15th Triennial World Congress, Barcelona, Spain

TWO OBSERVATIONS REGARDING COMMUNICATION IN DECENTRALIZED SUPERVISORY CONTROL

S.L. Ricker *,1

* *Department of Mathematics and Computer Science, Mount Allison University, 67 York Street, Sackville, NB E4L 1E6 Canada*
lricker@mta.ca

Abstract: Two assumptions regarding the synthesis of communication and control policies for decentralized discrete-event control problems are revisited. In particular, the construction of information structures that yield certain communication strategies relies on a specific relationship between the languages describing the plant and the legal behavior as well as the respective automata for these languages. An example that satisfies part of the initial assumption but generates a invalid information structure is presented. Secondly, clarification of an assumption on the finite nature of some additional procedures used in generating communication and control policies is provided.

Keywords: Decentralized control, discrete-event systems, communication protocols, automata theory.

1. INTRODUCTION

Establishing communication protocols for decentralized controllers of discrete-event systems has been the focus of recent investigation. These protocols range from full disclosure of information among controllers (e.g., pooling of partial observations or broadcast) (Wong and van Schuppen 1996), (van Schuppen 1998), (Ricker and Rudie 2000*b*), (Barrett and Lafortune 2000) to communication of observations from one controller to another along a two-way communication channel — but not necessarily always two-way broadcast — in a "minimal" fashion (Rudie *et al.* 1999), (Ricker and Rudie 1999), (Ricker and Rudie 2000*a*) to the communication of a "flag" that indicates that *something* has been observed without disclosing the specific observation (Ricker and Barrett 2001). It is worth noting that in all these cited works, communication is assumed to occur without any delay in the channel.

These protocols rely on the construction of an information structure to organize the partial observations of the decentralized controllers. In (Ricker and Rudie 1999) there are several informal conjectures regarding the nature of the specification of the plant and the legal language which allowed the design of communication protocols for a particular class of decentralized problems. In particular, there was an assumption that the legal automaton was always expressed as a subautomaton of the plant. Secondly, the paper stated that even when the language was not finite (i.e., the automaton contains cycles), that to solve the control problem, it was possible to consider only a finite number of sequences when trying to identify situations where a controller should communicate. In the discussion that follows, the "subautomaton assumption" is

[1] This research is supported in part by research grants from NSERC and Mount Allison University.

explained through the use of an example. Additionally, an algorithm is provided to address the latter conjecture.

2. BACKGROUND

The results in this paper follow the formulation of discrete-event systems as initiated in (Ramadge and Wonham 1987). Only the briefest introduction to the notation in this paper will be presented. For more comprehensive background material, the reader is referred to (Cassandras and Lafortune 1999).

The behavior of the system requiring control, the *plant*, is represented by sequences constructed from a non-empty set of symbols called an *alphabet*. The alphabet represents the set of all possible *events* that can occur within the system. Transitions from one system state to another do not depend on the passage of time, but rather, on the occurrence of an event. The goal is to develop a control strategy for an overseer, or *supervisor*, that will constrain the behavior of the plant to that of a pre-specified sublanguage (the legal language). The supervisor averts undesirable behavior of the plant by either preventing some events from taking place or allowing—but not forcing—others to occur.

More formally, the plant is modelled by an automaton

$$G = (Q^G, \Sigma, \delta^G, q_0^G),$$

where Q^G is a set of *states*; Σ is the alphabet; δ^G is the *transition function*, a partial function $\delta^G: \Sigma \times Q^G \rightarrow Q^G$; and $q_0^G \in Q^G$ is the *initial state*. The definition for δ^G can be extended to a partial function for $\Sigma^* \times Q^G$ such that $\delta^G(\varepsilon, q^G) := q^G$ and $(\forall \sigma \in \Sigma)(\forall t \in \Sigma^*)\ \delta^G(t\sigma) := \delta^G(\sigma, \delta^G(t, q_0^G))$. The set Σ^* contains all possible finite strings (i.e., sequences) over Σ plus the null string ε. The language generated by G, denoted $L(G)$, is also called the *closed behavior* of G:

$$L(G) := \{t \mid t \in \Sigma^* \text{ and } \delta^G(t, q_0^G) \text{ is defined}\}.$$

This language describes all possible event sequences that the discrete-event system can undergo. Thus $L(G) \subseteq \Sigma^*$.

The automaton describing the legal behavior, is denoted by $E = (Q^E, \Sigma, \delta^E, q_0^E)$.

