RECENT ADVANCES ON THE CONTROL OF PARTIALLY-OBSERVED DISCRETE-EVENT SYSTEMS

Stéphane Lafortune, Kurt Rohloff, and Tae-Sic Yoo
Department of Electrical Engineering and Computer Science
University of Michigan Ann Arbor, Michigan 48109-2122 USA *
{stephane,krohloff,tyoo}@eecs.umich.edu; www.eecs.umich.edu/umdes

Abstract This paper reviews and discusses some recent results on the control of partially-observed discrete-event systems. Several results related to the synthesis of safe solutions in decentralized control architectures are described and illustrated with examples.

Keywords: Supervisory control, partial-observation, decentralized control.

1. Introduction

Our objective in this paper is to review some recent results on the control of partially-observed discrete-event systems (abbreviated DES hereafter). Both centralized and decentralized control architectures are considered. The control framework adopted is that of the theory of supervisory control of DES, initiated by Ramadge & Wonham in the 1980's [Ramadge and Wonham, 1987]. We shall assume some familiarity with supervisory control theory in the rest of this paper. The reader is referred to [Ramadge and Wonham, 1989, Thistle, 1996] for surveys of this theory and Chapter 3 of [Cassandras and Lafortune, 1999] for a textbook treatment of the basic results. There are many modeling formalisms for DES, including automata, Petri nets, and process algebras. The automaton modeling formalism is used in this paper. However, the key concepts are presented in terms of languages, thus independent of any particular modeling formalism.

*This research is supported in part by NSF grant CCR-0082784 and by the DDR&E MURI on Low Energy Electronics Design for Mobile Platforms and managed by ARO under grant ARO DAAH04-96-1-0377.

Consider a DES modeled by an automaton (or state machine) denoted by G; let the set of event labels in G be denoted by E. Equivalently, the system is modeled by the languages generated and marked by G, denoted by $\mathcal{L}(G)$ and $\mathcal{L}_m(G)$, respectively. The prefix-closed language $\mathcal{L}(G)$ models all the traces of events that the system can execute while the marked language $\mathcal{L}_m(G)$ models those traces in $\mathcal{L}(G)$ that represent, by modeling choice, the completion of some operation or task. The notion of a marked language, or equivalently the notion of marked states in G, allows the consideration of blocking in the analysis of DES.

The automaton G models the uncontrolled behavior of the system. This behavior must be restricted by control in order to ensure that only legal traces of events are generated and that blocking does not occur (or its effect is mitigated if blocking cannot be completely eliminated). Control is exerted by means of a supervisor, denoted by S, that observes the events generated by G and controls the events that G is allowed to execute. The controlled system is denoted by S/G. In order to account for actuation and sensing limitations, the set of events E is partitioned in two ways. Regarding actuation limitations, E is partitioned into $E = E_c \cup E_{uc}$, where E_{uc} is the set of uncontrollable events and E_c is the set of controllable events. The controllable events are those events that can be enabled or disabled by the supervisor. Regarding sensing limitations, E is partitioned into $E = E_o \cup E_{uo}$, where E_{uo} is the set of unobservable events and E_o is the set of observable events. The observable events are those events that can be observed or "seen" by the supervisor, meaning that they are recorded by the sensors. When $E_{uo} \neq \emptyset$, the supervisor is often denoted by S_P, where the subscript P refers to "partial observation".

The control architecture described above is depicted in Fig. 1. A control theory of discrete-event systems has been developed to answer fundamental questions regarding necessary and sufficient conditions for the existence of supervisors that achieve a given legal behavior that captures all the requirements (or specifications) imposed on G in the context of the control architecture of Fig. 1. This theory is known as "supervisory control theory" of DES. The existence results of supervisory control theory have led to the development of algorithmic procedures for the synthesis of supervisors that are guaranteed to be *safe* and *nonblocking*. Safety means that the controlled behavior never exceeds the legal behavior. Nonblocking means that there is no deadlock or livelock. A deadlock happens when the system enters a state that is not marked and no transition is defined/enabled out of that state. A livelock happens when the system enters a cycle of unmarked states and there are no transitions defined/enabled out of the cycle.

Three "key" properties of discrete-event system theory are: *controllability*, *observability*, and *co-observability*; see, e.g., [Cassandras and Lafortune, 1999]. These properties are usually stated as language properties. The algorithmic procedures that have been developed for testing these properties and synthesizing supervisors are restricted at present to finite-state systems and employ automaton models for the DES (namely, G is a finite-state automaton) and for the legal behavior.

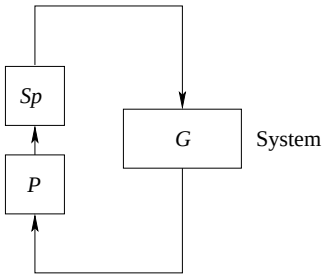


Figure 1. The feedback loop of supervisory control in the case of partial observation.
The projection $P : E^* \to E_o^*$ hides the unobservable events executed by G from supervisor S_P.

In this paper, we will focus mostly on the properties of observability and co-observability; they will be defined in the following sections. For the sake of completeness, we recall the definition of controllable language. The notation $\overline{K}$ denotes the prefix-closure of language K, i.e., all traces in K plus all of their prefixes, including the empty trace denoted by ε.

