Automatica 47 (2011) 2364–2372

Contents lists available at SciVerse ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

Mind the gap: Expanding communication options in decentralized discrete-event control☆

Laurie Ricker[a,1], Benoît Caillaud[b]

[a] Department of Mathematics and Computer Science, Mount Allison University, Sackville, NB, Canada

[b] INRIA/IRISA, Rennes, France

ARTICLE INFO

Article history:
Received 21 June 2010
Received in revised form
9 January 2011
Accepted 6 May 2011
Available online 15 September 2011

Keywords:
Discrete-event systems
Communication protocols
Decentralized control

ABSTRACT

Frameworks that incorporate communication into decentralized supervisory control theory address the following problem: find locations in the evolution of the plant behavior where supervisors send information so that a supervisor that was unable to make the correct control decision prior to receiving external information, is now capable of making the correct control decision. Existing solutions to this problem identify an earliest and a latest placement where the communication protocol leads to the synthesis of a correct control solution. In addition to the first and last communication opportunities, there may be a selection of intermediate possibilities where communication would also produce the correct control solution. We present a computable procedure to identify a broader range of suitable communication locations.

1. Introduction

The role of communication (with no delay) in decentralized discrete-event control problems has been explored under a variety of communication protocols. The basic idea of this class of problems is that in the absence of communication, the control objective cannot be met (i.e., the controlled system behavior does not stay within a specified behavior). The goal is to synthesize a communication protocol that the decentralized controllers follow – in addition to observing the behavior of an uncontrolled system – which results in controllers sending information about system behavior to other controllers and/or receiving similar information from other controllers. The communicated information combined with their own partial observations of system behavior, provide the recipients of the communication with enough information to definitively make the correct control decisions, thereby meeting the control objective.

Communication is introduced into a decentralized supervisory control problem when the system requiring control does not satisfy co-observability, a condition that determines whether or not the collection of control decisions for a set of decentralized supervisors allows *all* the correct control decisions to be taken (Rudie & Wonham, 1992). The design of a communication protocol for decentralized control problems requires the construction of a non-empty set of communications that will admit a correct control solution (van Schuppen, 2004). We are interested in bridging the gap between the "communicate as late as possible" strategy of Barrett and Lafortune (2000), where communication to resolve violations of co-observability occurs only along prefixes of behaviors that lead outside of the specification, and the "communicate as early as possible" strategy of Ricker (1999), where communication occurs along prefixes of any of the behaviors involved in a violation of co-observability, not just those that result in undesired behavior.

The "communicate as late as possible" protocol features senders and/or receivers that initiate communication only along behaviors that lead outside of the specification. The algorithm to find the protocol begins by considering the "last" potential communication location as the state where all entering transitions are part of behaviors in the specification (and at least one transition is observed by either the sender or receiver), but there is at least one exiting transition that leaves the specification and is involved in a violation of co-observability. If communication at this state does not result in the correct control decision being taken, the algorithm continues by backtracking along the prefix of this undesired behavior to find a state where communication will lead to a correct control decision. Such a state can always be found because communication can be initiated by either a sender or a receiver. Thus the communication policy allows the receiver to

☆ The material in this paper was presented at the 46th IEEE Conference on Decision and Control (CDC), December 12–14, 2007, New Orleans, Louisiana, USA. This paper was recommended for publication in revised form by Associate Editor Jan Komenda under the direction of Editor Ian R. Petersen.

E-mail addresses: lricker@mta.ca (L. Ricker), benoit.caillaud@irisa.fr (B. Caillaud).

1 Tel.: +1 5063642541; fax: +1 5063642583.

0005-1098/$ – see front matter
doi:10.1016/j.automatica.2011.08.040

request information from a sender, which is not the strategy used in Ricker (1999), where only senders initiate communication.

The "communicate as early as possible" protocol requires the identification of behaviors that violate co-observability followed by calculating (w.r.t observable events) their *longest common prefix*. The rationale for using the longest common prefix as a means of pinpointing a useful communication location is straightforward. A sender in this protocol is a controller whose partial view of the behaviors involved in a violation of co-observability holds information that will aid the receiver in making the correct control decision (w.r.t this particular violation of co-observability). The reception of information about a behavior that both sender and receiver observe (or, conversely, events that neither observes) may not introduce enough new information to facilitate a correct control decision on the part of the receiver. We can find a communication location by disregarding all centrally-observable common prefixes as potential communication locations and choose the first observable event – not seen by the receiver – following the longest common prefix, as the earliest communication location.

There exist a number of additional studies on communication and decentralized supervisory control. The literature ranges from the introduction of novel information structures in which to study communication (Mannani & Gohari, 2008) to the examination of strategies for acyclic systems that reduce a given set of communications to one that is optimal (Wang, Lafortune, & Lin, 2008). More recently, Hiraishi (2009) has confirmed the computational complexity of the problem, but he, too, assumes that the set of communications is given as part of the problem formulation. In contrast, our work focuses on the construction of sets of communications that will allow decentralized communicating controllers to reach the control objective.

We propose a new computable strategy for identifying a communication language, where communication options lie between the first and last communication opportunities (inclusive). An earlier version of this work first appeared as Ricker and Caillaud (2007).

2. Background

2.1. Supervisory control

Supervisory control, as formulated by Ramadge and Wonham (1987), assumes that both the system requiring control, called the *plant*, and the desired behavior, often called the *legal* behavior, are described by formal languages over a given *alphabet*, denoted by Σ. The analysis in this paper is restricted to *regular* languages, a class of languages that can be described by finite automata. The goal of the control problem is to synthesize a *supervisor* of the plant such that, based on its observations of the plant behavior, the supervisor permits the occurrence of only the legal behavior (or possibly some subset of legal behavior) by issuing either an *enable* or *disable* command for different behaviors. The synthesis of the resulting control policy (determining which behaviors to enable or to disable) is further complicated by the fact that there are some behaviors that a supervisor cannot disable.

Decentralized supervisory control problems (Rudie & Wonham, 1992) involve the synthesis of n supervisors (for $n \geq 2$), each of which has only a partial view of the plant. That is, some of the behavior in the plant is *unobservable* to each supervisor. Let $I = \{1, \ldots, n\}$ be the set of n decentralized supervisors. The ability to synthesize a control policy relies on the existence of *at least one* supervisor that can make the correct control decision to keep the plant performing within the legal behavior, a property called *co-observability* (Rudie & Wonham, 1992). The remainder of this section will focus on the notation for synthesizing decentralized control strategies.

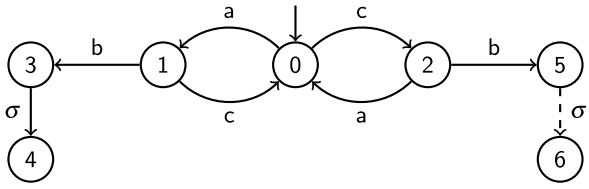

Fig. 1. A joint plant M_L and legal automaton M_K. The collection of all edges is M_L, whereas the collection of solid edges is M_K.

For any strings $s, t \in \Sigma^*$, where Σ^* is the set of finite words on an alphabet Σ, we say that t is a *prefix* of s, denoted $t \preceq s$, if $\exists w \in \Sigma^*$ such that $s = tw$. When $t \preceq s$, the left quotient of t with s is denoted as $t \setminus s = w$. If $L \subseteq \Sigma^*$, the *prefix-closure* of L is the language, denoted by $\bar{L}$, consisting of all prefixes of strings of L: $\bar{L} := \{t \in \Sigma^* \mid t \preceq s \in L\}$. Because every string is a prefix of itself, $L \subseteq \bar{L}$. A language is said to be *prefix-closed* if $L = \bar{L}$. As we are interested in all of the possible behaviors of the plant, we assume that the plant language is prefix-closed.

The partial view that a decentralized supervisor i has of the plant is described by the set of observable events, denoted by $\Sigma_{o,i} \subset \Sigma$, for $i \in I$. The set of all observable events is $\Sigma_o = \cup_{i \in I} \Sigma_{o,i}$. To describe a supervisor's view of the system behavior, we use the *natural projection* $P_i : \Sigma^* \to \Sigma_{o,i}^*$, for $i \in I$. This operator removes events σ from a string t that are not found in $\Sigma_{o,i}$: $P_i(\varepsilon) = \varepsilon$, where, throughout this paper, we use ε to denote the empty string; $P_i(\sigma) = \varepsilon$ if $\sigma \in \Sigma - \Sigma_{o,i}$, $P_i(\sigma) = \sigma$ if $\sigma \in \Sigma_{o,i}$, and $P_i(t\sigma) = P_i(t)P_i(\sigma)$, $t \in \Sigma^*$, $\sigma \in \Sigma$. Thus, if the plant generates sequence t, then $P_i(t)$ indicates the sequence of events observed by the decentralized supervisor i. The *inverse projection* of P_i is the mapping from $\Sigma_{o,i}^*$ to 2^{Σ^*}: $P_i^{-1}(t) = \{s \mid P_i(s) = t\}$.