The control problem of interest in this paper assumes that there are n supervisors responsible for making control decisions about the system. Each supervisor $\mathcal{S}_i$ has a partial view of the system and observes only events in $\Sigma_{i,o} \subseteq \Sigma$ and controls only events in $\Sigma_{i,c} \subseteq \Sigma$, for $i = 1, \ldots, n$. We consider here only two local supervisors. To describe a decentralized supervisor's view of the plant, the canonical projection P_i from Σ^* to $\Sigma_{i,o}^*$ is used, for $i = 1, 2$.

The projection operator P_i assumes that a supervisor is tracking only the partial view of the current sequence generated by the plant. Since a supervisor cannot see every event, there may be uncertainty as to the exact state the plant is in. A supervisor could keep track of the possible states the plant could be in, rather than (or in addition to) keeping track of a sequence. As an example, suppose that the plant is in state x and the occurrence of event σ would lead the plant to state y (i.e., $\delta^G(\sigma, x) = y$). If a supervisor cannot observe σ, the supervisor will not know whether the plant is in state x or y. Consequently, we could describe a supervisor's view of the current state of the plant as a set that includes x and y. To capture the view that $\mathcal{S}_i$ has of the plant, we use an *observer automaton* (Cassandras and Lafortune 1999), based on an algorithm in (Hopcroft and Ullman 1979) to translate a nondeterministic finite-state automaton into a deterministic finite-state automaton.

We will also find it convenient to construct a finite-state machine that allows us to simultaneously track the current state of the plant and the current state of each supervisor's projected view of the plant (via the observer automaton). Such a structure, which we call a *monitoring automaton* and denote by A, is a deterministic version of the nondeterministic automaton M described in (Rudie and Willems 1995)[2]. A complete characterization of A is found in (Ricker and Rudie 1999). By the way in which A is constructed, we have $L(A) = L(G)$.

The strategy for synthesizing control and communication policies that will be used in this paper appeared previously in (Ricker and Rudie 1999). In lieu of including all the technicalities here, the procedure will be illustrated through an example in subsequent sections. The important point to recall for subsequent sections is that supervisors are not communicating sequences of events that they have observed. Rather, when a supervisor communicates, it sends its current set of state estimates — states it considers the plant could be in based on its partial observations of plant behavior. Finally, all the cited works assume communication protocols can only be constructed if the system satisfies *observability* (Lin and Wonham 1988).

[2] A variation of this automaton, called an *estimator structure* appears in (Barrett and Lafortune 2000).

3. ASSUMPTIONS

Previously, in (Ricker and Rudie 1999), it was assumed that not only was $L(E) \subseteq L(G)$ but E was a subautomaton of G as described in the context of supervisory control in (Cho and Marcus 1989) and (Lafortune and Chen 1990). This assumption will now be justified.

Assumption 1 E is a subautomaton of G such that $Q^E \subseteq Q^G$, $q_0^E = q_0^G$ and $\delta^E(t, q_0^G) = \delta^G(t, q_0^G)$ for all $t \in L(E)$. This implies that $L(E) \subseteq L(G)$ but the converse is not true (Lafortune and Chen 1990). Note that in this case, it is always possible to alter E and G so that this additional condition is satisfied without changing the overall plant and legal languages (see (Cassandras and Lafortune 1999) for an algorithm).

Assumption 2 It is possible to consider only a finite number of sequences in $L(A)$ to construct a decentralized control and communication policy.

3.1 Justifying Assumption 1

Figure 1 contains a plant G, where $L(G) = (ab)^* f (aab)^*$. Suppose that we want to design a control and communication policy for two controllers when the legal language $L(E) = (ab)^* faab$, where $\Sigma = \{a, b, f\}$ such that $\Sigma_{1,o} = \{a\}$ and $\Sigma_{2,o} = \{b\}$. Let $\Sigma_{1,c} = \{a\}$.

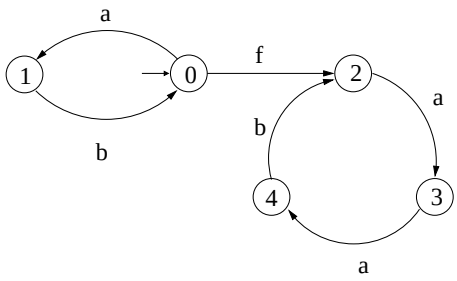

Fig. 1. Plant G generating language $L(G) = (ab)^* f (aab)^*$.