Definition 1 (**Controllability**) *Let K and $M = \overline{M}$ be languages over event set E. Let E_{uc} be a designated subset of E. K is said to be* controllable *with respect to M and E_{uc} if*

$$\overline{K}E_{uc} \cap M \subseteq \overline{K} \ .$$

By definition, controllability is a property of the prefix-closure of a language. The controllability condition is a central concept in supervisory control and it can be paraphrased by: *"If you cannot prevent it, then it should be legal."*

2. Centralized Control Under Partial Observation

Let us study the control architecture of Fig. 1 in more detail. The uncontrolled system is modeled by automaton $G = (X, E, f, x_0, X_m)$,

where X is the state space, E the set of event labels, f the state transition function, x_0 the initial state, and X_m the subset of X that consists of the marked states. Let $E_{uc} \subseteq E$ be the set of uncontrollable events and $E_o \subseteq E$ be the set of observable events. Let P be the projection operation from E^* to E_o^* that "erases" the unobservable events in any trace of events to produce its observable projection. Consider the language $K \subseteq \mathcal{L}_m(G)$, where $K \neq \emptyset$. K is to be interpreted as the legal language that models what is considered to be safe behavior for the controlled system.

The "Controllability and Observability Theorem" of supervisory control theory gives the necessary and sufficient conditions for the existence of a partial-observation supervisor S_P that achieves exactly the legal language K in closed-loop, without any blocking.

Theorem 1 *Consider a language $K \subseteq \mathcal{L}_m(G)$ where $K \neq \emptyset$. There exists a nonblocking supervisor S_P for G such that $\mathcal{L}_m(S_P/G) = K$ and $\mathcal{L}(S_P/G) = \overline{K}$ iff the three following conditions hold:*
1. *K is controllable w.r.t. $\mathcal{L}(G)$, E_{uc}.*
2. *K is observable w.r.t. $\mathcal{L}(G)$, P, and E_c.*
3. *$K = \overline{K} \cap \mathcal{L}_m(G)$.*

We defined earlier the property of controllability. The third condition, called $\mathcal{L}_m(G)$-closure, means that prefixes of traces in K that are marked by G should also be in K. It is easy to ensure that this condition holds when K is defined.

We now further discuss the observability condition of the Controllability and Observability Theorem. The formal definition of observability is as follows:

Definition 2 (**Observability**) *Let K and $M = \overline{M}$ be languages over event set E. Let E_c be a designated subset of E. Let E_o be another designated subset of E with P as the corresponding projection from E^* to E_o^*.*
K is said to be observable *with respect to M, P, and E_c if for all $s \in \overline{K}$ and for all $\sigma \in E_c$,*

$$(s\sigma \notin \overline{K}) \text{ and } (s\sigma \in M) \;\Rightarrow\; P^{-1}[P(s)]\sigma \cap \overline{K} = \emptyset.$$

Note the slight abuse of notation in the definition: $P^{-1}[P(s)]\sigma$ stands for $P^{-1}[P(s)]\{\sigma\}$. Intuitively, observability means: *"If you cannot differentiate between two traces, then these traces should require the same control action."* Another way to phrase this, from the point of view of event disablement, is: *"If you must disable an event after observing a trace, then by doing so you should not disable any trace that appears in the desired behavior."*

Observability is easily verified by examination of the observer of G with respect to the set of observable events E_o. An observer is a deterministic automaton built from G where unobservable transitions in G have been replaced by ε-transitions and the resulting nondeterministic automaton is transformed into a language-equivalent deterministic one; see, e.g., Chapter 2 of [Cassandras and Lafortune, 1999]. However, the state space of the observer of G may be exponential in the state space of G in the worst case. It turns out that there exists a polynomial test for observability [Tsitsiklis, 1989].

The observability properties of the events in E are related to the sensors attached to the system G. If the legal language K is observable with a given set of observable events E_o, one may be interested in finding a subset of E_o of *minimum cardinality* such that K remains observable with this new set of observable events. (Here, E_c remains unchanged.) Note that there may be several incomparable observable event sets, each minimal with respect to set inclusion, such that K is observable. This selection of a smaller set of observable events corresponds to removing sensors that are redundant from the viewpoint of being able to achieve K in closed-loop. It has been recently shown that this type of sensor selection problem is NP-complete [Yoo and Lafortune, 2001]. However, if more structure is included into the problem, for instance in the context of a probabilistic formulation, then there exist polynomial-time algorithms that minimize the cardinality of the set of sensors (i.e., observable events); see [Debouk et al., 1999].

3. Decentralized Control

We turn our attention to the decentralized control architecture depicted in Fig. 2, where a set of supervisors jointly control G by each observing subsets of E_o (denoted by $E_{o,i}$) and controlling subsets of E_c (denoted by $E_{c,i}$). C&P co-observability is a key component of the necessary and sufficient condition for the existence of a decentralized control system that exactly achieves the legal behavior in the context of Fig. 2 [Cieslak et al., 1988, Rudie and Wonham, 1992, Ricker and Rudie, 2000]. We follow the definition that is presented in [Barrett, 1999, Cassandras and Lafortune, 1999]. In that definition, P_i is the projection operation from E^* to $E_{o,i}^*$.

