedge f_H(x_0, s_i) = x]\,. \tag{10}$$

For $i = 1, 2$, let $t_i = s_i$ if $i \in I^c(\sigma)$ and $t_i = s$ if $i \notin I^c(\sigma)$. Let $x_1 := f_H(x_0, s)$, $x_2 := f_H(x_0, t_1)$ and $x_3 := f_H(x_0, t_2)$. Since $P_1(s) = P_1(t_1)$ and $P_2(s) = P_2(t_2)$, by the construction of the I-Verifier, we know that state (x_1, x_2, x_3) is reachable in V. Since $s\sigma \in \overline{K}$, we know that $f_H(x_1, \sigma)!$. Also, for $i \in I^c(\sigma)$, by $x_{i+1} = x \in D_H(\sigma)$, we know that $f_H(x_{i+1}, \sigma)\neg! \wedge f(x_{i+1}, \sigma)!$. Therefore, if $\sigma \in \Sigma_{c,1} \setminus \Sigma_{c,2}$, then (VIO-1) holds. Similarly, (VIO-2) holds if $\sigma \in \Sigma_{c,2} \setminus \Sigma_{c,1}$. If $\sigma \in \Sigma_{c,1} \cap \Sigma_{c,2}$, then (VIO-12) holds. Overall, we know that condition (VIO) holds at (x_1, x_2, x_3) for σ, which means that (x_1, x_2, x_3) is marked, i.e., $\mathcal{L}_m(V) \neq \emptyset$.

($\Rightarrow$) By contrapositive. Suppose that $\mathcal{L}_m(V) \neq \emptyset$. We know that there exists a string $s = (s_0, s_1, s_2) \in \mathcal{L}(V)$ such that condition (VIO) holds at $(f_H(x_0, s_0), f_H(x_0, s_1), f_H(x_0, s_2))$ for some $\sigma \in \Sigma$. If (VIO-1) holds, then we know that $\sigma \in \Sigma_{c,1} \setminus \Sigma_{c,2}$, $s_0\sigma \in \overline{K}$ and $s_1\sigma \in \mathcal{L}(G) \setminus \overline{K}$. If (VIO-2) holds, then we know that $\sigma \in \Sigma_{c,2} \setminus \Sigma_{c,1}$, $s_0\sigma \in \overline{K}$ and $s_2\sigma \in \mathcal{L}(G) \setminus \overline{K}$. Finally, if (VIO-12) holds, then we know that $\sigma \in \Sigma_{c,1} \cap \Sigma_{c,2}$, $s_0\sigma \in \overline{K}$, $s_1\sigma \in \mathcal{L}(G) \setminus \overline{K}$, $s_2\sigma \in \mathcal{L}(G) \setminus \overline{K}$ and $f_H(x_0, s_1) = f_H(x_0, s_2)$. Moreover, by the construction of V, we know that $P_1(s_0) = P_1(s_1)$ and $P_2(s_0) = P_2(s_2)$. Overall, we know that $(\exists s_0 \in \overline{K})(\exists \sigma \in \Sigma_c : s_0\sigma \in \overline{K})(\exists x \in D_H(\sigma))$ $(\forall i \in I^c(\sigma))(\exists s_i \in \overline{K})[P_i(s) = P_i(s_i) \wedge f_H(x_0, s_i) = x]$, which implies that $\overline{K}$ is not SEI-coobservable. □

Example V.1: Let us first revisit Example IV.1 and consider the system and the specification shown in Fig. 1. The corresponding I-Verifier is shown in Fig. 3(a). Since there is no marked state in V, we know that it is SEI-coobservable. Next, we return to Example IV.2 and consider the system and the specification shown in Fig. 2. Part of the corresponding I-Verifier is shown in Fig. 3(b). Since a marked state is reachable in V, we know that SEI-coobservability does not hold.