Suppose that $t' = abf$ and $t'' = faabaab$: both these sequences lead to state 2, even though t' is legal and t'' is illegal. Why should this cause any difficulty? If the supervisors simply communicate observed events, then it is possible to see that if $\mathcal{S}_2$ communicates after it sees event b, then $\mathcal{S}_1$ would know to disable a after seeing two consecutive a's followed by a communication of b from $\mathcal{S}_1$. It is not as straightforward to synthesize a control and communication policy for the plant in figure 1 when supervisors communicate sets of state estimates.

The monitoring automaton A for this plant is shown in figure 2. Note that the structure of the

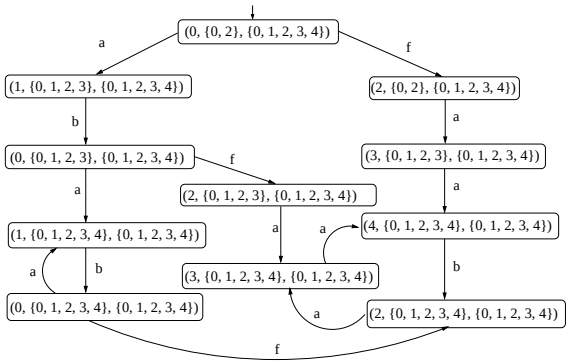

Fig. 2. Monitoring automaton for G in figure 1.

state in the automaton is a triple containing the true state of the plant, followed by the set of state estimates for each supervisor (constructed from the states of the observer automaton). For example, state $(0, \{0, 1, 2, 3\}, \{0, 1, 2, 3, 4\})$ represents the information that after the plant generates the sequence ab, the true state of the plant is 0, but $\mathcal{S}_1$, which sees only event a, cannot determine if the observation of event a means the plant is in state 1, or if perhaps unobservable event f happened before a, in which case the plant would be in state 3, or if after observing a perhaps unobservable event b occurred making the current plant state 0 or maybe even bf has occurred and the resulting plant state is really 2. Similar arguments follow for determining the state estimates of $\mathcal{S}_2$: after seeing b, it considers it possible that the plant is in any of its states.

To generate a control and communication policy, the monitoring automaton is used first to determine whether or not the system is *co-observable* (Rudie and Wonham 1992). That is, we check to see if there could be decentralized supervisors that would solve the control problem without communication. A procedure for identifying which states in the monitoring automaton give rise to a violation of the property of co-observability is described in (Ricker and Rudie 1999). If the system is observable but not co-observable, we can identify sequences along which supervisors should communicate their state estimates, subsequently allowing the other supervisor to determine its correct control decisions. This procedure will be addressed during the discussion of Assumption 2.

Before this test, however, there is a more fundamental structural property of the monitoring automaton that must be avoided. The monitoring automaton A must not contain a state with the following characteristic. Suppose that there exists a state q in A that can be reached via (at least two) paths (i.e., sequences) s and s'. Note that this implies the existence of at least one $\mathcal{S}_i$ that cannot distinguish s from s' (i.e., $\exists i \in \{1, 2\}$ such that $P_i(s) = P_i(s')$). Further suppose that $\exists \sigma \in \Sigma_{i,c}$

such that σ is defined as a transition from state q but that $s\sigma, s'\sigma \in L(G)$ but only $s\sigma \in L(E)$.

Structurally speaking, this creates a problem for the machinations of the construction of a communication/control policy. The strategy that is adopted for testing the feasibility of a communication/control policy assumes that at a given *state* of the plant structure it is possible for an omnipotent supervisor (i.e., a centralized supervisor that observed all observable events) to definitively make a control decision.

In figure 2, state $(2, \{0,1,2,3,4\}, \{0,1,2,3,4\})$ is a problem state. If one approaches this state via the sequence $faab$, then the next occurrence of a must be disabled; however if the state is reached via $ababf$, then the next occurrence of a must be enabled. From supervisor 1's perspective, it has observed the sequence aa. Even if it was aware that the system was at state 2, it could not make the correct control decision because the two options ("disable a" and "enable" a) are both valid!

This does not mean that we cannot generate a communication/control policy for this plant language and legal language. As long as the system is observable, we can continue. The next step is to make E a subautomaton of G. That is, we want to ensure that if there exists $t', t'' \in L(G)$ and $t' \in L(E), t'' \notin L(E)$ then there will not be $q \in Q^E \cap Q^G$ such that $q = \delta^E(t', q_0^E) = \delta^G(t'', q_0^G)$.