Definition 3 (**C&P co-observability**) *A language $K \subseteq M = \overline{M}$ is said to be C&P co-observable w.r.t. M, $E_{o,1}$, $E_{c,1}$,..., $E_{o,n}$, $E_{c,n}$, if $\forall s \in \overline{K}$ and $\forall \sigma \in E_c = \cup_{i=1}^{n} E_{c,i}$ s.t. $s\sigma \in M \setminus \overline{K}$,*

$$(\exists i \in \{1, \ldots, n\})[[P_i^{-1} P_i(s)\sigma \cap \overline{K} = \emptyset] \wedge [\sigma \in E_{c,i}]].$$

C&P co-observability is for the conjunctive architecture and hence the "C" in C&P co-observability refers to the *conjunctive* fusion rule for controllable events. Similarly, the "P" refers to the *permissive* decision strategy taken by local supervisors if there is insufficient local knowledge to determine the correct control action. The permissive local decision rule implies that the default control action for a supervisor under insufficient information is to "enable" an event. C&P co-observability can be tested in polynomial time [Rudie and Willems, 1995].

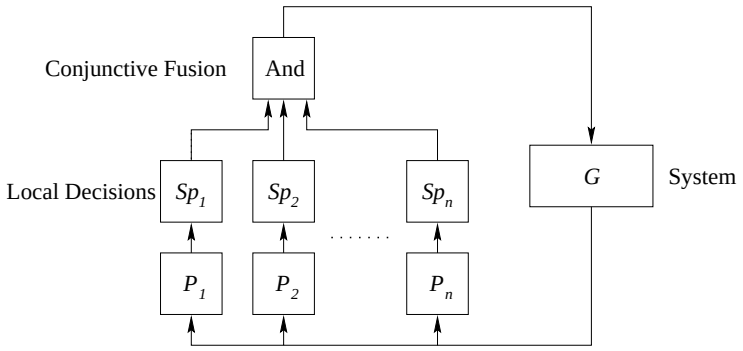


Figure 2. Decentralized control architecture.
This architecture is said to be "conjunctive" as the control actions (enabled events) of the local supervisors are fused by intersection.

Let us consider the following air traffic control example. We have two radar stations providing take-off commands and alert reports to airplanes. If one of the radar stations detects any object, it sends an alert and prevents the take-off of airplanes. The desired behavior is to provide safe take-offs and timely alerts.

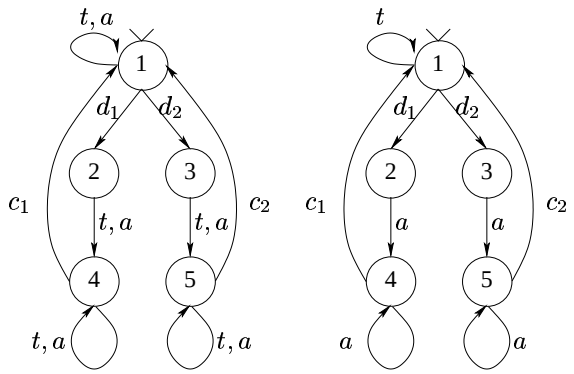
Simplified models of the uncontrolled and legal behaviors in this example are shown in Fig. 3. Figure 3(a) shows the automaton model, G, of the uncontrolled system, while Fig. 3(b) shows the automaton model, H, of the legal behavior. Observe that $\mathcal{L}_m(H) = \mathcal{L}(H)$ and $\mathcal{L}_m(G) = \mathcal{L}(G)$ (i.e., marking is omitted for all states). The events in G are defined as follows:

d_1	:	An object is detected by radar station 1
d_2	:	An object is detected by radar station 2
c_1	:	An object has cleared the range of radar station 1
c_2	:	An object has cleared the range of radar station 2
t	:	Allow take-off
a	:	Issue alert

The set of locally controllable and observable events are specified as follows:

$$E_{o,1} = \{d_1, c_1, t, a\},\ E_{o,2} = \{d_2, c_2, t, a\},\ E_{c,1} = E_{c,2} = \{t, a\}.$$

Since $\epsilon \in \mathcal{L}(H)$, $a \in \mathcal{L}(G) \setminus \mathcal{L}(H)$,



(a) G (b) H

Figure 3. Traffic control example.

$$d_2 a \in P_1^{-1} P_1(\epsilon) a \cap \mathcal{L}(H) \text{ and}$$
$$d_1 a \in P_2^{-1} P_2(\epsilon) a \cap \mathcal{L}(H).$$

We conclude that $\mathcal{L}(H)$ is not C&P co-observable w.r.t. $\mathcal{L}(G)$, $E_{o,1}$, $E_{c,1}$, $E_{o,2}$, $E_{c,2}$. Therefore, we cannot achieve this desired behavior exactly with the conjunctive architecture.

Let us consider a *disjunctive* architecture where the conjunctive fusion block in Fig. 2 is replaced by a disjunctive fusion block, meaning that an event is enabled whenever at least one of the local supervisors enables it. The analogue of C&P co-observability for the disjunctive architecture is called D&A co-observability and it is defined as follows [Yoo and Lafortune, 2000a].