Remark V.1: In the case of n agents, each state and each event in the I-Verifier are $(n+1)$-tuples rather than triples. Then, for each state $(x_1, x_2, \ldots, x_{n+1})$ and each event σ, we have the following two types of transitions defined in V:

1) The single transition

$$f_V\left((x_1, x_2, \ldots, x_{n+1}), (\sigma, e_1, \ldots, e_n)\right)$$
$$= \left(f_H(x_1, \sigma), f_H(x_2, e_1), \ldots, f_H(x_{n+1}, e_n)\right),$$
$$\text{where } \forall i \in \mathcal{I},\ e_i = \begin{cases} \sigma, & \text{if } \sigma \in \Sigma_{o,i} \\ \epsilon, & \text{if } \sigma \notin \Sigma_{o,i}. \end{cases}$$

2) The following transition for each $i \in \mathcal{I}$ s.t. $\sigma \notin \Sigma_{o,i}$:

$$f_V\left((x_1, x_2, \ldots, x_{n+1}), \left(\epsilon, \epsilon, \ldots, \epsilon, \underset{(i+1)^{\text{th}}}{\sigma}, \epsilon, \ldots, \epsilon\right)\right)$$
$$= (x_1, x_2, \ldots, x_i, f_H(x_{i+1}, \sigma), x_{i+2}, \ldots, x_{n+1})\,.$$

Moreover, the violation condition (VIO) is defined by

$$(\text{VIO}) \Leftrightarrow f_H(x_1, \sigma)!$$
$$\wedge\,(\forall i, j \in I^c(\sigma))\,[x_{i+1} = x_{j+1} =: x \wedge f_H(x, \sigma)\neg! \wedge f(x, \sigma)!]\,.$$

In this case, the I-Verifier V has at most $|X|^{n+1}$ states and $(n+1)|\Sigma|\,|X|^{n+1}$ transitions. Therefore, checking whether or not $\mathcal{L}_m(V) = \emptyset$ can be done in $O((n+1)|\Sigma||X|^{n+1})$, which is polynomial when n is fixed, but exponential when n is unbounded. Moreover, once SEI-coobservability has been verified off-line, the SEI-protocol can be implemented in a *online* manner. More specifically, according to (5), each supervisor only needs to remember its current state estimate and update it upon the occurrence of a new locally observable event, which also can be done in polynomial time for each step; see, e.g., [21].

VI. Properties of SEI-Coobservability

A. Comparison With Existing Architectures

In this section, we compare the languages that can be achieved by the SEI-protocol and those can be achieved under the existing architectures. First, we recall some definitions from [5], [6], [8].

Definition VI.1: A language $K \subseteq \mathcal{L}(G)$ is said to be N-inference observable w.r.t. $\mathcal{L}(G), \Sigma_{c,i}$ and $\Sigma_{o,i}, i \in \mathcal{I}$ if

$$(\forall \sigma \in \Sigma_c)\,[D_{N+1}(\sigma) = \emptyset \vee E_{N+1}(\sigma) = \emptyset] \tag{11}$$

where sets D_{N+1} and E_{N+1} are defined inductively as follows:

$$\begin{cases} D_0(\sigma) := \{s \in \overline{K} : s\sigma \in \mathcal{L}(G) \setminus \overline{K}\} \\ E_0(\sigma) := \{s \in \overline{K} : s\sigma \in \overline{K}\} \end{cases} \tag{12}$$

$$\begin{cases} D_{k+1}(\sigma) := D_k(\sigma) \cap \left(\bigcap_{i \in I^c(\sigma)} P_i^{-1}\left(P_i\left[E_k(\sigma)\right]\right)\right) \\ E_{k+1}(\sigma) := E_k(\sigma) \cap \left(\bigcap_{i \in I^c(\sigma)} P_i^{-1}\left(P_i\left[D_k(\sigma)\right]\right)\right). \end{cases} \tag{13}$$

Moreover, we say that $K \subseteq \mathcal{L}(G)$ is

- *coobservable* if it is 0-inference observable;
- *conditional coobservable* if it is 1-inference observable;
- *inference observable* if there exists a nonnegative integer $N \in \mathbb{N}$ such that K is N-inference observable.

By [6] and [8], we know that conditional coobservability is weaker than coobservability and inference observability is weaker than conditional coobservability. Hereafter, we will show that SEI-coobservability is *incomparable* with all of these notions. First, we show that none of these notions implies SEI-coobservability.

Example VI.1: Clearly, the system in Example IV.2, which is not SEI-coobservable, is coobservable, since supervisor 1 can disable c after ao occurs and supervisor 2 can disable c after bo occurs. More specifically, it is easy to compute that $D_1(c) = \emptyset$. Since coobservability is stronger than conditional coobservability or inference observability, we know that $\mathcal{L}(H)$ is also conditional coobservable and inference observable.

The next example shows that the SEI-protocol can achieve a language that cannot be achieved by any of the existing architectures.