The resulting plant G' and legal automaton E' is shown in figure 3. The monitoring automaton for G' is shown in figure 4. The dashed lines indicate that the corresponding transition in G' represents an illegal transition.

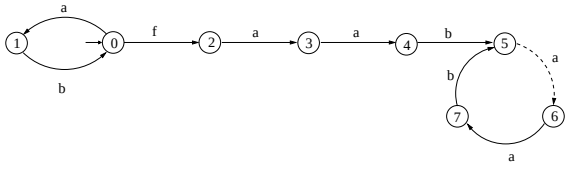

Fig. 3. The new legal automaton (collection of solid-line transitions) and plant automaton (collection of all transitions) for the system in figure 1 that satisfies Assumption 1. Illegal transitions are noted with a dashed line.

3.2 Formalizing Assumption 2

To determine where communication will be incorporated into the system, the first task is to use the monitoring automaton to find sequences that violate co-observability. That is, we want to locate sequences s and s' in $L(G)$, a supervisor $\mathcal{S}_i$ and some event $\sigma \in \Sigma_{i,c}$ such that $P_i(s) = P_i(s')$ and $s\sigma \in L(E)$ but $s'\sigma \notin L(E)$. Therefore, after observing $P_i(s)$ or $P_i(s')$, $\mathcal{S}_i$ cannot make the correct control decision regarding event σ. In the monitoring automaton, sequences that are indistinguishable to some $\mathcal{S}_i$ lead to states in A that have the same set of state estimates. In (Ricker and Rudie 1999) such states where s' occurs are states where it is formally proven that $\mathcal{S}_i$ does not possess sufficient knowledge to make the correct control decision regarding event σ.

When we find a state where a supervisor lacks knowledge, we use this state to reconstruct all such s and s' pairs that satisfy the conditions noted above. In figure 4 these states correspond to states at which an illegal transition is defined. For example, at state $(5, \{0,1,2,3,4,5\}, \{0,1,2,3,4,5,6,7\})$, $\mathcal{S}_1$ would like to disable a after $faab$ occurs. Unfortunately, $\mathcal{S}_1$ cannot distinguish this sequence from $abab$, at state $(0, \{0,1,2,3,4,5\}, \{0,1,2,3,4,5,6,7\})$, after which a must be enabled. Note that the local state estimates for $\mathcal{S}_1$ are the same at both states (i.e., $\{0,1,2,3,4,5\}$). We can find all such sequences by reconstructing the *regular expression* that leads to states where (i) $\mathcal{S}_1$'s local view is $\{0,1,2,3,4,5\}$; and (ii) at that state, a transition of a is defined. In this case, we would not check all paths to state $(1, \{0,1,2,3,4,5\}, \{0,1,2,3,4,5,6,7\})$ since there is no transition involving a defined at that state. There are well-known algorithms, albeit with daunting computational complexity, for calculating the unambiguous regular expressions of sequences in regular languages (e.g., (Hopcroft and Ullman 1979)).

Once the regular expressions have been calculated, it is a simple matter of performing language inclusion tests to isolate those sequences s' and s that will be used for designing the decentralized communication protocol.

For example, the regular expressions to the states in A where $\mathcal{S}_1$ lacks knowledge to make the correct control decisions are: $(5, \{0,1,2,3,4,5\}, \{0,1,2,3,4,5,6,7\})$ – $faab$ and $(5, \{0,1,2,3,4,5,6\}, \{0,1,2,3,4,5,6,7\})$ – $abfaab$+$ababfaab$. Note that the state $(5, \{0,1,2,3,4,5,6,7\}, \{0,1,2,3,4,5,6,7\})$ was excluded. This omission was intentional. This is a state which is reachable by a sequence that has an illegal prefix—namely, $faabaab$—so it is assumed that the correct control decision will be made for the illegal prefix and this state need not be considered any further.

The corresponding states where the local state estimates for $\mathcal{S}_1$ are {0,1,2,3,4,5} and {0,1,2,3,4,5,6} and the regular expressions for all paths to those states are:

- $(0, \{0,1,2,3,4,5\}, \{0,1,2,3,4,5,6,7\})$:$abab$
- $(2, \{0,1,2,3,4,5\}, \{0,1,2,3,4,5,6,7\})$:$ababf$
- $(0, \{0,1,2,3,4,5,6\}, \{0,1,2,3,4,5,6,7\})$:$abababab$

- $(2, \{0,1,2,3,4,5,6\}, \{0,1,2,3,4,5,6,7\})$:*abababf*

It is then straightforward to identify the pairs of sequences s', s that $\mathcal{S}_1$ cannot distinguish: *faab* and *abab*, *faab* and *ababf*, *abfaab* and *ababab*, *abfaab* and *abababf*.