Definition 4 (**D&A co-observability**) *A language $K \subseteq M = \overline{M}$ is said to be D&A co-observable w.r.t. M, $E_{o,1}$, $E_{c,1}$,..., $E_{o,n}$, $E_{c,n}$, if $\forall s \in \overline{K}$ and $\forall \sigma \in E_c = \cup_{i=1}^{n} E_{c,i}$ s.t. $s\sigma \in \overline{K}$,*

$$(\exists i \in \{1, \ldots, n\})[[(P_i^{-1} P_i(s) \cap \overline{K})\sigma \cap M \subseteq \overline{K}] \wedge [\sigma \in E_{c,i}]].$$

The "D" in D&A co-observability stands for *disjunctive* because D&A co-observability is formulated for the disjunctive architecture. Furthermore, the "A" in D&A co-observability stands for *antipermissive* because individual events are always disabled by a local supervisor whenever that supervisor is unsure if the events should be enabled. The intuitive meaning of the antipermissive rule is to permit the occurrence of a controllable event after a trace s has occurred only if the local supervisor has sufficient information to determine with certainty based on an "estimate" of the behavior, $P_i^{-1}P_i(s) \cap \overline{K}$, that enabling the controllable event will be legal.

Let us return to the traffic example depicted in Figs. 3(a) and 3(b). Since $t \in \mathcal{L}(H)$,

$$d_1 t \in (P_1^{-1}P_1(\epsilon) \cap \mathcal{L}(H))t \cap \mathcal{L}(G) \not\subseteq \mathcal{L}(H) \text{ and}$$
$$d_2 t \in (P_2^{-1}P_2(\epsilon) \cap \mathcal{L}(H))t \cap \mathcal{L}(G) \not\subseteq \mathcal{L}(H),$$

we conclude that $\mathcal{L}(H)$ is not D&A co-observable w.r.t. $\mathcal{L}(G)$, $E_{o,1}$, $E_{c,1}$, $E_{o,2}$, $E_{c,2}$. Therefore, we cannot achieve this desired behavior with the disjunctive architecture.

Consider a more general decentralized control architecture where the control actions of the individual supervisors are combined in a more flexible manner than in Fig. 2. Namely, the supervisors agree *a priori* about choosing "fusion by union" (of enabled events) for certain controllable events ($E_{c,d}$) and "fusion by intersection" for the other controllable events ($E_{c,e}$), as shown in Fig. 4. This control architecture is more pow-

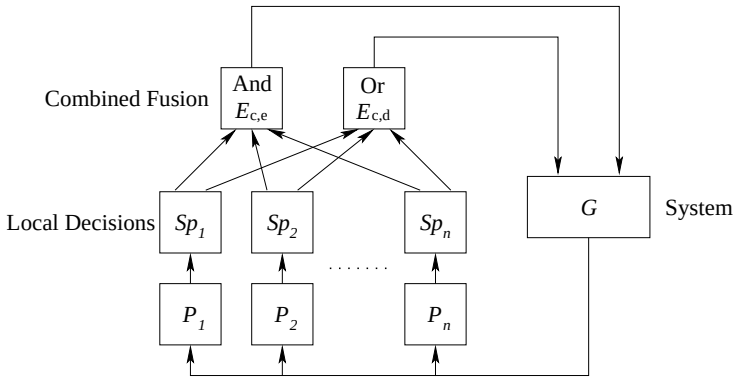


Figure 4. A more general decentralized control architecture.
The local control actions of the individual supervisors are combined by conjunction (intersection of enabled events) for some of the controllable events and by disjunction (union of enabled events) for the remaining controllable events.

erful than the purely conjunctive and disjunctive ones in the sense that

a relaxed version of co-observability appears in the necessary and sufficient conditions for the existence of a set of supervisors that achieves a given legal language [Yoo and Lafortune, 2000a].

In order to present this relaxed version of co-observability, let us define the following sets of events: For $i \in \{1, \ldots n\}$,

$$E_{c,e,i} := E_{c,i} \cap E_{c,e} \text{ and } E_{c,d,i} := E_{c,i} \cap E_{c,d},$$

where $E_{c,d} \dot{\cup} E_{c,e} = E_c$. $E_{c,e,i}$ is the set of locally controllable events whose default setting is enablement while $E_{c,d,i}$ is the set of locally controllable events whose default setting is disablement. We generalize C&P and D&A co-observability to embrace the partition of E_c; we call this generalized notion "co-observability" for the sake of simplicity.

Definition 5 (**Co-observability**) *A language $K \subseteq M = \overline{M}$ is said to be co-observable w.r.t. M, $E_{o,1}$, $E_{c,d,1}$, $E_{c,e,1}$, $E_{o,2}$, $E_{c,d,2}$, $E_{c,e,2}$,..., $E_{o,n}$, $E_{c,d,n}$, $E_{c,e,n}$, if*

1. *K is C&P co-observable w.r.t. M, $E_{o,1}$, $E_{c,e,1}, \ldots,$ $E_{o,n}$, $E_{c,e,n}$,*
2. *K is D&A co-observable w.r.t. M, $E_{o,1}$, $E_{c,d,1}, \ldots,$ $E_{o,n}$, $E_{c,d,n}$.*

With this generalized notion of co-observability, the existence result of the general architecture can be presented. The joint action of the local supervisors $S_{P_1}, ..., S_{P_n}$ in Fig. 4 is denoted by S_{gdec}.