To establish supervision on the plant, we partition the set of events Σ into disjoint sets Σ_c, *controllable* events, and Σ_{uc}, *uncontrollable* events. Controllable events are those events whose occurrence is preventable (i.e., may be disabled). Uncontrollable events are those events that cannot be prevented from occurring and are deemed permanently enabled. The set of controllable events is further divided into (not necessarily disjoint) subsets of events, which are controlled by supervisor i: $\Sigma_{c,i} \subseteq \Sigma_c$ (for $i \in I$), where $\Sigma_c = \cup_{i \in I} \Sigma_{c,i}$. We abuse our notation for I slightly and define the set of supervisors for which σ is controllable by $I_c(\sigma) = \{i \in I \mid \sigma \in \Sigma_{c,i}\}$. Finally, we denote the cardinality of any set S by $|S|$.

Formally, a deterministic automaton M is a tuple: $M = (Q, q_0, \Sigma, \delta, F)$, where Q is its set of states; $q_0 \in Q$ is the *initial state*; Σ is the finite alphabet; δ is a *transition function*, a partial function $\delta : \Sigma \times Q \to Q$; $F \subseteq Q$ is a set of final states. M is said to be finite whenever Q is finite. For $\sigma_0, \ldots, \sigma_{k-1} \in \Sigma$, a word $w = \sigma_0, \ldots, \sigma_{k-1} \in \Sigma^*$ is accepted by M iff there exists $q_1, \ldots, q_k$, where $q_k \in F$ such that $q_{j+1} = \delta(\sigma_j, q_j)$ for all $j = 0, \ldots, k-1$. The language generated by M, denoted by $\mathcal{L}(M)$, is the set of all words accepted by M. The automaton representing the plant is denoted by M_L, whereas the automaton representing the legal behavior is denoted by M_K. An example of a plant and associated legal behavior, where $I = \{1, 2\}$, is shown in Fig. 1. For this example, $\Sigma = \{a, b, c, \sigma\}$ and let $\Sigma_{o,1} = \{a, b, \sigma\}$, $\Sigma_{o,2} = \{b, c, \sigma\}$ and $\Sigma_{c,1} = \{a, b, \sigma\}$, $\Sigma_{c,2} = \{c\}$. Therefore $I(a) = I(b) = I_c(\sigma) = \{1\}$ and $I(c) = \{2\}$. Further, $Q = \{0, 1, 2, 3, 4, 5, 6\}$, while $F_L = Q$ and $F_K = Q - \{6\}$.

A prefix-closed language $K \subseteq L$ is *co-observable* with respect to L and P_i, for $i \in I$, if for all $t \in K$ and for all $\sigma \in \Sigma$,

$$(t\sigma \notin K) \quad \text{and} \quad (t\sigma \in L) \Rightarrow \exists i \in I_c(\sigma) \text{ such that}$$

$$P_i^{-1}[P_i(t)]\sigma \cap K = \emptyset. \tag{1}$$

It must be the case that based only on its partial view of a sequence, a decentralized supervisor can make the correct control decision.

The equivalent condition in the case of a single (centralized) supervisor is called *observability*. The observations of a centralized supervisor are captured by a projection operator $P : \Sigma^* \to \Sigma_o^*$. A prefix-closed language K is said to be *observable* with respect to L, P, and Σ_c if for all $t \in K$ and for all $\sigma \in \Sigma_c$, $(t\sigma \notin K)$ and $(t\sigma \in L) \Rightarrow P^{-1}[P(t)]\sigma \cap K = \emptyset$. For purposes of designing a decentralized communication protocol, K must be observable but must not be co-observable. In the absence of a centralized control solution, no amount of communication will allow decentralized supervisors to meet the control objective.

In Fig. 1, let $K = \mathcal{L}(M_K)$ and $L = \mathcal{L}(M_L)$. It is the case that K is observable, since all illegal sequences can be distinguished from all legal sequences. But K is *not* co-observable since it is possible to find sequences, such as $t = \mathsf{accacb}$, where $t\sigma \in L - K$, and for all $i \in I_c(\sigma) = \{1\}$ it is the case that $\mathsf{caab}\ \sigma \in P_1^{-1}[P_1(t)]\sigma \cap K \neq \emptyset$.

2.2. Labels

We want a means of characterizing the synchronization between event occurrences in the plant and the observation of these event occurrences by decentralized supervisors. To facilitate this, we make use of *synchronization vectors* (Arnold, 1994), which we refer to here as *labels*. For purposes of adapting labels for use in decentralized supervisory control problems, we begin by introducing an augmented set of supervisors $I_0 = \{0\} \cup I$, where 0 represents an agent that tracks all plant behavior. We also define an augmented alphabet $\Sigma^\varepsilon = \{\varepsilon\} \cup \Sigma$.

Definition 1. A *label* $\ell : I_0 \longrightarrow \Sigma^\varepsilon$ is a mapping from each supervisor to either an event from Σ or ε:$\ell = \langle \ell(0), \ell(1), \ldots, \ell(i), \ldots, \ell(n)\rangle$. The empty label is $\langle \varepsilon, \ldots, \varepsilon\rangle$, i.e., for all $i \in I_0$, $\ell(i) = \varepsilon$. The support of a label, denoted by $\|\ell\|$, is $\{i \mid \ell(i) \neq \varepsilon\}$.

We will be interested in several properties of labels.

Definition 2. Two labels ℓ_1, ℓ_2 are *compatible*, denoted as $\ell_1 \uparrow \ell_2$, iff $\forall i \in I_0, \ell_1(i) = \varepsilon$ or $\ell_2(i) = \varepsilon$ or $\ell_1(i) = \ell_2(i)$.

Suppose that $\Sigma^\varepsilon = \{\mathsf{a}, \mathsf{b}, \varepsilon\}$, $I_0 = \{0, 1, 2, 3\}$ and we have three labels such that $\ell_1 = \langle \mathsf{a}, \mathsf{a}, \varepsilon, \mathsf{a}\rangle$, $\ell_2 = \langle \varepsilon, \varepsilon, \mathsf{b}, \varepsilon\rangle$, and $\ell_3 = \langle \mathsf{b}, \mathsf{b}, \varepsilon, \mathsf{b}\rangle$. Then $\ell_1 \uparrow \ell_2$ and $\ell_2 \uparrow \ell_3$ but ℓ_1 is not compatible with ℓ_3, denoted as $\ell_1 \not\uparrow \ell_3$, because for $i = 0, 1, 3$ it is the case that $\ell_1(i) \neq \varepsilon, \ell_3(i) \neq \varepsilon$ and $\ell_1(i) \neq \ell_3(i)$.

Definition 3. Two labels, ℓ_1 and ℓ_2 are *independent*, denoted as $\ell_1|\ell_2$, iff $\forall i \in I_0\ \ell_1(i) = \varepsilon$ or $\ell_2(i) = \varepsilon$.

If we use the labels defined above, then $\ell_1|\ell_2$ and $\ell_2|\ell_3$ but ℓ_1 and ℓ_3 are not independent, denoted as $\ell_1 \not| \ell_3$, because for $i = 0, 1, 3$ neither $\ell_1(i)$ nor $\ell_3(i)$ is equal to ε. But if we also have $\ell_4 = \langle \varepsilon, \mathsf{b}, \mathsf{a}, \varepsilon\rangle$ and $\ell_5 = \langle \varepsilon, \mathsf{b}, \varepsilon, \varepsilon\rangle$, then $\ell_4 \uparrow \ell_5$ but $\ell_4 \not| \ell_5$.

It is possible to define a partial order relation on labels: $\ell_1 \leq \ell_2$ iff $\forall i \in I_0, \ell_1(i) \neq \varepsilon \Rightarrow \ell_1(i) = \ell_2(i)$. It is straightforward to check that $\leq$ is a partial order relation:

- transitive: $\ell_1 \leq \ell_2 \leq \ell_3 \Rightarrow \ell_1 \leq \ell_3$;
- reflexive: $\ell_1 \leq \ell_1$;
- antisymmetric: $\ell_1 \leq \ell_2 \leq \ell_1 \Rightarrow \ell_1 = \ell_2$.

This partial order relation subsumes a greatest lower bound $\wedge$ and a partial least upper bound $\vee$. The greatest lower bound is computed as follows:

$$\forall i \in I_0, \quad (\ell_1 \wedge \ell_2)(i) = \begin{cases} \ell_1(i), & \text{if } \ell_1(i) = \ell_2(i); \\ \varepsilon, & \text{otherwise.} \end{cases}$$

Let $\ell_1 = \langle \mathsf{a}, \mathsf{a}, \mathsf{a}, \varepsilon\rangle$ and $\ell_2 = \langle \varepsilon, \mathsf{b}, \mathsf{a}, \varepsilon\rangle$. Then $\ell_1 \wedge \ell_2 = \langle \varepsilon, \varepsilon, \mathsf{a}, \varepsilon\rangle$.