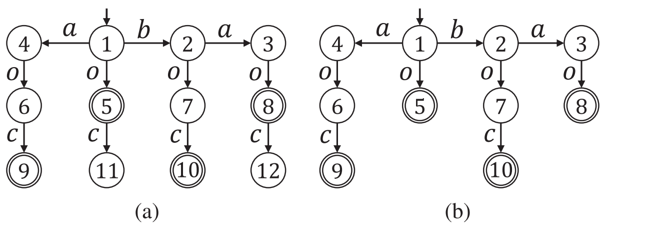


Fig. 4. $\Sigma_{o,1} = \{a, o, c\}$, $\Sigma_{o,2} = \{b, o, c\}$, and $\Sigma_{c,1} = \Sigma_{c,2} = \{c\}$. (a) G. (b) H.

Example VI.2: Consider the system G and the specification H in Fig. 4. Let us compute the sets $D_k(c)$ and $E_k(c)$ defined in (12) and (13). Initially, we have

$$D_0(c) = \{o, bao\} \text{ and } E_0(c) = \{ao, bo\}$$

Also, we have

$$\begin{cases} D_1(c) = D_0(c) \cap \left(\bigcap_{i\in I^c(\sigma)} P_i^{-1}\left(P_i\left[E_0(c)\right]\right)\right) = \{o, bao\} \\ E_1(c) = E_0(\sigma) \cap \left(\bigcap_{i\in I^c(\sigma)} P_i^{-1}\left(P_i\left[D_0(c)\right]\right)\right) = \{ao, bo\}. \end{cases}$$

Since $D_1(c) = D_0(c) \neq \emptyset$ and $E_1(c) = E_0(c) \neq \emptyset$, by (13), we get that $D_N(c) = D_0(c) \neq \emptyset$ and $E_N(c) = E_0(c) \neq \emptyset$ for any $N \in \mathbb{N}$. Therefore, K is not N-inference observable for any $N \in \mathbb{N}$, i.e., K is not inference observable, which also implies that K is not coobservable or conditional coobservable.

However, K is SEI-coobservable. For string ao such that $aoc \in \overline{K}$, we have $\mathcal{E}_1(ao) \cap \mathcal{E}_2(ao) \cap D_H(c) = \{6, 8\} \cap \{5, 6\} \cap \{5, 8\} = \emptyset$. Similarly, for string bo such that $boc \in \overline{K}$, we have $\mathcal{E}_1(bo) \cap \mathcal{E}_2(bo) \cap D_H(c) = \{5, 7\} \cap \{7, 8\} \cap \{5, 8\} = \emptyset$. Therefore, K is SEI-coobservable and it can be achieved by supervisor S^*_{int}.

By combining Examples VI.1 and VI.2 together, we have the following facts about the proposed protocol.

Fact 1: For arbitrarily given and fixed automata G and H, if $\mathcal{L}(H)$ is coobservable (w.r.t. $\mathcal{L}(G)$, $\Sigma_{c,i}$ and $\Sigma_{o,i}, i \in \mathcal{I}$), then it does not imply that $\mathcal{L}(H)$ is SEI-coobservable (w.r.t. H, G, $\Sigma_{c,i}$ and $\Sigma_{o,i}, i \in \mathcal{I}$). Consequently, neither $\mathcal{L}(H)$ conditional coobservable nor $\mathcal{L}(H)$ inference observable implies that $\mathcal{L}(H)$ is SEI-coobservable.

Fact 2: For arbitrarily given and fixed automata G and H, if $\mathcal{L}(H)$ is SEI-coobservable (w.r.t. H, G, $\Sigma_{c,i}$ and $\Sigma_{o,i}, i \in \mathcal{I}$), then it does not imply that $\mathcal{L}(H)$ is inference observable (w.r.t. $\mathcal{L}(G), \Sigma_{c,i}$ and $\Sigma_{o,i}, i \in \mathcal{I}$). Consequently, $\mathcal{L}(H)$ SEI-coobservable neither implies that $\mathcal{L}(H)$ is coobservable nor implies that $\mathcal{L}(H)$ is conditional coobservable.