Theorem 2 *Consider a language $K \subseteq \mathcal{L}_m(G)$ where $K \neq \emptyset$ and consider a fixed partition of E_c such that $E_c = E_{c,d} \dot{\cup} E_{c,e}$. There exists a nonblocking generalized supervisor S_{gdec} such that $\mathcal{L}_m(S_{gdec}/G) = K$ and $\mathcal{L}(S_{gdec}/G) = \overline{K}$ iff the three following conditions hold:*
1.K is controllable w.r.t. $\mathcal{L}(G), E_{uc}$.
2.K is co-observable w.r.t. $\mathcal{L}(G), E_{o,1}, E_{c,d,1}, E_{c,e,1}, \ldots, E_{o,n}, E_{c,d,n}, E_{c,e,n}$.
3.$K = \overline{K} \cap \mathcal{L}_m(G)$.

For the S_{gdec} supervisor, gdec stands for "General Decentralized Control Law". The proof of Theorem 2 is given in [Yoo and Lafortune, 2002]. It is constructive and the formula for the actions of the local supervisors composing S_{gdec} is given in the following equation.

$$\begin{aligned} \gamma_i^{gdec}(s) \quad = \quad & \{\sigma \in E_{c,d,i} : \left(P_i^{-1}(P_i(s)) \cap \overline{K}\right)\sigma \cap \mathcal{L}(G) \subseteq \overline{K}\} \\ & \cup \{\sigma \in E_{c,e,i} : P_i^{-1}(P_i(s))\sigma \cap \overline{K} \neq \emptyset\} \\ & \cup E_{uc} \cup E_{c,e} \setminus E_{c,i} \end{aligned} \tag{1}$$

The local supervisors using gdec enable permissive events that might lead to legal behavior and disable antipermissive events that never lead

to illegal behavior. Notice that the gdec control scheme uses $P_i^{-1}(P_i(s))$ as an estimate of system behavior that might occur and the supervisor needs to account for when deciding its next action. We discuss improvements in this state estimator later in the paper.

Again, consider the traffic control example depicted in Figs. 3(a) and 3(b). Let us set $E_{c,d} = \{a\}$ and $E_{c,e} = \{a\}$. With this setting, we can verify that $\mathcal{L}(H)$ is co-observable w.r.t. $\mathcal{L}(G)$, $E_{o,1}$, $E_{c,d,1}$, $E_{c,e,1}$, $E_{o,2}$, $E_{c,d,2}$, $E_{c,e,2}$. Since $\mathcal{L}(H)$ is controllable w.r.t. $\mathcal{L}(G)$, E_{uc} and $\mathcal{L}_m(G)$-closed as well, we can achieve the desired behavior with the general decentralized control architecture. This relaxed version of co-observability is also verifiable in polynomial time [Yoo and Lafortune, 2000a], building on the results in [Rudie and Willems, 1995].

Let us define the following classes of languages where M is assumed to be prefix-closed:

$\mathcal{L}_{cen}(K) = \{L \subseteq K : L$ is observable w.r.t. $M, E_o, E_c\}$,

$\mathcal{L}_{DA}(K) = \{L \subseteq K : L$ is D&A co-observable w.r.t. $M, E_{o,1}, E_{c,1}, E_{o,2}, E_{c,2}\}$,

$\mathcal{L}_{CP}(K) = \{L \subseteq K : L$ is C&P co-observable w.r.t. $M, E_{o,1}, E_{c,1}, E_{o,2}, E_{c,2}\}$,

$\mathcal{L}_{gdec}(K) = \{L \subseteq K : \exists E_{c,d}$ and $E_{c,e}$ s.t. $E_{c,d} \dot{\cup} E_{c,e} = E_c$ and L is co-observable w.r.t. $M, E_{o,1}, E_{c,d,1}, E_{c,e,1}, E_{o,2}, E_{c,d,2}, E_{c,e,2}\}$.

Since the controllability of the desired language is a common required condition for the existence of supervisors among all architectures, the classes of languages defined above determine the performance (the class of achievable languages) of the architectures. The relations between these classes of languages are summarized by the Venn diagram in Fig. 5.

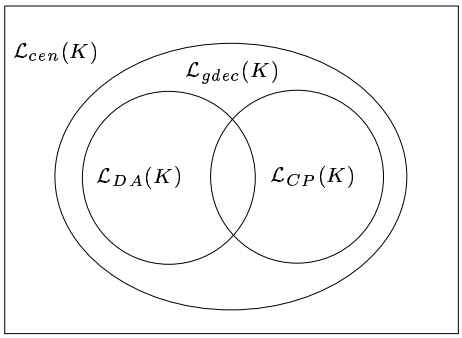


Figure 5. Performance comparison.

For the general decentralized control architecture, the partition of controllable events, $E_{c,d}$ and $E_{c,e}$, is fixed. As an alternative to this fixed

partition, we may consider a "dynamic" partition of controllable events in the sense that the partition varies as the system evolves. Certainly, an architecture with a dynamic partition outperforms an architecture with fixed partition. The synthesis of dynamic partitions in order to achieve a larger class of languages than $\mathcal{L}_{gdec}(K)$ in the context of the general decentralized control architecture is the subject of current research [Yoo, 2001].

4. Synthesis of Safe Solutions

If no partition of the controllable events can ensure co-observability for the architecture in Fig. 4, we are faced with the problem of synthesizing supervisors that ensure the safety of the controlled behavior for some partition.

It has recently been shown in [Lamouchi and Thistle, 2000] and [Tripakis, 2001] that the problem of synthesizing safe and nonblocking *decentralized* supervisors is undecidable.[1] We therefore ignore the blocking problem and focus only on the problem of synthesizing safe supervisors.