The algorithm for identifying the s', s sequences of interest in A is presented below:

Input: A, $q^A = (q, q_1, q_2)$, $q'^A = (q', q_1', q_2')$, i, $\Sigma_{i,c}$
Precondition: $q_i = q_i'$ and $\exists \sigma \in \Sigma_{i,c}$ such that σ is an illegal transition from $q'A$ but a legal transition from q^A
Output: an automaton M such that $L(M) = s + s'$

(1) Calculate the regular expression from the initial state of A to q^A. Call this ρ_g.
(2) Calculate the regular expression from the initial state of A to q'^A. Call this ρ_b
(3) Let $L(\rho_g)$ and $L(\rho_b)$ be the languages associated with the respective regular expressions.
(4) Calculate $P_i(L(\rho_g))$ and $P_i(L(\rho_b))$.
(5) If $P_i(L(\rho_g)) \cap P_i(L(\rho_b)) \neq \emptyset$
    (a) Calculate $\rho_1 = P^{-1}(P_i(L(\rho_g)))$.
    (b) Calculate $\rho_2 = P^{-1}(P_i(L(\rho_b)))$.
    (c) Build the automaton for ρ_1: $M(\rho_1)$.
    (d) Build the automaton for ρ_2: $M(\rho_2)$.
    (e) Return $M(\rho_1) \times M(\rho_2)$

Note that the output from this algorithm can then be used to determine where to add communication events to A so that $\mathcal{S}_i$ can subsequently take the correct control decision regarding its partial observation of $s\sigma$ and $s'\sigma$.

The strategy for adding communication to a decentralized discrete-event system developed in (Ricker and Rudie 1999) adopts a policy of communicating as early as possible. This corresponds to the first position along the s', s pairs of sequences where one supervisor observes something that, if the set of state estimates is communicated at that point, would allow the other supervisor to eventually distinguish between s' and s and make the correct control decision regarding event σ.

For instance, when $s' = abfaab$ and $s = abababf$, a communication from $\mathcal{S}_2$ to $\mathcal{S}_1$ along s when *abab* occurs (i.e., the set of state estimates to communicate is $\{0,1,2,3,4,5,6,7\}$), this being the first place where s differs from s' to an observer that sees all observable events. A similar procedure would take place for all other s', s pairs.

4. DISCUSSION

This paper presented several clarifications and corrections of assumptions that were made in previous analysis on incorporating communication into decentralized discrete-event control problems. In particular, the assumption of the subautomaton relationship between the legal behavior and the plant behavior was clarified. This was done to reflect some of the idiosyncracies in the topography of information structures that are used to develop communication and control policies for this class of control problems. Secondly, it was suggested that it is of some interest to calculate the regular expressions of all paths from the initial state of the information structure to easily identified illegal states—as well as to those legal states that are indistinguishable from the illegal states. The regular expressions represent all possible sequences that lead to both types of states. Manipulating the language generated by the regular expressions and subsequently dealing with the associated automata means that we need only consider a finite number of families of sequences. The finiteness arises from the observation that the algorithm considers pairs of states in A where correct control decisions cannot be made and A itself is a finite structure. The implication is that there are essentially only a finite number of sequences or families of sequences to consider when explicitly incorporating communication into the description of plant behavior.

ACKNOWLEDGMENTS

Many thanks to Raja Sengupta for supplying the example in figure 1.

REFERENCES

Barrett, G. and S. Lafortune (2000). Decentralized supervisory control with communicating controllers. *IEEE Trans. on Automat. Contr.* **45**(9), 1620–1638.

Cassandras, C. and S. Lafortune (1999). *Introduction to Discrete Event Systems*. Kluwer.

Cho, H. and S.I. Marcus (1989). Supremal and maximal sublanguages arising in supervisor synthesis problems with partial observations. *Mathematical Systems Theory* **22**, 177–211.

Hopcroft, J.E. and J.D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Lafortune, S. and E. Chen (1990). The infimal closed controllable superlanguage and its application in supervisory control. *IEEE Trans. Automat. Contr.* **35**(4), 398–405.