The least upper bound of two compatible labels, denoted by $\ell_1 \vee \ell_2$, is computed as follows:

$$\forall i \in I_0, \quad (\ell_1 \vee \ell_2)(i) = \begin{cases} \ell_1(i), & \text{if } \ell_1(i) \neq \varepsilon; \\ \ell_2(i), & \text{if } \ell_2(i) \neq \varepsilon; \\ \varepsilon, & \text{otherwise.} \end{cases}$$

Suppose that $\ell_1 = \langle \mathsf{b}, \varepsilon, \varepsilon, \mathsf{b}\rangle$ and $\ell_2 = \langle \varepsilon, \mathsf{a}, \varepsilon, \mathsf{b}\rangle$. Since $\ell_1 \uparrow \ell_2$, then $\ell_1 \vee \ell_2$ is defined and is equal to $\langle \mathsf{b}, \mathsf{a}, \varepsilon, \mathsf{b}\rangle$.

Although this algebra on labels is not a Boolean algebra, a residual operation of the least upper bound can be defined. Let $\ell_1 \leq \ell_2$, then $\ell_2 \setminus \ell_1$ is the least label such that $\ell_1 \vee (\ell_2 \setminus \ell_1) = \ell_2$. The computation of $\ell_2 \setminus \ell_1$ is straightforward:

$$\forall i \in I_0, \quad \ell_2 \setminus \ell_1(i) = \begin{cases} \ell_2(i), & \text{if } \ell_1(i) = \varepsilon; \\ \varepsilon, & \text{otherwise.} \end{cases}$$

Suppose that $\ell_1 = \langle \varepsilon, \mathsf{a}, \mathsf{a}, \varepsilon\rangle$ and $\ell_2 = \langle \varepsilon, \mathsf{a}, \mathsf{a}, \mathsf{a}\rangle$. Then $\ell_2 \setminus \ell_1 = \langle \varepsilon, \varepsilon, \varepsilon, \mathsf{a}\rangle$.

In Section 4.1, we define a set of labels for the decentralized control and communication problem.

2.3. Transducers

A rational transducer is a tuple $\mathcal{J} = (Q, q_0, \Sigma_1, \ldots, \Sigma_m, E, F)$, where Q is the state set; $q_0 \in Q$ is the initial state; Σ_i, for $i = 1, \ldots, m$, are alphabets; $E \subseteq Q \times \Sigma_1^* \times \cdots \times \Sigma_m^* \times Q$ is the transition relation; $F \subseteq Q$ is a set of final states, which will be of use in the sequel. Note that Q need not be finite, nor does E need to be deterministic. A sequence of transitions from state q_1 to state q_k' (i.e., $(q_1, \sigma_{11}, \ldots, \sigma_{1m}, q_1') \ldots (q_k, \sigma_{k1}, \ldots, \sigma_{km}, q_k')$) is a *path* if and only if $q_j' = q_{j+1}$, for $j = 1, \ldots, k-1$. We denote such a path by $q_1 \overset{s_1,\ldots,s_m}{\twoheadrightarrow} q_k'$, where $s_i = \sigma_{i1} \ldots \sigma_{ik}$, for $i = 1, \ldots, m$.

Two particular cases of interest are (i) one-dimensional transducers, which correspond to automata, and (ii) two-dimensional transducers, which define functions or binary relations over words. In the case of transducers of dimension two, Σ_1 is referred to as the input alphabet and Σ_2 is the output alphabet (typically denoted by Γ). It is customary to denote transition labels, consisting of an input letter $\sigma \in \Sigma_1$ and an output letter $\gamma \in \Gamma$, as σ/γ.

3. Decentralized control and communication

In the standard formulation of a decentralized discrete-event control problem, when co-observability is satisfied, supervisors make control decisions based only on their partial view of plant behavior. For each control decision that must be taken, there is at least one supervisor that has enough information from its own observations to make the correct control decision. When co-observability is violated, there is some set of behaviors for which none of the relevant supervisors can make the correct control decision. That is, there is some legal behavior u, followed by a controllable event σ which exits the legal behavior and *all* supervisors that control σ cannot distinguish u from some other behavior (or set of behaviors), say v, that remains within the legal behavior when followed by σ. For the remainder of the paper, we use the notation $u\sigma$ to refer to a representative illegal sequence, $v\sigma$ as a legal sequence, and assume that for all $i \in I_c(\sigma), P_i(u) = P_i(v)$.

When supervisors cannot distinguish behaviors and make the correct control decision, then we can introduce communication into the problem formulation and identify *where* additional information could lead to correct control solutions. When a system satisfies observability, then there must be at least one observable event that allows a centralized supervisor to distinguish $u\sigma$ from $v\sigma$ (i.e., $P(u) \neq P(v)$). A sequence that leads to this location will be included in the *communication language*. After the occurrence of such a sequence, the new information introduced by the sender

Table 1
Potential sequences in the communication language for supervisor 2 when u = accacaaccb and v = accaacab.

$u' \preceq u$	$P(u') \sqcap P(v)$	$(P(u') \sqcap P(v)) \diagdown P(u')$
1. accacaaccb	acca	caaccb
2. accacaacc	acca	caacc
3. accacaac	acca	caac
4. accacaa	acca	caa
5. accaca	acca	ca
6. accac	acca	c
7. acca	acca	ε
8. acc	acc	ε
9. ac	ac	ε
10. a	a	ε
11. ε	ε	ε

allows a formerly-confused supervisor (the receiver) to ultimately make the correct control decision about σ.

To determine potential candidates for sequences along u to be included in the communication language, we identify all $u' \preceq u$ that have a non-empty left quotient with the longest common prefix of u' and v, denoted here by $P(u') \sqcap P(v)$. (Equivalently, to find potential sequences along v for the communication language, we examine prefixes of v w.r.t u.)

Useful information is only imparted to the receiver when the sender communicates information about system behavior that the receiver does not know about with certainty. In this case, we insist that a sequence in the communication language for the sender ends with an event that is observed by the sender but not by the receiver. For a sequence $s \in \Sigma^*$ and $\sigma_1, \ldots, \sigma_k \in \Sigma$ such that $s = \sigma_1 \ldots \sigma_k$, define $\eta : \Sigma^* \to \Sigma$ such that $\eta(s) = \sigma_k$.

Definition 4. Given $u, v \in L$ and $\sigma \in \Sigma_c$ such that for all $i \in I_c(\sigma)$ $P_i(u) = P_i(v)$ and $u\sigma \in L - K$ while $v\sigma \in K$. The communication language for sender i w.r.t receiver $j \in I$ along $u\sigma$ is defined as follows:

$$C_{i,j}(u, v) = \{u' \preceq u \mid (P(u') \sqcap P(v)) \diagdown P(u') \neq \varepsilon \text{ and } \eta(u') \in \Sigma_{o,i} - \Sigma_{o,j}\}.$$

We can analogously describe the communication language along $v\sigma$, denoted by $C_{i,j}(v, u)$.

In Table 1, we consider the potential sequences along u for the communication language when u = accacaaccb and v = accaacab, and we use the controllable and observable event sets from the example in Fig. 1. Thus we are seeking a communication language for supervisor 2, as it is supervisor 1 for which u and v represent a violation of co-observability w.r.t σ.

After discarding sequences u' because they either end with an inappropriate event (e.g., sequences 1, 4, 5) or have an empty left quotient with the longest common prefix w.r.t v (e.g., sequences 7–11), the communication language for supervisor 2 along u is $C_{2,1}(u, v)$ = {accacaacc, accacaac, accac}. The communication language for supervisor 2 along v can also be calculated: $C_{2,1}(v, u)$ = {accaac}.

The final output from the procedures presented in this paper is the communication language for each violation of co-observability w.r.t σ in the plant language. Further refinement may be required, if incorporating all communication opportunities results in too much redundancy (e.g., only one element of $C_{i,j}(u, v) \cup C_{i,j}(v, u)$ may be necessary to resolve the violation of co-observability). The next section provides construction details for the structures we use to generate the communication language for all violations of co-observability w.r.t σ in the system.

4. The construction of a communication language for decentralized control

We require three new structures to construct our communication language and present their construction in detail below.

(i) A finite structure U_σ, such that $\mathcal{L}(U_\sigma)$ is a language on the alphabet of labels, that encodes all sequences $u\sigma$ and $v\sigma$ that violate co-observability w.r.t σ.
(ii) A finite structure $\mathcal{J}_{\mathcal{A}}$ that, given a sequence s as input, will output prefixes s' of s, i.e., $s' \preceq s$, such that s' ends with an event observable to the sender and *not* observable to the receiver (who makes the control decision about σ).
(iii) A structure $\mathcal{J}_{\mathcal{B}}$ that, given two sequences s and t as input, will accept s iff the left quotient of $P(s)$ with the longest common observable prefix (i.e., longest centralized observation) of s and t is non-empty: $(P(s) \sqcap P(t)) \diagdown P(s) \neq \varepsilon$.