Is there some way we could partition the controllable events so that we could guarantee that $\mathcal{L}(S_{gdec}/G) \subseteq \overline{K}$? From the work in [Rudie and Wonham, 1992], we know that if all events are assigned to be permissive, $\mathcal{L}(S_{gdec}/G)$ will be safe if and only if the system is C&P co-observable. We also know that all events assigned to $E_{c,d}$ will never cause illegal behavior. In order to guarantee that a system is safe for gdec and save the trouble of identifying events that violate C&P co-observability, why not assign all events to $E_{c,d}$? This seems like an reasonable plan, but as more events are assigned to $E_{c,d}$, the generated language becomes progressively smaller [Yoo and Lafortune, 2000b]. We should therefore assign as few events as possible to $E_{c,d}$ in order to generate the largest possible safe language.

It therefore stands to reason that if all events that violate C&P co-observability are assigned to $E_{c,d}$, the system will be safe for a gdec control system. This statement has been proven in [Yoo and Lafortune, 2002]. A polynomial time method to identify all controllable events that violate C&P co-observability is discussed in [Rudie and Willems, 1995]. This method consists of constructing a nondeterministic machine $\mathcal{M}$ such that the set of terminal events of the traces in $\mathcal{L}_M(\mathcal{M})$, denoted by $E_{ter}(\mathcal{L}_M(\mathcal{M}))$, identifies all events that violate C&P co-observability. However, not all events identified by $E_{ter}(\mathcal{L}_M(\mathcal{M}))$ need be assigned to $E_{c,d}$ to guarantee safe solutions as discussed in [Yoo and Lafortune, 2002]. It is still an open problem to find partitions on the controllable

events that can be used to generate safe and maximal languages with gdec.

4.1 Improving on gdec

It is possible to improve upon the gdec supervisor discussed in the previous subsection. We first recognize that the local supervisors have knowledge of their own local control actions, where the control policy of supervisor i is denoted by $\gamma_i(\cdot)$. Knowledge of these local control actions could be used to generate system state estimates better than the inverse projections used by gdec (cf. Eqn.(1)). It should be obvious that if a supervisor were to disable an event that it knows no one else would enable globally, then the supervisor can disregard any behavior after that disabled event when calculating later system state estimates. We formalize this logic with the following system state estimation functions introduced in [Rohloff and Lafortune, 2001].

$$PS_i(s) = \begin{cases} [(\gamma_i(\varepsilon) \cup E_{c,d}^{-i}) \cap E_{uo,i}^* \cap \mathcal{L}(G) & \text{for } s = \epsilon \\ (PS_i(s')P_i(\sigma)[(\gamma_i(s'\sigma) \cup E_{c,d}^{-i}) \cap E_{uo,i}]^*) \cap \mathcal{L}(G) & \text{for } s = s'\sigma, s \neq \epsilon \end{cases}$$

$$PS_i^+(s) = \begin{cases} E_{uo,i}^* \cap \mathcal{L}(G) & \text{for } s = \epsilon \\ (PS_i(s')P_i(\sigma)E_{uo,i}^* \cap \mathcal{L}(G) & \text{for } s = s'\sigma, s \neq \epsilon \end{cases}$$

We use the notation that $E_{c,d}^{-i} \equiv \bigcup_{j \in I, j \neq i} E_{c,d,j}$; all of these events could be globally enabled by supervisors other than supervisor i. $PS_i(s)$ represents an estimate of what behavior i should believe may have been generated by the system after the trace of events s has occurred. The $PS_i^+(s)$ function represents what might have occurred if after the last event in s all behavior is uncontrolled. The $PS_i(\cdot)$ and $PS_i^+(\cdot)$ functions are necessarily recursive because these functions update as events are observed and control actions are taken.

It is proved in [Rohloff and Lafortune, 2001] that $PS_i(s)$ and $PS_i^+(s)$ always contain s and that these functions always perform at least as well as the function $P_i^{-1}(P_i(s))$.

When manipulating a system, a local supervisor needs to look at what behavior might occur if no supervisor was present. The gdec can be thought of as taking a "shotgun" approach to local control - it looks at what the system might do if there were no control at all after the last observed event and from that estimate, gdec decides the control actions all at once. We therefore need the $PS_i^+(\cdot)$ function that will return an

estimate of what the system might do if all events after the last observed event were uncontrolled.

We can now define a new control scheme that makes use of these new estimators. We call this control scheme "gmdec" for "General Memory-based Decentralized Control Law".

$$\begin{aligned}\gamma_i^{gmdec}(s) &= \left\{\sigma \in E_{c,d,i} : \left(PS_i^+(s) \cap \overline{K}\right)\sigma \cap \mathcal{L}(G) \subseteq \overline{K}\right\} \\ &\quad \cup \left\{\sigma \in E_{c,e,i} : PS_i^+(s)\sigma \cap \overline{K} \neq \emptyset\right\} \cup E_{uc} \cup E_{c,e} \setminus E_{c,e,i}.\end{aligned}$$

Notice that gmdec is the same as gdec except for the estimation method. Because of the improved knowledge of gmdec, it is shown in [Rohloff and Lafortune, 2001] that:
(i) when $\mathcal{L}(S_{gdec}/G)$ is safe, $\mathcal{L}(S_{gdec}/G) \subseteq \mathcal{L}(S_{gmdec}/G)$ and
(ii) if all events that violate C&P co-observability are assigned to be antipermissive, then gmdec generates a safe language.