Lin, F. and W.M. Wonham (1988). On observability of discrete-event systems. *Info. Sci.* **44**, 173–198.

Ramadge, P.J. and W.M. Wonham (1987). Supervisory control of a class of discrete event processes. *SIAM J. Contr. Optim.* **25**(1), 206–230.

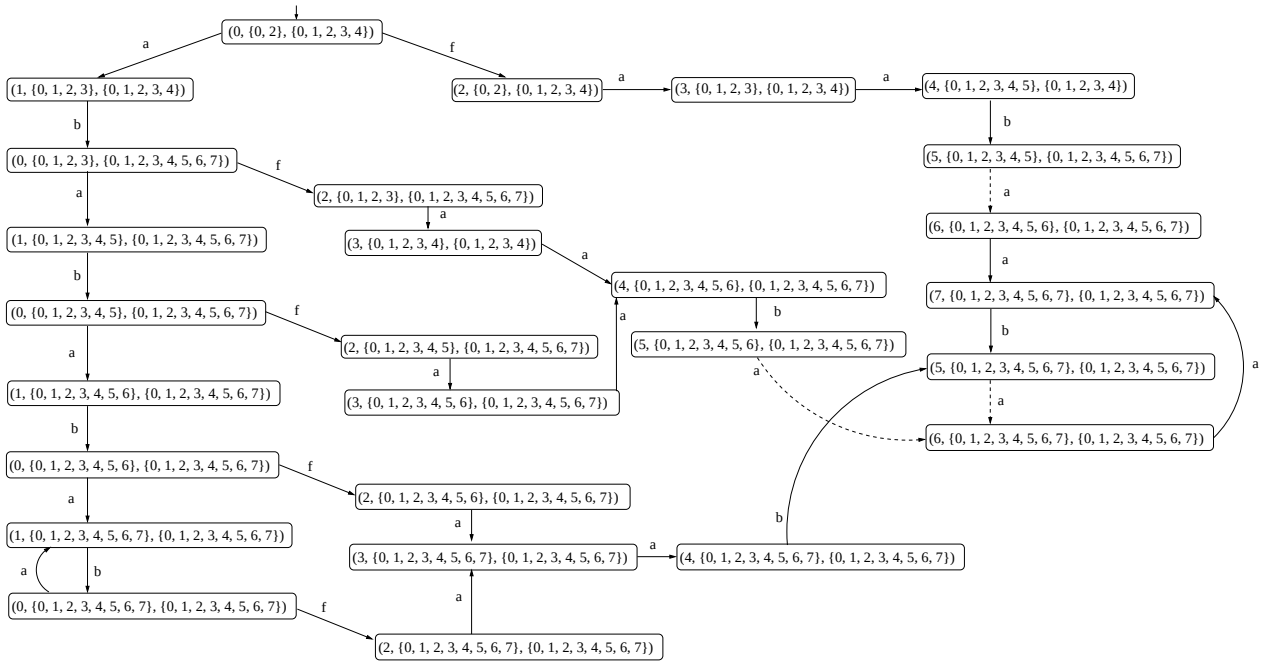

Fig. 4. Monitoring automaton A for the plant G' in figure 3.

Ricker, S.L. and G. Barrett (2001). Decentralized supervisory control with single-bit communications. In: *Proc. Am. Contr. Conf.* pp. 965–966.

Ricker, S.L. and K. Rudie (1999). Incorporating communication and knowledge into decentralized discrete-event systems. In: *Proc. 38th Conf. Decision Contr.* pp. 1326–1332.

Ricker, S.L. and K. Rudie (2000*a*). Distributed knowledge for communication in decentralized discrete-event systems. In: *Proc. 39th Conf. Decision Contr.* pp. 9–15.

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

Rudie, K. and J.C. Willems (1995). The computational complexity of decentralized discrete-event control problems. *IEEE Trans. Automat. Contr.* **40**(7), 1313–1319.

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

Rudie, K., S. Lafortune and F. Lin (1999). Minimal communication in a distributed discrete-event control system. In: *Proc. Am. Contr. Conf.* pp. 1965–1970.

van Schuppen, J.H. (1998). Decentralized supervisory control with information structures. In: *Proc. Int. Workshop on Discrete Event Systems.* pp. 36–41.

Wong, K.C. and J.H. van Schuppen (1996). Decentralized supervisory control of discrete-event systems with communication. In: *Proc. Int. Workshop on Discrete Event Systems.* pp. 284–289.