We also use an existing structure, a single-state transducer $\mathcal{P}$ that implements the projection operator P. Because we need to calculate projections for two different sequences in parallel, we make a copy of $\mathcal{P}$ and denote it by $\mathcal{P}'$.

4.1. A finite automaton to detect violations of co-observability

The automaton introduced here, U_σ, encodes all legal and illegal sequences that end in event σ and that violate (1), i.e., co-observability. This is not to be confused with the nondeterministic automaton described in Rudie and Willems (1995), which generates the complete sublanguage of L that is not co-observable. In particular, the non-deterministic structure in Rudie and Willems (1995), when defined for 2 supervisors, is used to simultaneously track sequences in $P_1^{-1}(P_1(\mathcal{L}(M_K)))$, $P_2^{-1}(P_2(\mathcal{L}(M_K)))$, $\mathcal{L}(M_K)$ and $\mathcal{L}(M_L)$. In the language of the automaton we define below, when defined for 2 supervisors, sequences in $\mathcal{L}(M_{L-K})$ and one copy of $\mathcal{L}(M_K)$ for each supervisor are tracked simultaneously.

The alphabet for U_σ is constructed from the labels introduced in Section 2.2. Labels are generated from a given finite set of *atoms*, denoted by A. The set of atoms is defined as the union of the following two label sets.

The first atom $a^{uo}_{i,e} : I_0 \to (\Sigma - \Sigma_{o,i}) \cup \{\varepsilon\}$ represents transition labels for events unobservable to supervisor i, for $i \in I_0$. This corresponds to the idea that supervisor i guesses about the occurrence of event e in the plant that it does not directly observe. Formally, for $i, j \in I_0$ and $e \in \Sigma - \Sigma_{o,i}$, where for $i = 0$, we interpret $\Sigma_{o,i}$ to be Σ_o: $a^{uo}_{i,e}(j) = e$ when $i = j$, otherwise it is equal to ε. For example, in Fig. 1, since c $\in \Sigma - \Sigma_{o,1}$, we have $a^{uo}_{1,\mathrm{c}} = \langle \varepsilon, \mathrm{c}, \varepsilon \rangle$, meaning that supervisor 1 has no idea if c occurred in the plant ($a^{uo}_{1,\mathrm{c}}(0) = \varepsilon$), but it guesses that c could have occurred ($a^{uo}_{1,\mathrm{c}}(1) = \mathrm{c}$) and it makes no assumptions about the observations of the other supervisors ($a^{uo}_{1,\mathrm{c}}(2) = \varepsilon$).

The second atom $a^o_e : I_0 \to \Sigma_o \cup \{\varepsilon\}$ corresponds to the occurrence of event e in the plant and the synchronized observation of e by all supervisors i for which $e \in \Sigma_{o,i}$. Formally, $a^o_e(i) = e$ if $i = 0$ or if $i \neq 0$ and $e \in \Sigma_{o,i}$; otherwise it is ε. For example, in Fig. 1, since c $\in \Sigma_{o,2}$, we have $a^o_{\mathrm{c}} = \langle \mathrm{c}, \varepsilon, \mathrm{c} \rangle$, meaning that event c occurred in the plant ($a^o_{\mathrm{c}}(0) = \mathrm{c}$), supervisor 1 did not observe the occurrence of c ($a^o_{\mathrm{c}}(1) = \varepsilon$,) and supervisor 2 observed the occurrence of c ($a^o_{\mathrm{c}}(2) = \mathrm{c}$).

The set of atoms constructed from the events in the alphabet for the example shown in Fig. 1: $A = \{\langle \varepsilon, \mathrm{c}, \varepsilon \rangle, \langle \varepsilon, \varepsilon, \mathrm{a} \rangle, \langle \mathrm{a}, \mathrm{a}, \varepsilon \rangle, \langle \mathrm{b}, \mathrm{b}, \mathrm{b} \rangle, \langle \sigma, \sigma, \sigma \rangle, \langle \mathrm{c}, \varepsilon, \mathrm{c} \rangle\}$.

Let $A^\varepsilon = \langle \varepsilon, \ldots, \varepsilon \rangle \cup A$. The alphabet of U_σ, denoted by $\mathbb{A}$, is the least set such that $A^\varepsilon \subseteq \mathbb{A}$ and for all $a, a' \in \mathbb{A}$, $a \uparrow a' \Rightarrow a \vee a' \in \mathbb{A}$. For the set of atoms noted above: $\mathbb{A} = A^\varepsilon \cup \{\langle \mathrm{c}, \mathrm{c}, \mathrm{c} \rangle, \langle \varepsilon, \mathrm{c}, \mathrm{a} \rangle, \langle \mathrm{a}, \mathrm{a}, \mathrm{a} \rangle\}$.

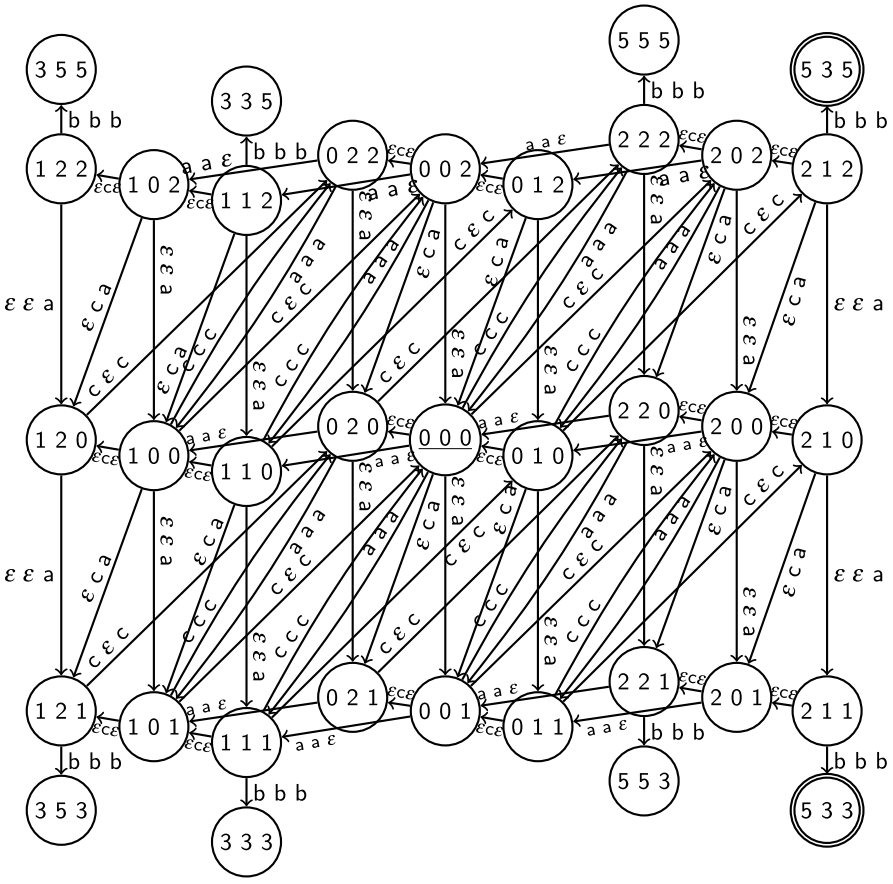

Fig. 2. Automaton U_σ for M_L and M_K in Fig. 1. The initial state is identified by its underlined state name.

To construct the remaining components of U_σ, we begin with our prefix-closed regular languages L and K, such that $K \subseteq L$. It can be assumed, w.l.o.g. that M_L and M_K have identical transition relations and state structures.[2] Hence M_L and M_K may differ only in their final state sets: $M_L = (Q, q_0, \Sigma, \delta, F_L)$, $M_K = (Q, q_0, \Sigma, \delta, F_K)$. We define a third language (and its corresponding automaton representation) that accepts only illegal sequences: $B = L - K$ and $M_B = (Q, q_0, \Sigma, \delta, F_B)$, where $F_B = F_L - F_K$.

Our goal is to construct an automaton whose marked sequences encode those sequences that violate co-observability w.r.t event $\sigma \in \Sigma_c$. So we must further refine B and K. Let $B_\sigma = \{s \in K \mid s\sigma \in B\}$ be the language of sequences that leave K via the event σ. The corresponding automaton is $M_{B_\sigma} = (Q, q_0, \Sigma, \delta, F_{B_\sigma})$, where $F_{B_\sigma} = \{q \in F_K \mid \delta(q, \sigma) \in F_B\}$. Let $K_\sigma = \{s \in K \mid s\sigma \in K\}$ be the language of sequences in K that remain in K after the occurrence of the event σ. The corresponding automaton is $M_{K_\sigma} = (Q, q_0, \Sigma, \delta, F_{K_\sigma})$, where $F_{K_\sigma} = \{q \in F_K \mid \delta(q, \sigma) \in F_K\}$.