B. Case of Finite Languages

In [16], a notion called joint observability was studied. We recall the definition[1]:

Definition VI.2: A language $K \subseteq \mathcal{L}(G)$ is said to be jointly observable w.r.t. $\mathcal{L}(G)$, $\Sigma_{c,i}$ and $\Sigma_{o,i}, i \in \mathcal{I}$ if

$$(\forall\, \sigma \in \Sigma_c)(\forall\, s\sigma \in \overline{K})(\forall\, s'\sigma \in \mathcal{L}(G) \setminus \overline{K})\,(\exists i \in I^c(\sigma)) \quad [P_i(s) \neq P_i(s')]\,.$$

[1]In [16], joint observability was defined for the decentralized observation problem. Here we provide a slightly modified version of the definition in [16] for the purpose of decentralized control.

It was also shown in [16] that K is jointly observable iff for any $\sigma \in \Sigma_c$, there exists a function $f : \Sigma^*_{o,i_1} \times \cdots \times \Sigma^*_{o,i_k} \to \{0, 1\}$, where $\{i_1, \ldots, i_k\} = I^c(\sigma)$, such that $\forall\, s \in \mathcal{L}(G)$, if $s\sigma \in \overline{K}$, then $f(P_{i_1}(s), \ldots, P_{i_k}(s), \sigma) = 1$ and if $s\sigma \in \mathcal{L}(G) \setminus \overline{K}$, then $f(P_{i_1}(s), \ldots, P_{i_k}(s), \sigma) = 0$. Intuitively, this result says that joint observability together with controllability provide the necessary and sufficient conditions for the existence of an arbitrary memoryless architecture under which the specification language can be achieved. Unfortunately, checking joint observability is undecidable in general. Hereafter, we provide a sufficient condition under which joint observability and SEI-coobservability are equivalent. First, given an automaton G, we say that G is a tree if

$$(\forall\, x \in X)\,(\forall\, s, t \in \mathcal{L}(G))\,[f(x_0, s) = f(x_0, t) \Rightarrow s = t]$$

Clearly, if a language is finite, then it can always be generated by a finite tree.

The following theorem reveals that SEI-coobservability coincides with joint observability when the system is a tree.

Theorem VI.1: Suppose that H and G are trees. Then $\mathcal{L}(H)$ is jointly observable if and only if $\mathcal{L}(H)$ is SEI-coobservable.

Proof: The "if" part is trivial, since (2) and (5) define a memoryless decision function. We prove the "only if" part by contrapositive. Suppose that $\mathcal{L}(H)$ is not SEI-coobservable. As discussed earlier, we know that (10) holds. Consider the state x and strings $s_i, i \in I^c(\sigma)$ in (10). Under the assumption that H is a tree, we know that $s_i = s_j,\ \forall\, i, j \in I^c(\sigma)$. We denote this unique string by s'. Therefore, we know that $(\exists \sigma \in \Sigma_c)(\exists s\sigma \in \overline{K})(\exists s'\sigma \in \mathcal{L}(G) \setminus \overline{K})(\forall\, i \in I^c(\sigma))$ $[P_i(s) = P_i(s')]$ which implies that $\mathcal{L}(H)$ is not jointly observable. □

Remark VI.1: The intuition of the above theorem is as follows. Since G is a tree, the string that leads to a state is unique. Therefore, by sending the set of states, we send all useful local (language) information to the fusion sites, i.e., $\mathcal{E}_i(P_i(s))$ and $P_i^{-1}(P_i(s))$ carry the same information in this case. Clearly, checking joint observability is decidable if the language is finite, since we can enumerate all the strings. However, even for the case of a finite language, it is not clear how joint observability can be efficiently verified and what architecture can be used in order to achieve the language if joint observability holds. By using Theorem VI.1, we see that SEI-coobservability and the SEI-protocol answer these two questions, respectively, when the specification language is finite.

VII. Conclusion

We presented a new decentralized control architecture, called intersection-based architecture. Under this architecture, we proposed a new decentralized control protocol called the SEI-protocol. This protocol is interesting and useful in practice, since:

(i) Whether or not it achieves the specification language can be verified in polynomial time.
(ii) If the specification is achievable, then the protocol can be implemented online with polynomial complexity at each step.
(iii) The languages that can be achieved by the SEI-protocol and the languages that can be achieved under existing decentralized architectures are incomparable.

Therefore, the SEI-protocol provides a new tool for the decentralized control problem that complements existing architectures. If the behavior can be achieved by using unconditional or conditional architectures, then, of course, there is no need to use the proposed architecture. However, if none of the existing architectures works, then the proposed architecture provides a new opportunity for achieving the specification.