There are other ways to possibly improve upon gdec and gmdec, most notably by enabling events in a greedy iterative manner rather than the all-at-once approach as was previously used. The reader is referred to [Rohloff and Lafortune, 2001] for further results in this regard.

To better understand these new control schemes, consider the following example.

Example 1 *Suppose there are two supervisors for the uncontrolled system G seen in Fig. 6. Suppose further that $K = \mathcal{L}(H)$. The supervisors have the following properties:*

$$E_{o,1} = \{\phi\},\ E_{o,2} = \{\beta\},\ E_{c,1} = \{\beta, \alpha\},\ E_{c,2} = \{\lambda, \theta\},\ E_{uc} = \{\phi, \omega\},$$
$$E_{c,e} = \emptyset,\ E_{c,d} = \{\alpha, \beta, \lambda, \phi\}.$$

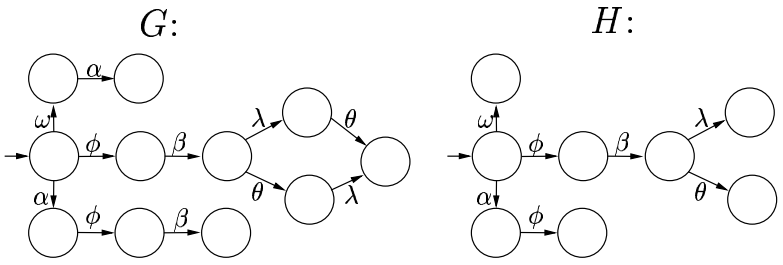


Figure 6. Example 1.

Let us calculate $\mathcal{L}(S_{gdec}/G)$:
$P_1^{-1}(P_1(\varepsilon)) \cap \mathcal{L}(G) = \{\varepsilon, \alpha, \omega, \omega\alpha\}$
$\gamma_1^{gdec}(\varepsilon) = \{\beta, \omega, \phi\}$

$P_1^{-1}(P_1(\phi)) \cap \mathcal{L}(G) = \{\alpha\phi, \alpha\phi\beta, \phi, \phi\beta, \phi\beta\lambda, \phi\beta\lambda\theta, \phi\beta\theta, \phi\beta\theta\lambda\}$
$\gamma_1^{gdec}(\phi) = \{\alpha, \omega, \phi\}$
$P_2^{-1}(P_2(\varepsilon)) \cap \mathcal{L}(G) = \{\varepsilon, \alpha, \alpha\phi, \phi, \omega, \omega\alpha\}$
$\gamma_2^{gdec}(\varepsilon) = \{\lambda, \theta, \omega, \phi\}$.
Therefore, $\mathcal{L}(S_{gdec}/G) = \{\varepsilon, \omega, \phi\}$. *This system can be seen in Fig. 7. Next, we calculate* $\mathcal{L}(S_{gmdec}/G)$:
$PS_1^+(\varepsilon) = \{\varepsilon, \alpha, \omega, \omega\alpha\}$
$\gamma_1^{gmdec}(\varepsilon) = \{\beta, \omega, \phi\}$
$PS_1(\phi) = \{\phi, \phi\beta, \phi\beta\lambda, \phi\beta\lambda\theta, \phi\beta\theta, \phi\beta\theta\lambda\}$
$\gamma_1^{gmdec}(\phi) = \{\alpha, \beta, \omega, \phi\}$
$PS_2^+(\varepsilon) = \{\varepsilon, \alpha, \alpha\phi, \phi, \omega, \omega\alpha\}$
$\gamma_2^{gmdec}(\varepsilon) = \{\lambda, \theta, \omega, \phi\}$
$PS_2(\varepsilon) = \{\varepsilon, \alpha, \alpha\phi, \phi, \omega, \omega\alpha\}$
$PS_2^+(\phi\beta) = \{\alpha\phi\beta, \phi\beta, \phi\beta\lambda, \phi\beta\lambda\theta, \phi\beta\theta, \phi\beta\theta\lambda\}$
$\gamma_2^{gmdec}(\phi\beta) = \{\phi, \omega\}$
$PS_2(\phi\beta) = \emptyset$.
Therefore, $\mathcal{L}(S_{gmdec}/G) = \{\varepsilon, \omega, \phi, \phi\beta\}$. *This system can be seen in Fig. 7. Obviously, for this example,* $\mathcal{L}(S_{gdec}/G) \subset \mathcal{L}(S_{gmdec}/G)$.

It is also possible to synthesize a safe supervisor that generates a language strictly larger than gmdec for this example. The control scheme that synthesizes such a supervisor is called VLP-GM2 and is discussed in [Rohloff and Lafortune, 2001]. This control scheme locally enables events in an iterative manner and results in the following control actions:
$\gamma_1^{VLPGM2}(\epsilon) = \{\beta, \omega, \phi\}$
$\gamma_1^{VLPGM2}(\phi) = \{\alpha, \beta, \omega, \phi\}$
$\gamma_2^{VLPGM2}(\varepsilon) = \{\lambda, \theta, \omega, \phi\}$
$\gamma_2^{VLPGM2}(\phi\beta) = \{\phi, \omega, \lambda\}$.
Therefore, $\mathcal{L}(S_{VLPGM2}/G) = \{\varepsilon, \omega, \phi, \phi\beta, \phi\beta\lambda\}$. *This system can be seen in Fig. 7. VLP-GM2 places a prioritization on locally enabling events. For this example, we arbitrarily allow* λ *to be more important than* θ.