To facilitate the synchronization required for the partially-observed events in K, we augment M_{B_σ}, M_{K_σ} and M_K with self-loops labeled ε at each state (e.g., for all $q \in Q$), thereby updating the alphabets of each automaton from Σ to Σ^ε.

We build U_σ to identify pairs of sequences that violate co-observability, by composing M_{B_σ} with $|I_c(\sigma)|$ copies of the M_{K_σ} automaton and $n - |I_c(\sigma)|$ copies of M_K. The state set of U_σ is simply the reachable part of the Cartesian product of the states of the component automata. The composition can be defined in two steps: (1) the Cartesian product of the transition relations; and (2) the restriction of the resulting transition relation to those transitions with labels in $\mathbb{A}$.

Now we can construct automaton U_σ as follows: $U_\sigma = (X, (q_0)^{n+1}, \mathbb{A}, \delta^{U_\sigma}, F_{U_\sigma})$, where $X \subseteq (Q)^{n+1}$; $F_{U_\sigma} \subseteq F_{B_\sigma} \times (F_{K_\sigma})^{|I_c(\sigma)|} \times (F_K)^{n-|I_c(\sigma)|}$ is the set of reachable final states, and the transition function is defined according to: $\delta^{U_\sigma}(x, \ell) = x'$ iff $\ell \in \mathbb{A}$ and $\forall i \in I_0, \delta(x(i), \ell(i)) = x'(i)$. Every marked sequence in $\mathcal{L}(U_\sigma)$ (i.e., one leading to a state in F_{U_σ}) can be decomposed into a sequence in $L \setminus K$ and $|I_c(\sigma)|$ sequences in K, thereby indicating a violation of co-observability. Note that the state structure of U_σ and its transition function do not depend on the value of σ. Thus, we need only calculate U_σ once and adjust the marking w.r.t the various values for σ that lead to other violations of co-observability.

For L and K depicted in Fig. 1, we assume identical state structures and transition relations for M_{B_σ}, M_{K_σ} and M_K, namely $Q = \{0, 1, 2, 3, 4, 5, 6\}$, and δ includes selfloops of ε at each state in Q; however, the final state sets are $F_{B_\sigma} = \{5\}$, $F_{K_\sigma} = \{3\}$ and $F_K = Q - \{6\}$. The resulting U_σ, shown in Fig. 2, has 28 states (2 of which are marked) and 84 transitions.

It will be useful to refer to sequences of label components. Let $w = \ell_0 \ldots \ell_k \in \mathcal{L}(U_\sigma)$. Then $w(i) = \ell_0(i) \ldots \ell_k(i)$ is the sequence formed by the ith components of the labels in w, where $i \in I_0$. Suppose that $w = \langle \mathsf{a}, \mathsf{a}, \varepsilon \rangle \langle \varepsilon, \varepsilon, \mathsf{a} \rangle \langle \mathsf{c}, \varepsilon, \mathsf{c} \rangle \langle \mathsf{c}, \varepsilon, \mathsf{c} \rangle \langle \varepsilon, \varepsilon, \mathsf{a} \rangle \langle \varepsilon, \varepsilon, \mathsf{a} \rangle \langle \mathsf{b}, \mathsf{b}, \mathsf{b} \rangle$. Then $w(0) = \mathsf{a}\varepsilon\mathsf{cc}\varepsilon\varepsilon\mathsf{b}$, $w(1) = \mathsf{a}\varepsilon\varepsilon\varepsilon\varepsilon\varepsilon\mathsf{b}$ and $w(2) = \varepsilon\mathsf{accaab}$.

4.2. A non-deterministic finite transducer to find particular prefixes of a sequence

Given a sequence s, we would like to be able to find all of its prefixes that end with an event possessing particular

[2] If M_L and M_K do not share the same state structure, we can transform them by taking the Cartesian product of their respective state sets.

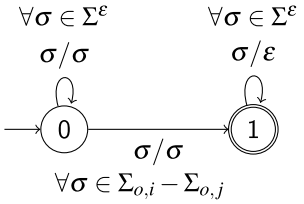

Fig. 3. Transducer $\mathcal{J}_{\mathcal{A}}$ outputs specific prefixes of an input sequence.

characteristics. In the context of generating sequences for the communication language, we want s to end with an event observable to the sender, supervisor i, but not observable to the receiver, supervisor j. We use a non-deterministic finite transducer, $\mathcal{J}_{\mathcal{A}} = (\{0, 1\}, 0, \Sigma^{\varepsilon}, \Sigma^{\varepsilon}, E, \{1\})$, which is shown in Fig. 3.

For purposes of building communication protocols, the input to $\mathcal{J}_{\mathcal{A}}$ will be $u' \preceq u$ (resp. $v' \preceq v$) where u (resp. v) is derived from marked sequences $w \in \mathcal{L}(U_\sigma)$: we consider $u = w(0)$ and $v = w(j)$, choosing one of the $j \in I_c(\sigma)$ for which v forms a violation of co-observability with u. Recall that by construction, a sequence is marked in $\mathcal{L}(U_\sigma)$ iff $w(0)\sigma \in \mathcal{L}(M_B)$ and $\forall j \in I_c(\sigma)$, $w(j)\sigma \in K$ violate co-observability.

Let w be a marked sequence in $\mathcal{L}(U_\sigma)$ from Fig. 2: $w = \langle \mathsf{a}, \mathsf{a}, \varepsilon\rangle\langle \mathsf{c}, \varepsilon, \mathsf{c}\rangle\langle \varepsilon, \varepsilon, \mathsf{a}\rangle\langle \mathsf{c}, \varepsilon, \mathsf{c}\rangle\langle \varepsilon, \varepsilon, \mathsf{a}\rangle\langle \varepsilon, \varepsilon, \mathsf{a}\rangle\langle \mathsf{b}, \mathsf{b}, \mathsf{b}\rangle$. Then we have $u = w(0) = \mathsf{ac}\varepsilon\mathsf{c}\varepsilon\varepsilon\mathsf{b}$ and $v = w(1) = \mathsf{a}\varepsilon\varepsilon\varepsilon\varepsilon\varepsilon\mathsf{b}$. We input u to $\mathcal{J}_{\mathcal{A}}$ for $i = 2$ and $j = 1$. The valid outputs are acc and ac. When the input is v, then the only valid output is ε.

4.3. An infinite transducer to find the left quotient of a sequence s with the longest common prefix of s and t

The next step in generating sequences for the communication language involves a transducer that accepts those sequences s such that $(P(s) \sqcap P(t)) \diagdown P(s) \neq \varepsilon$. To accomplish this, we construct an infinite deterministic transducer, which takes a pair of sequences as input and outputs their longest common prefix w.r.t P.

We form a transducer $\mathcal{J}_{\mathcal{B}}$ with an input alphabet of $\Sigma^{\varepsilon} \times \Sigma^{\varepsilon}$, an output alphabet $\Gamma = \emptyset$, a state set $Q = \{\{Left, Right\} \times \Sigma^{+}\} \cup \{\top, \bot\}$, where the initial state is $\top$ and the final state is $\bot$. The set of transition rules is defined in Table 2.

In the context of generating sequences along u that could be included in the communication language, we consider input pairs as comprising (i) a valid prefix of $u = w(0)$ (as determined by $\mathcal{J}_{\mathcal{A}}$) and (ii) $v = w(j)$, for $w \in \mathcal{L}(U_\sigma)$ and $j \in I_c(\sigma)$. Equivalently, for sequences along v, we consider pairs comprised of a valid prefix of v and u.