References

[1] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya, “Supervisory control of discrete-event processes with partial observations,” *IEEE Trans. Autom. Control*, vol. 33, no. 3, pp. 249–260, 1988.

[2] K. Rudie and W. Wonham, “Think globally, act locally: Decentralized supervisory control,” *IEEE Trans. Autom. Control*, vol. 37, no. 11, pp. 1692–1708, 1992.

[3] J. Prosser, M. Kam, and H. Kwatny, “Decision fusion and supervisor synthesis in decentralized discrete-event systems,” in *Proc. Amer. Control Conf.*, 1997, vol. 4, pp. 2251–2255.

[4] S. Ricker and K. Rudie, “Know means no: Incorporating knowledge into discrete-event control systems,” *IEEE Trans. Autom. Control*, vol. 45, no. 9, pp. 1656–1668, 2000.

[5] T.-S. Yoo and S. Lafortune, “A general architecture for decentralized supervisory control of discrete-event systems,” *Discrete Event Dyn. Sys.: Theory Appl.*, vol. 12, no. 3, pp. 335–377, 2002.

[6] T.-S. Yoo and S. Lafortune, “Decentralized supervisory control with conditional decisions: Supervisor existence,” *IEEE Trans. Autom. Control*, vol. 49, no. 11, pp. 1886–1904, 2004.

[7] S. Ricker and K. Rudie, “Knowledge is a terrible thing to waste: Using inference in discrete-event control problems,” *IEEE Trans. Autom. Control*, vol. 52, no. 3, pp. 428–441, 2007.

[8] R. Kumar and S. Takai, “Inference-based ambiguity management in decentralized decision-making: Decentralized control of discrete event systems,” *IEEE Trans. Autom. Control*, vol. 52, no. 10, pp. 1783–1794, 2007.

[9] H. Chakib and A. Khoumsi, “Multi-decision supervisory control: Parallel decentralized architectures cooperating for controlling discrete event systems,” *IEEE Trans. Autom. Control*, vol. 56, no. 11, pp. 2608–2622, 2011.

[10] S. Ricker and H. Marchand, “A parity-based architecture for decentralized discrete-event control,” in *Proc. ACC*, 2013, pp. 5678–5684.

[11] C. Seatzu, M. Silva, and J. H. Van Schuppen, *Control of Discrete-Event Systems: Automata and Petri Net Perspectives*. London, U.K.: Springer-Verlag, 2013.

[12] R. Debouk, S. Lafortune, and D. Teneketzis, “Coordinated decentralized protocols for failure diagnosis of discrete event systems,” *Discrete Event Dyn. Sys.: Theory Appl.*, vol. 10, no. 1–2, pp. 33–86, 2000.

[13] Y. Wang, T.-S. Yoo, and S. Lafortune, “Diagnosis of discrete event systems using decentralized architectures,” *Discrete Event Dyn. Sys.: Theory Appl.*, vol. 17, no. 2, pp. 233–263, 2007.

[14] R. Kumar and S. Takai, “Inference-based ambiguity management in decentralized decision-making: Decentralized diagnosis of discrete-event systems,” *IEEE Trans. Autom. Sci. Eng.*, vol. 6, no. 3, pp. 479–491, 2009.

[15] M. Panteli and C. Hadjicostis, “Intersection based decentralized diagnosis: Implementation and verification,” in *Proc. 52nd IEEE Conf. Decision Control*, 2013, pp. 6311–6316.

[16] S. Tripakis, “Undecidable problems of decentralized observation and control on regular languages,” *Inform. Processing Lett.*, vol. 90, no. 1, pp. 21–28, 2004.

[17] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. 2nd ed. New York, NY, USA: Springer-Verlag, 2008.

[18] P. Ramadge and W. Wonham, “Supervisory control of a class of discrete event processes,” *SIAM J. Cont. Opt.*, vol. 25, no. 1, pp. 206–230, 1987.

[19] H. Cho and S. I. Marcus, “On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation,” *Math. Control Sig. Syst.*, vol. 2, no. 1, pp. 47–69, 1989.

[20] K. Rudie and J. Willems, “The computational complexity of decentralized discrete-event control problems,” *IEEE Trans. Autom. Control*, vol. 40, no. 7, pp. 1313–1319, 1995.

[21] P. Kozak and W. Wonham, “Fully decentralized solutions of supervisory control problems,” *IEEE Trans. Autom. Control*, vol. 40, no. 12, pp. 2094–2097, 1995.