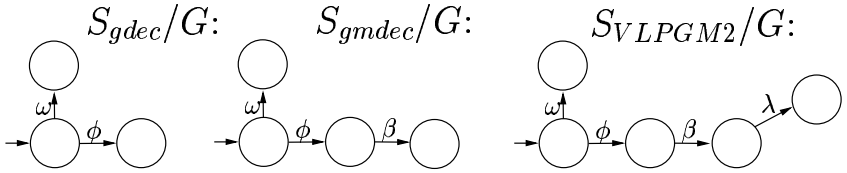


Figure 7.

5. Conclusion

The fundamentals and some recent results on supervisory control of partially-observed discrete-event systems have been reviewed. Decentralized supervisory control is an area with many open problems. In view of the undecidability results on the synthesis of safe and nonblocking solutions in decentralized control, more structure will have to be introduced in these problems if progress is to be made.

Acknowledgments

The first author would like to thank the organizers of the "Symposium on the Supervisory Control of Discrete Event Systems - SCODES'2001" for their invitation to present this paper at the symposium.

Notes

1. In [Yoo, 2001], it is shown that the problem of synthesizing a safe and nonblocking *centralized* supervisor is decidable.

References

[Barrett, 1999] Barrett, G. (1999). *Modeling, analysis and control of centralized and decentralized logical discrete-event systems.* PhD thesis, The University of Michigan.

[Cassandras and Lafortune, 1999] Cassandras, C. G. and Lafortune, S. (1999). *Introduction to Discrete Event Systems.* Kluwer Academic Publishers.

[Cieslak et al., 1988] Cieslak, R., Desclaux, C., Fawaz, A., and Varaiya, P. (1988). Supervisory control of discrete event processes with partial observation. *IEEE Trans. on Automat. Contr.*, 33(3):249–260.

[Debouk et al., 1999] Debouk, R., Lafortune, S., and Teneketzis, D. (1999). On an optimization problem in sensor selection for failure diagnosis. In *Proc. 38th IEEE Conf. on Decision and Control*, pages 4990–4995.

[Lamouchi and Thistle, 2000] Lamouchi, H. and Thistle, J. (2000). Effective control synthesis for discrete event systems under partial observations. In *Proc. 39th IEEE Conf. on Decision and Control*, pages 22–28.

[Ramadge and Wonham, 1987] Ramadge, P. J. and Wonham, W. M. (1987). Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25(1):206–230.

[Ramadge and Wonham, 1989] Ramadge, P. J. and Wonham, W. M. (1989). The control of discrete event systems. *Proc. IEEE*, 77(1):81–98.

[Ricker and Rudie, 2000] Ricker, S. L. and Rudie, K. (2000). Know means no: Incorporating knowledge into discrete-event control systems. *IEEE Trans. on Automat. Contr.*, 45(9):1656–1668.

[Rohloff and Lafortune, 2001] Rohloff, K. and Lafortune, S. (2001). Advances in state estimation and controller synthesis for general decentralized control. Technical Report CGR-01-11, The University of Michigan.

[Rudie and Willems, 1995] Rudie, K. and Willems, J. C. (1995). The computational complexity of decentralized discrete-event control problems. *IEEE Trans. on Automat. Contr.*, 40(7):1313–1318.

[Rudie and Wonham, 1992] Rudie, K. and Wonham, W. M. (1992). Think globally, act locally: Decentralized supervisory control. *IEEE Trans. on Automat. Contr.*, 37(11):1692–1708.

[Thistle, 1996] Thistle, J. G. (1996). Supervisory control of discrete event systems. *Mathematical and Computer Modelling*, 23(11/12):25–53.

[Tripakis, 2001] Tripakis, S. (2001). Undecidable problems of decentralized observation and control. In *Proc. 40th IEEE Conf. on Decision and Control*.

[Tsitsiklis, 1989] Tsitsiklis, J. N. (1989). On the control of discrete-event dynamical systems. *Math. Control Signals Systems*, 2:95–107.

[Yoo and Lafortune, 2000a] Yoo, T. and Lafortune, S. (2000a). A general architecture for decentralized supervisory control of discrete-event systems. In *Discrete Event Systems: Analysis and Control*, pages 111–118. Kluwer Academic Publishers.

[Yoo and Lafortune, 2000b] Yoo, T. and Lafortune, S. (2000b). New results on decentralized supervisory control of discrete-event systems. In *Proc. 39th IEEE Conf. on Decision and Control*, pages 1–6.

[Yoo and Lafortune, 2002] Yoo, T. and Lafortune, S. (2002). A general architecture for decentralized supervisory control of discrete-event systems. *Discrete Event Dynamical Systems: Theory and Applications*. In print.

[Yoo, 2001] Yoo, T.-S. (2001). *Monitoring and Control of Partially-Observed Discrete-Event Systems*. PhD thesis, The University of Michigan. In preparation.

[Yoo and Lafortune, 2001] Yoo, T.-S. and Lafortune, S. (2001). On the computational complexity of some problems arising in partially-observed discrete-event systems. In *Proc. 2001 American Control Conf.*, pages 307–312.