Let us examine sequence 6 from Table 1, which can be derived from the following marked sequence in $\mathcal{L}(U_\sigma)$ in Fig. 2: $w = \langle \mathsf{a}, \mathsf{a}, \varepsilon\rangle\ \langle \mathsf{c}, \varepsilon, \mathsf{c}\rangle\langle \varepsilon, \mathsf{c}, \varepsilon\rangle\ \langle \varepsilon, \varepsilon, \mathsf{a}\rangle\langle \mathsf{c}, \varepsilon, \mathsf{c}\rangle\langle \varepsilon, \mathsf{c}, \varepsilon\rangle\langle \mathsf{a}, \mathsf{a}, \varepsilon\rangle\langle \varepsilon, \varepsilon, \mathsf{a}\rangle\langle \mathsf{c}, \varepsilon, \mathsf{c}\rangle\langle \mathsf{a}, \mathsf{a}, \varepsilon\rangle\langle \varepsilon, \mathsf{c}, \varepsilon\rangle\langle \mathsf{a}, \mathsf{a}, \varepsilon\rangle\ \langle \varepsilon, \varepsilon, \mathsf{a}\rangle\ \langle \mathsf{c}, \varepsilon, \mathsf{c}\rangle\ \langle \varepsilon, \varepsilon, \mathsf{a}\rangle\langle \mathsf{c}, \varepsilon, \mathsf{c}\rangle\langle \mathsf{b}, \mathsf{b}, \mathsf{b}\rangle$. The first input sequence for $\mathcal{J}_{\mathcal{B}}$ is the prefix of $w(0)$ that corresponds to sequence 6: $\mathsf{ac}\varepsilon\varepsilon\mathsf{c}\varepsilon\mathsf{a}\varepsilon\mathsf{c}$. The second input sequence is $w(1)$. Note that it is necessary to pad the end of the shorter of the two input sequences with εs to ensure that they are of the same length. In this example, the first input sequence has eight additional εs added to its end. Then we have the following path in $\mathcal{J}_{\mathcal{B}}$ that accepts the first input sequence by applying, in order, rules 3, 1, 6, 18, 1, 6, 3, 18, 1, 10: $\top \xrightarrow{(\mathsf{a},\mathsf{a})} \top \xrightarrow{(\mathsf{c},\varepsilon)} (Left, \mathsf{c}) \xrightarrow{(\varepsilon,\mathsf{c})} \top \xrightarrow{(\varepsilon,\varepsilon)} \top \xrightarrow{(\mathsf{c},\varepsilon)} (Left, \mathsf{c}) \xrightarrow{(\varepsilon,\mathsf{c})} \top \xrightarrow{(\mathsf{a},\mathsf{a})} \top \xrightarrow{(\varepsilon,\varepsilon)} \top \xrightarrow{(\mathsf{c},\varepsilon)} (Left, \mathsf{c}) \xrightarrow{(\varepsilon,\mathsf{a})} \bot$. Once the final state is reached, rule 5 is repeatedly applied until the end of the inputs are processed. Thus, the accepted sequence is in the communication language.

Table 2
The set of transition rules for transducer $\mathcal{J}_{\mathcal{B}}$ where $e, f, g \in \Sigma$, $\Sigma^{+} = \Sigma^{*} - \{\varepsilon\}$ and $q \in Q$.

1. $\top \xrightarrow{(e,\varepsilon)} (Left, e)$	
2. $\top \xrightarrow{(\varepsilon,e)} (Right, e)$	
3. $\top \xrightarrow{(e,e)} \top$	
4. $\top \xrightarrow{(e,f)} \bot$,	$(e \neq f)$
5. $\bot \xrightarrow{(x,y)} \bot, x, y \in \Sigma^{\varepsilon}$	
6. $(Left, f) \xrightarrow{(\varepsilon,f)} \top$	
7. $(Left, z) \xrightarrow{(e,\varepsilon)} (Left, ze)$	$z \in \Sigma^{+}$
8. $(Left, fz) \xrightarrow{(e,f)} (Left, ze)$	$z \in \Sigma^{*}$
9. $(Left, fz) \xrightarrow{(\varepsilon,f)} (Left, z)$	$z \in \Sigma^{+}$
10. $(Left, fz) \xrightarrow{(\varepsilon,g)} \bot$	$(f \neq g), z \in \Sigma^{*}$
11. $(Left, fz) \xrightarrow{(e,g)} \bot$	$(f \neq g), z \in \Sigma^{*}$
12. $(Right, e) \xrightarrow{(e,\varepsilon)} \top$	
13. $(Right, z) \xrightarrow{(\varepsilon,f)} (Right, zf)$	$z \in \Sigma^{+}$
14. $(Right, ez) \xrightarrow{(e,f)} (Right, zf)$	$z \in \Sigma^{*}$
15. $(Right, ez) \xrightarrow{(e,\varepsilon)} (Right, z)$	$z \in \Sigma^{+}$
16. $(Right, ez) \xrightarrow{(g,\varepsilon)} \bot$	$(e \neq g), z \in \Sigma^{*}$
17. $(Right, ez) \xrightarrow{(g,f)} \bot$	$(e \neq g), z \in \Sigma^{*}$
18. $q \xrightarrow{(\varepsilon,\varepsilon)} q$	

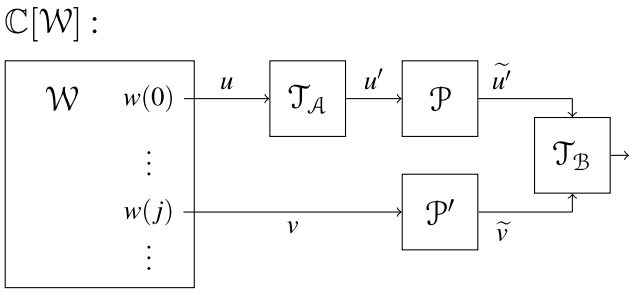

Fig. 4. Building a communication protocol w.r.t u. We denote by $\mathbb{C}[\mathcal{W}]$ a synchronized product of all the components displayed in the diagram, where $\mathcal{W}$ is a place holder that corresponds to an automaton that generates a language over an alphabet of labels.

5. Mind the gap

5.1. An infinite mind-the-gap automaton

A graphical interpretation of the interaction of the three components we define for building a communication protocol w.r.t $u \in L{-}K$ is provided in Fig. 4. The system we work with, denoted by $\mathbb{C}[\mathcal{W}]$, corresponds to the synchronized product of the components $\mathcal{W}$, $\mathcal{J}_{\mathcal{A}}$, $\mathcal{P}$, $\mathcal{P}'$ and $\mathcal{J}_{\mathcal{B}}$ in the following way. We take the Cartesian product of the transition relation restricted to transitions such that the output $w(0)$ of $\mathcal{W}$ is equal to the input of $\mathcal{J}_{\mathcal{A}}$, the output of $\mathcal{J}_{\mathcal{A}}$ is equal to the input to $\mathcal{P}$, the output $w(j)$ of $\mathcal{W}$ is equal to the input to $\mathcal{P}'$, and the output from $\mathcal{P}$ is equal to the first input to $\mathcal{J}_{\mathcal{B}}$, whereas the output from $\mathcal{P}'$ is equal to the second input to $\mathcal{J}_{\mathcal{B}}$. Note that $\mathcal{W}$ is a place holder for any automaton that generates a language over the alphabet of labels. For our infinite mind-the-gap automaton, we use U_σ for $\mathcal{W}$.

By construction, marked sequences in the language of U_σ represent violations of co-observability. Let w be a marked sequence in $\mathcal{L}(U_\sigma)$. Thus $w(0)\sigma \in L - K$ and for all $j \in I_c(\sigma)$, $w(j)\sigma \in K$ and $P_j(w(0)) = P_j(w(j))$. We have $u = w(0)$ and $v = w(j)$. It is supervisor i that will be the sender of information about this particular violation of co-observability. We then want to identify relevant prefixes of u (i.e., ones that end in events that are observable to the sender and not to the receiver). Recall that

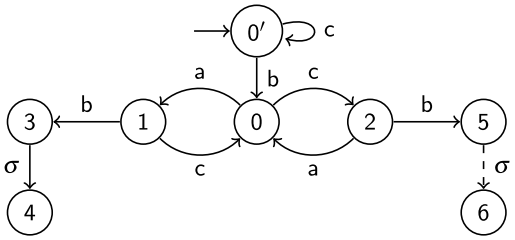

Fig. 5. An automaton that when transformed into U_σ generates sequences that produce an infinite number of states in $\mathcal{J}_{\mathcal{B}}$.

the transducers $\mathcal{P}$ and $\mathcal{P}'$ implement the projection operator P, where we denote the result of $P(t)$ by $\tilde{t}$. Finally, a prefix u' is in the communication language if the left quotient of $P(u')$ with $P(u') \sqcap P(v)$ is non-empty, as calculated by $\mathcal{J}_{\mathcal{B}}$. A similar diagram can be constructed for a communication language w.r.t $v \in K$. The complete communication language (for supervisor 2) for the example in Fig. 1 consists of the sequences $w(2)$ that reach any of the eight states in U_σ with an outgoing transition of $\langle \mathsf{c}, \mathsf{c}, \mathsf{c} \rangle$.

The difficulty with the output from U_σ is that the input to transducer $\mathcal{J}_{\mathcal{B}}$ is not synchronized. That is, when $\mathcal{J}_{\mathcal{B}}$ is processing pairs of letters, $\mathcal{J}_{\mathcal{B}}$ has to perform the matching of the two sequences. It is possible that $\mathcal{J}_{\mathcal{B}}$ will require unbounded memory to store all the observations seen prior to the matching of two events. For example, suppose that we expand M_L (and M_K) in Fig. 1 so that we introduce a new initial state $0'$ and add two new transitions $\delta(0', c) = 0'$ and $\delta(0', b) = 0$, as shown in Fig. 5. Let the two sequences for which we want to find the longest common prefix be $\mathsf{c}^r\mathsf{bcb}$ and $\mathsf{c}^{r'}\mathsf{bab}$. Recall that these two sequences are extracted from some word w in $\mathcal{L}(U_\sigma)$, where $w(0)$ projects to the former sequence and $w(1)$ projects to the latter, by discarding occurrences of ε. Since supervisor 1 does not observe c, the encoding of $\mathsf{c}^{r'}\mathsf{bab}$ as $w(1)$ can be chosen to start with $\mathsf{c}^{r'}\varepsilon^r$. The resulting effect is that an arbitrarily large set of states of the form $(Right, c^{r''})$, with $r'' \leq r'$, can be reached. Hence the set of reachable states is infinite. This is clearly not desirable for the effective synthesis of a communication protocol, and we must somehow restrict U_σ so that it generates a relation that will reach only finitely many states of $\mathcal{J}_{\mathcal{B}}$.

5.2. A finite mind-the-gap automaton

Because co-observability is a language-theoretic property, we now study language-theoretic properties of U_σ. It will be useful to prove certain properties of U_σ so that we can perform the appropriate transformation on the input to $\mathcal{J}_{\mathcal{B}}$. In particular, we are interested in the step property of step transition systems (Stark, 1989) and their relationship to Mazurkiewicz trace languages (Mazurkiewicz, 1977) and, more precisely, generalized trace languages (Hoogers, Kleijn, & Thiagarajan, 1995).

Definition 5 (*Adapted From Stark (1989)*)**.** Given a deterministic transition system $\mathcal{M} = (Q, q_0, \mathbb{A}, \delta)$, where Q is the state set, $\mathbb{A}$ is an alphabet of labels, and $\delta: Q \times \mathbb{A} \to Q$ is the transition function. Transition system $\mathcal{M}$ satisfies the *step property* if it ensures that transitions with non-atomic labels can be decomposed into arbitrary sequences of decompositions of the label. Formally,

(Axiom 1) $\forall q_1, q_2 \in Q, \forall \ell, \ell' \in \mathbb{A}, \delta(q_1, \ell) = q_2$ and $\ell' \leq \ell \Rightarrow \exists q_3$ such that $\delta(q_1, \ell') = q_3$ and $\delta(q_3, \ell \setminus \ell') = q_2$.

Independent transitions originating from the same state may be merged:

(Axiom 2) $\forall q_1, q_2, q_3 \in Q, \forall \ell_1, \ell_2 \in \mathbb{A}, \delta(q_1, \ell_1) = q_2$ and $\delta(q_2, \ell_2) = q_3$ and $\ell_1 | \ell_2 \Rightarrow \delta(q_1, \ell_1 \vee \ell_2) = q_3$.

Fig. 6 illustrates the step property.

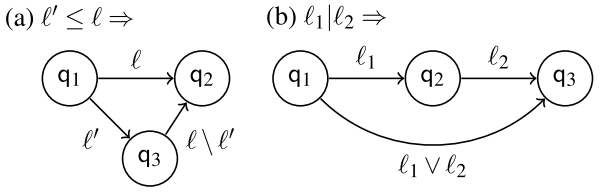

Fig. 6. The step property of deterministic transition systems: (a) Axiom 1 of Definition 5; (b) Axiom 2 of Definition 5.

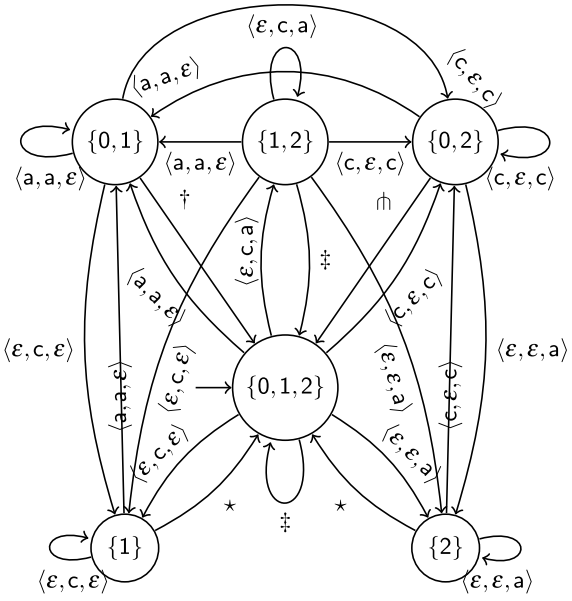

Fig. 7. Automaton N for the example from Fig. 1. Key: $\ddagger = \langle \mathsf{a}, \mathsf{a}, \mathsf{a} \rangle, \langle \mathsf{b}, \mathsf{b}, \mathsf{b} \rangle, \langle \mathsf{c}, \mathsf{c}, \mathsf{c} \rangle, \langle \sigma, \sigma, \sigma \rangle$; $\dagger = \langle \mathsf{b}, \mathsf{b}, \mathsf{b} \rangle, \langle \mathsf{c}, \mathsf{c}, \mathsf{c} \rangle, \langle \sigma, \sigma, \sigma \rangle$; $⫛ = \langle \mathsf{a}, \mathsf{a}, \mathsf{a} \rangle, \langle \mathsf{b}, \mathsf{b}, \mathsf{b} \rangle, \langle \sigma, \sigma, \sigma \rangle$; and $\star = \langle \mathsf{b}, \mathsf{b}, \mathsf{b} \rangle, \langle \sigma, \sigma, \sigma \rangle$.

The language of a transition system with the step property is a generalized trace language (Hoogers et al., 1995), generalizing Mazurkiewicz trace languages (Mazurkiewicz, 1977), and is defined as follows.

Recall that a congruence $\sim$ is an equivalence relation such that for all ℓ, ℓ', ℓ'', if $\ell' \sim \ell''$ then $\ell\ell' \sim \ell\ell''$. Let $\sim$ be the least congruence on $\mathbb{A}^*$ such that: $\forall \ell, \ell' \in \mathbb{A}, \ell | \ell' \Rightarrow \ell\ell' \sim \ell'\ell$ and $\ell \leq \ell' \Rightarrow \ell' \sim \ell \cdot (\ell' \setminus \ell)$. Let $\simeq$ be the least congruence on $\mathbb{A}^*$ such that $\forall \ell, \ell' \in \mathbb{A}, \ell \mid \ell' \Rightarrow \ell\ell' \simeq \ell'\ell$.

Definition 6 (*Adapted from Mazurkiewicz (1977) and Hoogers et al. (1995)*)**.** $D \subseteq \mathbb{A}^*$ is a *trace language* iff it is closed under $\simeq$. $D \subseteq \mathbb{A}^*$ is a *generalized trace language* iff it is closed under $\sim$: $\forall w \in D, \forall z \in \mathbb{A}^* \; w \sim z \Rightarrow z \in D$.

It is straightforward to check that U_σ has the step property and that its language is a generalized trace language, with relation $|$ as the independence relation.

Proposition 1. *U_σ is a step transition system.*

Proof. Since U_σ is the product of sequential components, Axiom 1 of Definition 5 holds since every non-atomic transitions can be decomposed as in Fig. 7(a).

Note that two transitions in U_σ of the form (q_1, ℓ_1, q_2) and (q_2, ℓ_2, q_3) such that $\ell_1 | \ell_2$ alter disjoint components of the state vector. Hence, there *is* a third transition $(q_1, \ell_1 \vee \ell_2, q_3)$, thus satisfying Axiom 2 of Definition 5. □

It is the case that the language of a step transition system is a Mazurkiewicz trace language, whenever labels are decomposed into arbitrary sequences of atoms. Then we can compute a normal form called Cartier–Foata Normal Form (CFNF) as presented in Cartier and Foata (1969).

Definition 7. A word $w = \ell_1 \dots \ell_k \in \mathbb{A}^*$, where $\mathbb{A}$ is the alphabet of labels, is in *Cartier–Foata Normal Form* iff $\forall i = 1, \dots, k-1, \forall \ell' \in A, \ell' \not\leqslant \ell_{i+1}$ or $\ell_i \not\parallel \ell'$.

Intuitively, words in CFNF correspond to greedy behaviors of the system, where no label belonging to step ℓ_{i+1} can be moved into step ℓ_i.

When the language of U_σ is in CFNF, the effect is to have performed a pruning of transitions in U_σ so that any sequence of the pruned U_σ will be in CFNF. Note that this pruning depends only on the elements in the alphabet of labels, and not on the state of the plant. We will show why this leads to a bound on the memory required to compute $\mathcal{J}_{\mathcal{B}}$.

Since $\mathcal{L}(U_\sigma)$ is a generalized trace language, we can restrict U_σ so that its language is in CFNF. This effectively breaks cycles in U_σ that could cause divergence of the size of reachable states of $\mathcal{J}_{\mathcal{B}}$ and leaves only states that require a bounded memory. This implies that only finitely many states of $\mathcal{J}_{\mathcal{B}}$ remain reachable when U_σ is in CFNF.

To compute the CFNF of U_σ we need to compute one further automaton, N, based only on labels, that recognizes words in CFNF: $N = (Q^N, q_0^N, \mathbb{A}, \delta^N, F^N)$, where $Q^N = 2^{I_0}$; $q_0^N = I_0$; $\mathbb{A}$ is the alphabet of labels; $F^N = Q^N$; the transition function $\delta^N \subseteq Q^N \times \mathbb{A} \times Q^N$ is defined as follows. A transition $\delta^N(q, \ell) = q'$ iff $\forall \ell' \in A$, $\ell' \leq \ell \Rightarrow (\|\ell'\| \cap q \neq \emptyset)$ and $q' = \|\ell\|$. Fig. 7 contains N for the example in Fig. 1.

We denote the CFNF of U_σ, the product of U_σ and N, by $\widehat{U}_\sigma$.

Theorem 1. $\mathbb{C}[\widehat{U}_\sigma]$ *has finitely many reachable states.*

Proof. States of $\mathbb{C}[\widehat{U}_\sigma]$ are tuples of the form $(q, q^{\mathcal{A}}, q^{\mathcal{B}})$, where q is a state of $\widehat{U}_\sigma$, $q^{\mathcal{A}} \in \{0, 1\}$ is a state of $\mathcal{J}_{\mathcal{A}}$ and $q^{\mathcal{B}}$ is a state of $\mathcal{J}_{\mathcal{B}}$. W.l.o.g. we can omit the single-state transducers $\mathcal{P}$ and $\mathcal{P}'$. For the sake of clarity, we will consider only labels of $\widehat{U}_\sigma$ and disregard the components of transition labels of $\mathbb{C}[\widehat{U}_\sigma]$ that do not correspond to $\widehat{U}_\sigma$. Recall that in U_σ events that are observable by the receiver (supervisor j) are synchronized with the occurrence of these events in the plant. Thus, these events are encoded by a label ℓ where $\ell(0) = \ell(j) \neq \varepsilon$. This means that states of $\mathbb{C}[\widehat{U}_\sigma]$ of the form $(q, 0, (\mu, z))$, where $\mu \in \{Left, Right\}$ and $z \in \Sigma^*$ contains observable events to the receiver (supervisor j), are not reachable in $\mathbb{C}[\widehat{U}_\sigma]$.

Assume that there exists an infinite path λ in $\mathbb{C}[\widehat{U}_\sigma]$ that causes a divergence of the state size (e.g., the length of z in $q^{\mathcal{B}}$). $\widehat{U}_\sigma$ is finite state; therefore, there exists a state q in $\widehat{U}_\sigma$ that is visited infinitely often by λ. Thus λ can be factored as $\lambda = \beta\rho\lambda'$ (for finite words β and ρ) such that $(q_0, 0, \top) \xrightarrow{\beta} (q, 0, (\mu, z_0)) \xrightarrow{\rho} (q, 0, (\mu, z_1))$, where $z_0, z_1 \in \Sigma^*$, $|z_0| < |z_1|$, and $\mu \in \{Left, Right\}$. In particular, word ρ defines a cycle in $\widehat{U}_\sigma$, as it returns $\widehat{U}_\sigma$ to q. This cycle can be iterated to form an ultimately periodic word $\beta\rho^\omega$ that also causes a divergence of the state sizes in $\mathbb{C}[\widehat{U}_\sigma]$. Thus $(q_0, 0, \top) \xrightarrow{\beta} (q, 0, (\mu, z_0)) \xrightarrow{\rho} (q, 0, (\mu, z_1)) \xrightarrow{\rho} (q, 0, (\mu, z_2)) \dots$, where $|z_0| < |z_1| < |z_2| < \cdots$, $\mu \in \{Left, Right\}$ and $\forall k \geq 0, z_k$ is unobservable to the receiver.

By symmetry and w.l.o.g., assume that $\mu = Left$. Given a word $\rho = \ell_0\ell_1 \cdots$ generated by $\mathbb{C}[\widehat{U}_\sigma]$, recall that $\rho(k) = \ell_0(k)\ell_1(k) \dots$ returns the kth component of each label in ρ. Inequality $|z_0| < |z_1|$ implies that in ρ, the plant (component 0) produces more events than the receiver (supervisor j) believes have occurred:

$$|\rho(0)| > |\rho(j)|. \tag{2}$$

W.l.o.g. assume that $|z_0| > |\rho(0)|$. This implies that ρ may not contain any observable events w.r.t the receiver (supervisor j). Thus, $\rho(j)$ and $\rho(0)$ are unobservable to the receiver. Inequality (2) implies that the last label of ρ commutes with one atom of the first label of ρ. Thus $\rho \cdot \rho$ is not in CFNF. This contradicts the assumption that every path in $\widehat{U}_\sigma$ is in CFNF. Hence no ultimately periodic path of $\widehat{U}_\sigma$ causes state size divergence. Therefore, there exists a bound on the size of states reachable in $\mathbb{C}[\widehat{U}_\sigma]$ and $\mathbb{C}[\widehat{U}_\sigma]$ has finitely many reachable states. □

6. Conclusion

We have defined the mathematical formulas required to calculate a finite state machine that defines a more comprehensive communication language for a specific class of decentralized discrete-event control problems (e.g., control with synchronous communication). In its current form, the corresponding computational complexity of our procedure is daunting; however, we are investigating efficient implementations of the mathematical formulas presented here for the construction of our communication language.

There have been several studies of communication with bounded and unbounded delay (e.g., Hiraishi (2009) and references therein); however, this literature assumes that *every* observation is broadcast by the observing supervisor. Regardless of the communication language, there is no guarantee that a solution exists in the presence of unbounded delay. We can use $\mathbb{C}[\widehat{U}_\sigma]$ to identify an appropriate communication sublanguage where the control objective is met under restricted conditions of either known finite delay or delay with a known upper bound, but this remains as future work.

While we can now calculate more than just the "first" and "last" possible places to communicate in the plant language, we offer no suggestions for which communication sublanguages will form more desirable communication protocols. At the conclusion of our procedures, we have, for all violations of co-observability for (u, v) w.r.t σ, a communication language that contains all sequences such that each identified communication allows the receiver to distinguish $u\sigma$ from $v\sigma$. At this point it is up to the designer of the communication protocol to choose a communication sublanguage based on properties of interest. These properties could include some notion of optimality (either quantitative or qualitative); however this discussion extends beyond the scope of the paper.

References

Arnold, A. (1994). *Finite transition systems*. Prentice–Hall.

Barrett, G., & Lafortune, S. (2000). Decentralized supervisory control with communicating controllers. *IEEE Transactions on Automatic Control*, *45*(9), 1620–1638.

Cartier, P., & Foata, D. (1969). Problèmes combinatoires de permutations et réarrangements. *Lecture Notes in Mathematics*, *85*.

Hiraishi, K. (2009). On solvability of a decentralized supervisory control problem with communication. *IEEE Transactions on Automatic Control*, *54*(3), 468–480.

Hoogers, P. W., Kleijn, H. C. M., & Thiagarajan, P. S. (1995). A trace semantics for petri nets. *Information and Computation*, *117*, 98–114.

Mannani, A., & Gohari, P. (2008). Decentralized supervisory control of discrete-event systems over comunication networks. *IEEE Trans. Autom. Control*, *53*(2), 547–559.

Mazurkiewicz, A. *Concurrent program schemes and their interpretation. Technical report*, Aarhus University DAIMI PB-78, 1977.

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

Ricker, S.L. *Knowledge and communication in decentralized discrete-event control. Ph.D. thesis*, Queen's University, 1999.

Ricker, S.L., & Caillaud, B. Mind the gap: expanding communication options in decentralized discrete-event control. In *Proc. 46th IEEE Conf. Decision Contr.*, pp. 5924–5929, 2007.

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

Rudie, K., & Wonham, W. M. (1992). Think globally, act locally: decentralized supervisory control. *IEEE Trans. Autom. Control*, *37*(11), 1692–1708.

Stark, E. W. (1989). Concurrent transition systems. *Theoretical Computer Science*, *64*(3), 221–269.

van Schuppen, J. H. (2004). Decentralized control with communication between controllers. In *Unsolved problems in mathematical systems and control theory* (pp. 144–150). Princeton University Press.

Wang, W., Lafortune, S., & Lin, F. (2008). Minimization of communication of event occurrences in acyclic discrete event systems. *IEEE Trans. Autom. Control*, *53*(9), 2197–2202.

Laurie Ricker received her Ph.D. in 2000 from Queen's University (Canada). She was an ERCIM postdoctoral fellow in 1999–2000. Since 2001, she has been at Mount Allison University, where she is an Associate Professor in the Department of Mathematics & Computer Science. She holds adjunct appointments at Concordia University (Department of Electrical & Computer Engineering) and McMaster University (Department of Computing and Software). Her research interests include distributed systems and the control of decentralized discrete-event systems.

Benoît Caillaud is a researcher at INRIA and head of the S4 embedded systems analysis team at INRIA-Rennes. He has published over 40 papers in refereed journals or conferences. His areas of interest include distributed and embedded systems design and hybrid systems modeling.